Exchange

Issue 1, 1987

Exchange of IBM PC Information

IBM

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

Editor	Michael Engelberg
Associate Editor/ Design Director	Karen Porterfield
Contributing Editor	John Warnock
User Group Editor	Sheryl Cragun
Illustrator	Jeff Jamison
Automation Consultant	Sherry Reardon
User Group Support Manager	Gene Barlow

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

IBM PC User Group Support
IBM Corporation (4704)
P.O. Box 3022
Boca Raton, FL 33431-0922

To correspond with *Exchange*, please write to the editor at the above address.

© 1987 International Business Machines Corporation.
Printed in the United States of America.
All rights reserved.

Articles in *Exchange of IBM PC Information* are written by IBM employees and members of PC user groups. Articles written by members of PC user groups which contain an individual copyright notice and articles written by IBM employees cannot be reprinted without the permission of the editor of *Exchange* and the permission of the author. Other articles written by members of PC user groups can be reprinted by PC user groups provided that credit is given to the author, the author's user group, and *Exchange*.

Groups and individuals seeking permission to reprint should direct their inquiries to the editor of *Exchange*.

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 does not guarantee that the programs contained in this publication 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.

The name of the author of the article "Change the Prompt When You Drop into DOS," published in the November/December 1986 issue of *Exchange*, was given incorrectly. The correct name of the author is Daniel Ehrmann, president of the Chicago Computer Society. We regret the error.

User Group Support
(305) 998-PCUG
User Group (Voice) Locator System
(305) 998-1575
Electronic Bulletin Board System
(305) 998-EBBS

Set your program with the following parameters:

1. 300, 1200, or 2400 baud.
2. 8 data bits, no parity, one stop bit (for binary data transmission).
3. Host echo (full duplex) can be changed to local echo (half duplex) for a noisy line.

Trademarks

Crosstalk is a trademark of Microstuf, Inc.

Electric Desk is a trademark licensed to Alpha Software by Electric Software, Inc.

Epson is a trademark of Epson America, Inc.

Hayes Smartmodem 1200 is a trademark of Hayes Microcomputer Products, Inc.

Microsoft is a registered trademark of Microsoft Corporation.

Norton Utilities is a trademark of Peter Norton Computing, Inc.

Quadram is a trademark of Quadram Corporation.

Statistical Analysis System is a registered trademark of SAS Institute, Inc.

Hard Disk Data Recovery

Jack K. Wright
Princeton IBM-PC Users Group

There are many ways to lose data on your hard disk—accidental erasure, bad sectors, and damage to the "map," which is the area that contains the directory and File Allocation Table (FAT) information.

The first way, accidental erasure, is fairly easy to reverse by using one of the many unerase utilities now available. DOS 3.00 makes it even easier by using a new scheme of allocating space on hard disks, thus increasing the chances that erased files will not be written over by new files as quickly as in earlier versions of DOS.

Bad sector problems are the most serious—you may be able to recover most of the file, but the part that resides in the bad sector is usually impossible to read again. Bad sectors in the directory can make it impossible to access anything on the disk, but all your files are still there and can usually be recovered with the right procedures.

The third way to lose data, map damage, is the most confusing, but is not always as big a catastrophe as it seems. Your hard disk has map damage if the DOS CHKDSK program gives you one or more error messages such as "File.xyz is cross linked on cluster nn," "File.xyz - allocation error, size adjusted," or "XX lost clusters in YY chains."

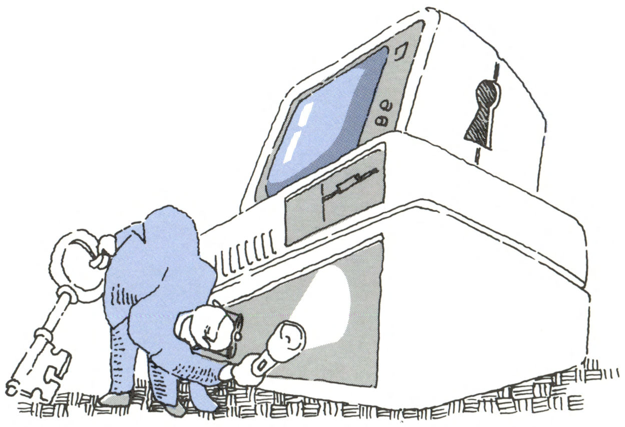
Unfortunately, there are no easy recipes for file recovery involving map damage, but I will discuss some important cautions to prevent making the file recovery more difficult. Sometimes a situation that initially seems hopeless can be turned around by the skillful use of the right utility programs, such as IBM's Professional Debug Facility and DOS's DEBUG, among many others.

You should run CHKDSK often on your hard disk (*without* the /F option) to catch map damage problems right away. If several of these problems accumulate over a period of time, the combination of CHKDSK errors can be much more difficult to analyze and repair than each problem by itself.

CHKDSK /F

Getting a lot of CHKDSK error messages can be very scary; the typical reaction is to promise ourselves that tomorrow we will definitely start backing up the hard disk every day. The only clue about what to do is in the brief discussion of the CHKDSK /F option in the DOS manual. Although the F in that option stands for FIX, the way it "fixes" your hard disk may not always be what you had in mind. There is a reason that the /F is an option—it is not always the correct thing to do when you are faced with an array of CHKDSK error messages. In some cases, using /F can make your data even more difficult to recover, as I will illustrate later. Also, using CHKDSK /F with a multitasking program, such as TopView, can interfere with temporary files created in other windows. *Never* put CHKDSK /F in your AUTOEXEC.BAT file; however, just plain CHKDSK is okay.

In my first example, I show where CHKDSK /F *should* be used. Suppose you are using a data base program, you have added a sequence of new records, and you've noticed some disk activity indicating that the new records have been stored to the disk. Now the power goes out while you are still adding records to the data base. When it returns, you run a CHKDSK on the data base's data file and get just one message: "Allocation error, size adjusted." The "allocation error" message means that the file size in the directory does not agree with the file size in the FAT. The reason this happened is that, during the time you were entering data, updates were made in the FAT and were forced out to disk; however, the directory was not updated before the power loss, because the file had not been closed. The filesize in the directory was therefore incorrect.



But the size of your data file will not actually be adjusted unless you include the /F parameter. In this case you should specify /F, because CHKDSK /F will adjust the file size in the directory to match the allocation shown in the FAT. This way you stand a good chance of recovering most of the file's data.

Lost Clusters and Cross-Linked Files

Many people assume that, if CHKDSK gives messages about lost clusters and cross-linked files, it means their FAT has been destroyed. This is not always the case. These types of problems can be caused when the FAT is perfectly normal but there is a problem in the directory. This can happen because the FAT actually starts in the directory—each directory entry has a word that starts the FAT chain for that file.

The following example shows what can happen with just a one-byte change in a file's directory entry or by a one-byte change in the FAT. Running CHKDSK shows the following:

```
C:\FILE1.TXT
 Allocation error, size adjusted.

XX lost clusters found in YY chains.
 Convert lost chains to files (Y/N)?
 xxxxx bytes disk space
       would be freed.

FILE1.TXT
   is cross-linked on cluster nn.
FILE2.TXT
   is cross-linked on cluster nn.
```

Here the FILE1.TXT entry is mistakenly pointing into the FILE2.TXT storage (or allocation) area, due to the one-byte error in FILE1's directory entry. When CHKDSK compares FILE1's directory filesize with its now-faulty FAT allocation chain, it discovers a discrepancy and gives the "allocation error" message. Nothing is now pointing to the original FILE1.TXT FAT allocation chain, so CHKDSK sees that allocation chain as "lost clusters." But there is nothing at all wrong with FILE2.TXT.

Looking at all these messages, it does not seem possible that just a one-byte fix in the directory would completely fix this problem. But, as I mentioned earlier, that's all that is required to repair this disk.

Now notice what happens if you run CHKDSK /F. It will change the filesize of FILE1.TXT in the directory to correspond to the new length of the *invalid* allocation chain (*without* asking you first). If you then answer the question "Convert lost chains to files?" with No, it will *erase* the "lost clusters." Once a file allocation chain is erased in this way, it *cannot* be easily recovered with most "unerase" utilities, because there is no directory entry tied to the cluster chain. You have just made FILE1.TXT much more difficult to recover.

However, if you instead answer the question "Convert lost chains to files?" with Yes, it will create a new file called FILE0000.CHK. If the cause of all the error messages was only in the directory, as discussed above, FILE0000.CHK is your original FILE1.TXT. But if the problem was in the FAT, FILE0000.CHK will represent only part of FILE1.TXT, and CHKDSK /F has made the rest of it more difficult to recover.

The Second FAT

Another technique is to use the second copy of the FAT. DOS keeps two copies of the FAT on the disk, so this may initially seem to be a promising solution for repairing the original FAT. Unfortunately, the second FAT will usually be just as messy as the original FAT. This is because DOS updates the second copy any time it writes to the first one. If an error has caused it to write garbage to the first FAT, it will write the same garbage to the second one. The second FAT is really designed to be helpful in cases where the first FAT develops a bad sector so that it is unreadable. When that happens, DOS automatically reads the second FAT, and you will usually not even be aware that there is a problem.

There is, however, a way to determine if FAT 2 is being used. If CHKDSK finds that FAT 1 is bad, it prints the error message "Disk error reading FAT 1," then continues with its disk checking using the good FAT 2.

Cross-Linked Files

Earlier I gave an example involving cross-linked files. That problem was repairable by simply changing one byte in the directory. Cross-linking is usually more difficult to repair than that, but it can be made easier by keeping in mind a few general rules.

The first rule is that, whenever two files are cross-linked, one of them is almost always undamaged. Also, there is usually a chain of lost clusters reported, and this chain usually belongs to the damaged file. CHKDSK will tell you where the cross-link occurs, and then the file up to that point can be re-linked to the lost clusters to recover the whole file. This can be done with a FAT editor, such as the one in IBM's Professional Debug Facility.

Another rule to keep in mind is that you should *not* delete either of the cross-linked files before attempting recovery, or you may lose part of the good one. Copy them both to another disk before attempting the repair.

One thing that makes file recovery so confusing is that when we talk about file damage as shown by CHKDSK, such as cross-linked files, we really are not talking about damage to the file itself. The file is still intact. The damage has occurred to the information in the directory or the FAT that tells us where the file is. Thus, even with extensive "file damage" of this type, we still have a chance of completely recovering the file. In addition, even if the root directory is wiped out, much of the subdirectory information may still be available, because it is kept in a different area of the disk.

Map Damage is Not File Damage

A key point to remember when you are faced with map damage is that map damage does not always mean file damage. Usually only the map (the FAT and directory) has been affected, and if the file contains text data, it can sometimes be recovered.

Your first attempt to recover lost files on your hard disk should focus on repairing the map damage. If this is impractical, you can still search for the file among the thousands of data clusters on the disk, and piece it back together. This is not much fun, but there do exist a few tools and a few tricks that make it easier. The approach to take depends on how extensive the damage is.

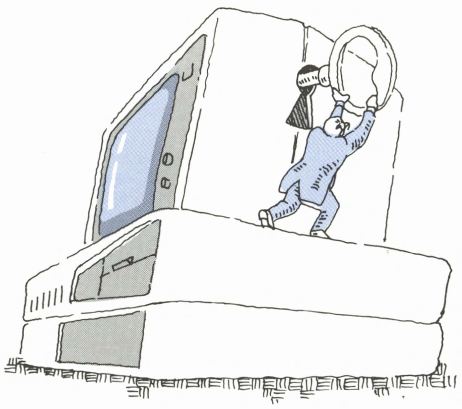
Keyword Scanning

If the directory is in reasonably good shape, it contains a pointer to the first cluster of the file. So, even if the FAT is completely gone, you can still find the first cluster of the file. If the directory is too damaged, or unreadable due to a bad sector, you are still in luck if the file was in a subdirectory. If you know the file name you are trying to recover, you can search for the subdirectory cluster using this name. Once the subdirectory cluster is found, it will have the pointer to the first cluster of the file, and also information telling you the length of the file, so you can calculate how many clusters to recover.

Another technique that can be used here, once you've found your file clusters, is to accumulate them in memory and then write them out to another disk using the DOS DEBUG program.

Beware the RECOVER Command

In the case of total directory loss (but undamaged FAT), DOS has provided the RECOVER command. This is actually two very different programs in one—"RECOVER d:" and "RECOVER filename." The RECOVER filename version is for the case where one of your files has developed a bad sector, and it will be discussed later.



RECOVER d:, where d: is the drive to recover, assumes that the FAT is undamaged, and attempts to make the best of that information to recover your files. But RECOVER d: also assumes the entire directory has been wiped out or has been overwritten by another disk's directory. Because of what the RECOVER d: process will do to your disk, it should be used only if you are certain that your root directory was *completely* wiped out, and if you are reasonably sure that the FAT was *not* damaged. Because of the difficulty of determining this, RECOVER d: should not be used by the average PC user who is not technically oriented. Also, RECOVER will not help if your directory has bad sectors that are unreadable.

What will happen if you use RECOVER d:? It will completely obliterate any information that was in your root directory, since it assumes there was nothing of value there. It will change all the subdirectory clusters (the ones that hold the subdirectory information) to anonymous files. All files in subdirectories will be converted to root directory files, up to the limit of the root directory capacity. When it is done, your directory will consist of nothing but files of the form FILEnnnn.REC, where nnnn is a number that is incremented for each file. If your FAT really was in good shape, your files will be there, and you will now have to find them and rename them. You also will have to sort out the subdirectory clusters and change their attribute to the subdirectory attribute, using a disk utility program or DEBUG. Rename them to the subdirectory first.

Once you've gotten this far, if you run CHKDSK, you'll find you have quite a few cross-linked files because the subdirectories you've recovered and some of the FILEnnnn.REC files will both think they own the same areas of the disk. This situation can be salvaged with care, but it's messy. A better way is to ignore the RECOVER d: command entirely and to use a program that searches for the subdirectory clusters on the disk; then manually reconstruct the root directory to point to these clusters. Next, run

CHKDSK /F to collect any remaining lost FAT clusters into files in the root directory. The FILEnnnn.CHK files produced by CHKDSK /F should be the rest of the files that were in the root directory. You'll have to identify and rename them.

Use "RECOVER filename" when you can't read a file because it has a bad sector. The term "bad sector" is used here to mean a disk sector that cannot be read correctly or cannot be read at all. You will know the file has a bad sector if you get the message

```
General Failure error reading drive C:
Abort, Retry, Ignore?
```

RECOVER filename reads all the sectors that are still readable, marks the bad sectors' clusters as bad in the FAT, then recreates the original file from the readable clusters. Once a cluster has been marked bad, DOS will not use it for data again. If you use RECOVER in this way, the resulting file will have missing pieces that are 512 characters long where the bad sectors were. If the original file is a text file, filling in those gaps should be a lot easier than starting over. If it is not a text file, the resulting file is probably going to be unusable. But at least RECOVER will prevent you from using the bad sectors again.

There are several types of bad sectors. For example, if the bad sector is a "bad ECC" sector (discussed later), you may be able to retrieve at least some of it. Thus you should not use RECOVER until you have investigated that possibility.

Files and Sectors

Each file on the disk is made up of a series of sectors. Each sector is 512 bytes, or characters, in size. A file that is 2KB in size theoretically occupies 4 sectors on the disk, but actually occupies 8 cylinders. This is because the smallest unit of allocation on a 10MB disk is 8 sectors, or 4KB, and is called a cluster.

Any sector can turn bad with little warning. A clean report by CHKDSK does not mean there are no bad sectors on the disk. CHKDSK does not check the entire disk; it checks only the directory and FAT area. So the first hint of a problem is usually the "Abort, Retry, or Ignore" message that appears when the file containing the bad sector is used.

Most hard disks greater than 10MB in size have bad sectors. These sectors were detected when the disk was formatted, and were marked at that time so that DOS won't use them. Whenever new bad sectors show up, they also can be marked as unusable, either by using DOS's RECOVER filename command, by formatting the disk again, or by using a special utility program designed for the purpose. But be wary of public-domain versions of this type of utility, because bugs are common in many of them. And don't mark the sectors bad until you've tried the suggestions below to recover them. Once the sectors are marked, DOS can no longer access them.

To determine whether any files contain bad sectors, you can scan your disk by running the command COPY *.* NUL in each of the directories on the disk. When it gets to the file with the bad sector, you will see the "Abort, Retry, Ignore?" message under the file name. Disk-scanning utilities are faster and will check the entire disk, but if the bad sector is in a file, they still won't tell you *which* file.

Retry Techniques

DOS can respond to a bad sector or sectors in several ways. The four messages below indicate the major problems with bad sectors, in order of seriousness. In all cases you should retry the operation several times. If the errors occur when the system is cold, try again when it has warmed up. If they occur when the system is warm, turn it off and try again later as soon as the system is turned on. If your system is operating on its side, try putting it on a table. If you finally manage to read the bad sector(s), copy the file to another disk. Then try to get the bad sectors to show up again and use DOS's "RECOVER filename" command to mark them so they won't be used again. If the bad sectors are in the directory, however, RECOVER won't help. This is the case if you get one of these error messages when you try the DIR command on the disk. I'll discuss this later.

The major bad sector error messages are:

1.
```
Seek error reading drive C
     Abort, Retry, Ignore?
```

   This message means that DOS can't even find the requested track (cylinder). The disk may need to be low-level formatted again, or the disk controller or the disk itself could be bad. Try to remember where you put the backup disks. It's probably still worthwhile to retry the operation under different conditions as suggested above. Fortunately, this error is rare.

2.
```
Disk error reading drive C
            (DOS 2.x)
     Abort, Retry, Ignore?
```
   or
```
General Failure error reading drive C
            (DOS 3.x)
     Abort, Retry, Ignore?
```

   Here DOS can't find the formatting for the requested sector (more technically, "Address mark not found"). You are not in much better shape than you were above, although if this error did not occur after a DIR command, you may have a chance of recovering most of the file using the "RECOVER filename" command.

3.
   ```
   Sector not found error reading drive
   C
        Abort, Retry, Ignore?
   ```

   This message means that DOS can read the basic sector formatting, but can't read or find the right sector number there. Therefore DOS can't read the sector data it is looking for. Give this one lots of retries under different conditions before resorting to RECOVER.

4.
   ```
   Data error reading drive C
        Abort, Retry, Ignore?
   ```

   This is the least serious of the four errors. If we're lucky, only a few bytes in the sector are bad. This error message says that DOS can read the entire sector, but the Error Checking and Correction code (ECC) stored with the sector says that some or all of the data is not valid, so DOS stops reading the file. This is usually due to a bad spot on the disk. The ECC code on a hard disk is analogous to the CRC code on a floppy. They both detect errors in data integrity, but the ECC code, being bigger, also can provide a small amount of error correction. If you get this error, it means the small amount was not enough.

If you get a Data Error message, *some* of the data may still be valid. If it's a text file, it is usually worthwhile to try to salvage it. Let's look at how to do this.

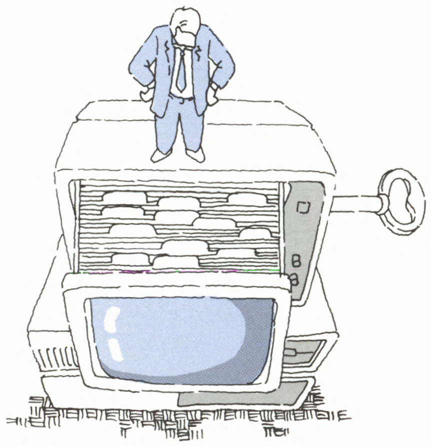
Salvaging "Data Error" Files

Suppose the Data Error message appeared during a file copy: COPY C:FILE1 A:. If you choose the "Ignore" option, DOS will continue the copy, ignoring the error. However, the destination file will usually have many more faulty sectors of data than there were bad sectors on the disk. This is because files are copied in groups of sectors, and an error anywhere in the group will abort the transfer of the rest of the group. So the file copied to A: will very likely contain the bad sector followed by a number of sectors that contain garbage.

What is really needed is some way to copy all the sectors one by one to a new file, including the bad sector(s). Then a text editor can be used to fix up the parts of the bad sector that caused the error.

It's possible to do this one-by-one sector copy with the DOS DEBUG program, but it is very tedious, especially if the file is large. Looking for an easier way, I wrote a short program, IGNORE.COM, that is now available as a public-domain program on some bulletin boards. IGNORE fools DOS into ignoring the data error and doing the one-by-one sector copy automatically. If I run into a file with a data error, I install IGNORE, then copy the file to another disk. I run IGNORE again to return DOS to normal. Next, I edit the copied file to fix up the bad spot, and run DOS's "RECOVER filename" command on the original file, which marks the bad sector(s) so that DOS won't use them again. Finally I copy my fixed-up file back to the original disk.

If you don't use RECOVER to mark the sector bad, and then later write a file to the same area of the disk, you will usually find that the former bad sector now seems to be good. Most people, however, prefer to mark it bad rather than risk their data again to a known borderline sector.



Bad Sector in the Directory

If you get a bad sector error when trying to run the DIR command, it means the bad sector is in the directory. In that case, if you're skilled with disk utilities, you can try to recover as much as possible using the information in those FAT and directory sectors that *are* good, using some of the techniques I discussed previously. The next step is to delete the DOS partition and try to re-partition with FDISK, moving the new DOS partition to another area of the disk, away from the bad sector area. You do this by specifying a starting cylinder other than cylinder 0 (i.e., don't accept FDISK's default for the starting cylinder).

Remember that, in attempting recovery of data from a damaged disk, one slip can quickly make the situation much worse than it was. The same caveats that applied to DOS's RECOVER command apply also to other commercial programs that are designed to do the same type of recovery. Operate on a copy of the data if possible, or enlist the help of a knowledgeable expert.

Conclusion

My main point throughout this article is that file recovery is not always as hopeless as it may seem. Although in many cases it may demand much time and effort, it may still be much less costly than the time and effort needed to create the file from scratch. On the other hand, it's a lot more difficult than backing up regularly.

Keypad 5

Chuck Gaston
Poughkeepsie IBM Club Microcomputer Club

There are many reasons to want to use the center (5) key on the numeric keypad. A game might use the four arrow keys to initiate movement, and the center key to stop it. In a word processor or similar program, the directional keys might produce larger movements when preceded by the center key. Sometimes you just want one more function than the number of keys available.

Often I have been asked if there is any way to recognize the center key of the numeric keypad. Before now I have always said "No." But I am glad to report that this previous advice was wrong. BASIC releases 2.0 and later extend the ON KEY(n) statement to allow defining, in addition to the regular function and cursor keys, six special keys to be trapped. These extra keys are defined in terms of keyboard scan codes rather than as the ASCII or Extended ASCII codes found in Appendix D. This means that some rather exotic key definitions are possible, including that infamous keypad 5 and a combination like Ctrl-Spacebar. The program listing below shows these two definitions.

Although the ON KEY(n) statement *allows* extra key definitions, it is the KEY n, CHR$(KBflag)+CHR$(scan code) statement (as in lines 10 and 20) that actually does the magic. In the case of these special trapped keys (15 thru 20), it requires a shift code (almost any combination of the various shift states) and a keyboard scan code. I have not begun to explore all the possibilities, but I believe they include uniquely identifying the numeric keypad's plus and minus keys, and distinguishing six different shift combinations with the Enter key.

Often my programs do most of their keyboard handling in a single subroutine, using INKEY$ to pick up each keystroke. I can imagine using this key-trapping feature to simulate normal key input. The trap subroutine could set a special flag so that the normal keyboard handler would not look for INKEY$, but instead would return a special two-character string representing the trapped key. For example, 0-75 is cursor left and 0-77 is cursor right; 0-76 could and should represent the unshifted keypad 5.

Sometimes logic, consistency, convenience or other factors call for the use of a few keys or key combinations not part of the regular or extended ASCII sets. Those few gaps in the normal key recognition routines can be closed by using the "KEY n, X$" and "ON KEY(n) GOSUB" statements.

```
10     KEY 15, CHR$(0)+CHR$(76)           'Keypad 5 is scan code 76
20     KEY 16, CHR$(4)+CHR$(57)           'Ctrl is shift 4; spacebar is code 57
30     ON KEY (15) GOSUB 1500             'Set up key trapping
40     ON KEY (16) GOSUB 1600
50     KEY (15) ON                        'Turn trapping on
60     KEY (16) ON
100    WHILE X$<>CHR$(27)                 '[Esc] key ends program
110      X$=INKEY$                        'Get keypress
120      IF X$<>"" THEN PRINT X$; " ";    'Print "ordinary" keys, spaced
130    WEND : END
1500   PRINT "[Keypad 5] ";: RETURN     'Subroutines show key recognition
1600   PRINT "[Ctrl]+[Space]";: RETURN
```

PCjr Ins and Outs

LeRoy Tabb, Jr.
Philadelphia Area Computing Society

A frequent question about the PC*jr* is "What is the difference between a PC*jr* and its full-grown big brother?"

The differences can be listed briefly. The PC*jr* does not have direct memory access (DMA); its video RAM is not located in memory block B0000 or B8000 hex; it cannot accept an 8087 Math Coprocessor; it has two slots for ROM cartridges; and it has only one disk drive and no expansion slots (but can be upgraded by using sidecars). Also, it has a larger allocation for video RAM and uses the TI SN76469N sound generator chip. Got it? If not, join the crowd, but I'll try to shed some light on all of this.

The PC*jr* does not have DMA. This means that, unlike its big brothers, the PC*jr* does not have a DMA controller (the 8237A) to allow the diskette drive to interact directly with memory. Therefore, diskette input/output must be handled by the 8088 CPU through a process called bit nibbling. The most apparent result of this difference is that you can't type ahead while the disk drive is being addressed. The most important result is probably that the CPU has to suspend running our programs while it writes to or reads from the diskette drive, so our PC*jr* is a bit slower than the rest of the family. The fact that the PC*jr* doesn't use DMA may also be a point of potential incompatibility with some software written exclusively for the PC (but that's a rare problem).

In my mind, the most important difference between the PC*jr* and the PC is the location of video memory. Before we go on, however, let's look at how the PC*jr* and the PC use their memory. To understand this, let's digress one step further and begin by talking about the brain and heart of the IBM PC, XT and PC*jr*—the 8088 microprocessor.

The 8088 is a 16-bit processor, and through some trickery called segmented addressing, is capable of addressing up to 1,024K bytes, or one megabyte, of memory. But not all of that memory is available for our use. To see exactly how IBM uses that memory, let's pull out a map. (See Figure 1.)

```
                  IBM MEMORY MAP

 F0000    IBM ROM BIOS, ROM BASIC, etc.

 E0000    ROM Cartridge
 D0000      Programs

 C0000    Hard Drive

 B0000    PC Monochrome
 B8000    PC Color Graphics
 A0000    PC Enhanced Graphics

 90000                    \
 80000                     |
 70000                     |
 60000                     |
 50000                     |
 40000                     >  Working
 30000                     |      memory
 20000                     |
                           |
 10000    PCjr             |
 00000    (128K)          /
```

Figure 1. IBM Memory Map

Figure 1 is not a traditional map, but a map of how the 8088's memory capacity is used by IBM computers. Each line in the memory map (00000 through F0000) represents the starting hexadecimal address for a 64K block of memory. Each block of memory has a specific application. Blocks 00000 thru 90000 are used for working memory. That's what we mean when we speak of a computer's memory. These blocks represent the memory available for our programs; they add up to the 640K maximum amount of memory in the PC, XT, and PC*jr*. Above that, block A0000 is used for IBM's enhanced video options, the Enhanced Graphics Adapter and the Professional Graphics Adapter. Continuing to climb the map, we find block B0000 which, as I mentioned earlier, is used for the video on the PC and XT. Our PC*jr* does not use that block of memory. Still climbing, block C0000 is used for the hard disk controller on the XT.

The next two blocks (D0000 and E0000) are of particular interest to PC*jr* users, because they are addresses into which we plug our ROM cartridges. This area is unused by the rest of the IBM family. Finally, we come to the top of memory, block F0000, where all of the built-in ROM resides. That includes ROM BASIC (remember, Cartridge BASIC would be in either block D0000 or E0000); the built-in diagnostics; the ROM BIOS; and the ever-popular Keyboard Adventure! This top part of memory holds a lot of interesting things, but they will have to wait to be explained in a future article.

We've seen that block B0000 is not used by the PC*jr*. Well, if our video isn't stored there, where *is* it stored? As you might guess, the PC*jr* allocates a part of our normal RAM memory for video. The highest part of RAM on a standard PC*jr* is reserved for the video. In text mode, 16K of our memory is allocated to video; in graphics mode, 32K is allocated out of RAM.

The PCjr is extremely compatible with its big brothers.

There are several RAMifications (!) of this arrangement:

First, some of our program memory is used up for video. Therefore, some programs that can run on a 128K PC don't have enough room to run on a standard PC*jr*.

Second, when IBM originally released the PC*jr*, they must have felt no one would want to expand it beyond 128K. So they placed the video RAM at the top of the 10000 block of memory. That caused a separation between the first 128K of RAM and the rest of the working memory (20000-90000), and that meant our programs wouldn't know where to find additional memory, even if we did find a way to install it. Well, as we all know, ways were found to install additional RAM and to get around the gap in our working memory. Getting around that gap is the primary purpose of the CONFIG.SYS files used with IBM's expanded memory sidecar.

The third problem associated with the location of the PC*jr*'s video RAM is that it is physically located at an address different from the one expected by PC software. (This is, by far, the biggest source of incompatibility between the PC*jr* and the PC.) In other words, if a program goes to an address in block B0000 on our PC*jr*, it would find no one home. It would be like trying to send a letter to an address that doesn't exist. Fortunately, the post office can be instructed to forward such mail to an alternate address. In the case of the PC*jr*, that post office is called a Video Gate Array (VGA). The VGA traps any instructions aimed at the B8000 memory area (where the PC Color/Graphics card would be; the Color card is at B8000H and the Monochrome card is at B0000H) and redirects them to the area of memory that we are actually using. This all works fine, and software that plays by the rules will have no problem running on our PC*jr*s. The problems come when programs attempt to circumvent the system. Using the letter example, suppose you are in a hurry to have it delivered, so you deliver it personally. Surprise! Not only is no one home, there isn't even a house in the area. Well, some software programs, typically arcade-style games that rely on speed, behave this way. Those programs make direct calls to the addresses they need, thereby bypassing our VGA postman. Finding themselves somewhere in the netherworld, they crash. Almost all the arcade-style games produced before the relaese of the PC*jr* will not run on our machine. Fortunately, after the release of the PC*jr*, many software developers recognized the problem and wrote versions of software that would also run on the PC*jr*.

An advantage of the PC*jr*'s use of working RAM for the video memory is that it allows us to allocate 32K of video RAM rather that the 16K available on the PC. This gives the PC*jr* a better color graphics capability than the standard PC.

Another of PC*jr*'s advantages is that it uses the TI SN76469N chip for generating sound. This allows the PC*jr* to play music in three-part harmony. The PC*jr* also can be made to synthesize human speech!

Don't panic after reading all of this! In spite of its differences, the PC*jr* is extremely compatible with its big brothers. After I expanded my PC*jr* to 640K, I found minimal compatibility problems. As previously mentioned, incompatibilities came from older arcade-style games (the newer ones are better anyway) and from large-scale business-oriented software that insists on using lots of memory and two disk drives. If you add a second drive, you'll reduce these problems too.

I've touched on a few of the most important functional differences between the PC*jr* and its bigger brothers. There are other differences (the keyboard, for instance). If you're interested in doing more reading on your own, I can suggest three books by Peter Norton: *Discovering the IBM PCjr Home Computer*, *Exploring the IBM PCjr Home Computer*, and *Programmer's Guide to the IBM PC*.

Putting PDS Graphs to Work

John Warnock
IBM Corporation

This article will show you how to use IBM Personal Decision Series (PDS) Graphs to map results and display them in graph form. I will use the recent opinion survey sent out to *Exchange* readers as an example.

Asking Your Opinion

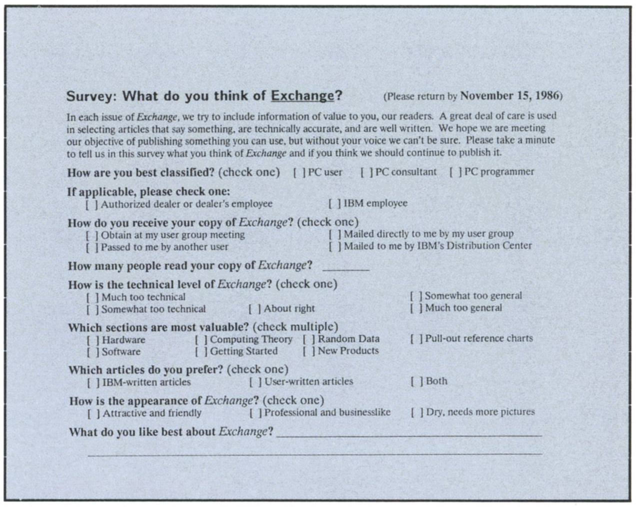
The survey (see Figure 1) consisted of 16 questions: 10 multiple-choice, two numeric answers, and four fill-in-comments. It was printed on one side of a return-mail form and sent to user groups for distribution at meetings and to dealers with their copies of *Exchange*. (*Editor's note: Results of the survey can be found in this month's editorial on the inside back cover.)*

Entering Your Responses

Given the thousands of responses we received, we attempted to tally and report the responses in the most efficient way possible. We used Personal Editor 2 to enter survey responses. Each survey became one line in an ASCII file.

To enter data quickly, we numbered the responses to multiple-choice questions so that nine of the responses could be entered with a single keystroke. For example, the response from a programmer (3), who was neither a dealer or employee (0), received *Exchange* at a user group meeting (1), shared his or her copy with six people (6), and thought the technical level was about right (3) became 3016 3 in our response file.

We allowed two digits for the number of readers per copy and four digits for any dollar value people wanted to place on *Exchange*. We asked which of seven sections you found valuable, and recorded those responses as x's in each of seven columns. Then we recorded your comments in abbreviated form.



Survey: What do you think of <u>Exchange</u>? (Please return by **November 15, 1986**)

In each issue of *Exchange*, we try to include information of value to you, our readers. A great deal of care is used in selecting articles that say something, are technically accurate, and are well written. We hope we are meeting our objective of publishing something you can use, but without your voice we can't be sure. Please take a minute to tell us in this survey what you think of *Exchange* and if you think we should continue to publish it.

How are you best classified? (check one) [ ] PC user [ ] PC consultant [ ] PC programmer

If applicable, please check one:
[ ] Authorized dealer or dealer's employee [ ] IBM employee

How do you receive your copy of *Exchange*? (check one)
[ ] Obtain at my user group meeting [ ] Mailed directly to me by my user group
[ ] Passed to me by another user [ ] Mailed to me by IBM's Distribution Center

How many people read your copy of *Exchange*? ________

How is the technical level of *Exchange*? (check one)
[ ] Much too technical [ ] Somewhat too general
[ ] Somewhat too technical [ ] About right [ ] Much too general

Which sections are most valuable? (check multiple)
[ ] Hardware [ ] Computing Theory [ ] Random Data [ ] Pull-out reference charts
[ ] Software [ ] Getting Started [ ] New Products

Which articles do you prefer? (check one)
[ ] IBM-written articles [ ] User-written articles [ ] Both

How is the appearance of *Exchange*? (check one)
[ ] Attractive and friendly [ ] Professional and businesslike [ ] Dry, needs more pictures

What do you like best about *Exchange*? ________________________________
____

Figure 1. ***Exchange*** **Survey Form**

Making Sense of Your Responses

As we collected more and more responses, we needed a way to show the statistical proportions. We divided the respondents into five groups: users, consultants, programmers, dealers, and IBM employees. We also looked at the combined results.

We wrote a BASIC program to tally responses for each audience and printed the results as percentages. This worked well, but we wanted graphs because they are easier to read, and have a greater visual impact than a statistical report. We selected PDS Graphs because it let us use input from a PDS Data file, and we could set up procedures to let it run unattended.

PDS is a family of products that act as a database manager, report writer, spreadsheet generator, host communication facility, and graphing facility. The Data Edition is the base PDS product. We added the Graphs Edition to do charts. We'll show the PDS menus and screens we used to graph the survey results.

Counting for PDS

With some minor modifications, the program that counted and printed a report became a program that counted responses and wrote the results to a file that PDS could read (SVYGRAPH.BAS). This saved us from having to enter summary data every time we wanted a current graph. Figure 2 shows the segment of the program that counted dealer responses.

We tried to relate variable names with the audience and the question. All user variables start with the letter U. All programmer variables start with the letter P. IBM employee variables start with the letter E. The number of the variable relates to the order of the possible choices. The first variable is always used to count non-responses.

The variable for counting dealers whose source of *Exchange* is another reader becomes DSE(3). The variable for counting users who think the technical level is about right is UTL(4).

The program starts at the first two characters in the file, determines the audience, and branches to the appropriate set of counters. The program converts the numbers to values as it scans each line, and adds 1 to the appropriate counter. When questions went unanswered, we entered zeroes and eliminated them from the results.

Once responses to all questions were counted, it was simple to add the individual counters together to get a combined set of responses. (Figure 3 shows the BASIC statements we used.) Master variables began with the letter M. Where there was a possibility of dividing by zero, we tested and branched around that calculation.

We added up the additional number of readers, and divided by the number of responses to get the average numbers of readers for each group. Then, we added up the number of respondents who put a dollar value on *Exchange* and divided the total into the total value to get an average value.

Next, we created a fixed-length file in the LIB1 directory of PDS. We created a record for each audience type (the variable B$) and each question topic (the variable C$). Later, we used these two fields to point PDS to specific records.

```
10 REM SAVE "SVYGRAPH.BAS",A
30 CLS:OPTION BASE 1:DEF SEG=0
40 DIM DCS(6),DTL(6),DSV(7),DAP(4),DAE(4),DOR(6),DCE(6),DPE(3)
50 DIM ECS(6),ETL(6),ESV(7),EAP(4),EAE(4),EOR(6),ECE(6),EPE(3)
60 DIM CCS(6),CTL(6),CSV(7),CAP(4),CAE(4),COR(6),CCE(6),CPE(3)
70 DIM PCS(6),PTL(6),PSV(7),PAP(4),PAE(4),POR(6),PCE(6),PPE(3)
80 DIM UCS(6),UTL(6),USV(7),UAP(4),UAE(4),UOR(6),UCE(6),UPE(3)
90 DIM MCS(6),MTL(6),MSV(7),MAP(4),MAE(4),MOR(6),MCE(6),MPE(3)
100 OPEN "N:SURVEY.86" FOR INPUT AS #1 LEN=254
110 IF EOF(1) THEN GOTO 3300
120 LINE INPUT #1, A$:Z=Z+1
130 LOCATE 12,12:PRINT "        Analyzing responses of survey: ";Z;
140 IF A$="" OR MID$(A$,1,1)=" " GOTO 110
150 MC=MC+1 'Master Counter
160 A=(VAL(MID$(A$,2,1)))+1
170 ON A GOTO 1420, 180, 800
180 DC=DC+1 'Dealer Counter
190 A=VAL(MID$(A$,3,1))+1 'Copy Source
200 ON A GOTO 210,220,230,240,250,251
210 DCS(1)=DCS(1)+1:GOTO 260
220 DCS(2)=DCS(2)+1:GOTO 260
230 DCS(3)=DCS(3)+1:GOTO 260
240 DCS(4)=DCS(4)+1:GOTO 260
250 DCS(5)=DCS(5)+1:GOTO 260
251 DCS(6)=DCS(6)+1:GOTO 260
260 IF MID$(A$,4,2)="  " GOTO 270 ELSE DNR=DNR+VAL(MID$(A$,4,2))
    :DRC=DRC+1 'Number of Readers
270 A=VAL(MID$(A$,6,1))+1 'Technical Level
280 ON A GOTO 290,300,310,320,330,340
290 DTL(1)=DTL(1)+1:GOTO 350
300 DTL(2)=DTL(2)+1:GOTO 350
310 DTL(3)=DTL(3)+1:GOTO 350
320 DTL(4)=DTL(4)+1:GOTO 350
330 DTL(5)=DTL(5)+1:GOTO 350
340 DTL(6)=DTL(6)+1:GOTO 350
```

Figure 2. BASIC Program Segment to Count Responses

```
3310 REM Add Results
3320 CLS:LOCATE 12,12:PRINT "    Calculating results of survey. ";
3330 FOR I=1 TO 6:MCS(I)=DCS(I)+ECS(I)+CCS(I)+PCS(I)+UCS(I):NEXT I
3880 MNR=DNR+ENR+CNR+PNR+UNR
3890 MRC=DRC+ERC+CRC+PRC+URC
3900 FOR I=1 TO 6:MTL(I)=DTL(I)+ETL(I)+CTL(I)+PTL(I)+UTL(I):NEXT I
4450 FOR I=1 TO 7:MSV(I)=DSV(I)+ESV(I)+CSV(I)+PSV(I)+USV(I):NEXT I
4940 FOR I=1 TO 4:MAP(I)=DAP(I)+EAP(I)+CAP(I)+PAP(I)+UAP(I):NEXT I
5490 FOR I=1 TO 4:MAE(I)=DAE(I)+EAE(I)+CAE(I)+PAE(I)+UAE(I):NEXT I
6040 FOR I=1 TO 6:MOR(I)=DOR(I)+EOR(I)+COR(I)+POR(I)+UOR(I):NEXT I
6590 FOR I=1 TO 6:MCE(I)=DCE(I)+ECE(I)+CCE(I)+PCE(I)+UCE(I):NEXT I
7140 FOR I=1 TO 3:MPE(I)=DPE(I)+EPE(I)+CPE(I)+PPE(I)+UPE(I):NEXT I
7690 MPI=DPI+EPI+CPI+PPI+UPI
7700 MPV=DPV+EPV+CPV+PPV+UPV
7710 IF MRC=0 OR MNR=0 THEN MNRAVG=0 ELSE MNRAVG=MNR/MRC
7720 IF DRC=0 OR DNR=0 THEN DNRAVG=0 ELSE DNRAVG=DNR/DRC
7730 IF ERC=0 OR ENR=0 THEN ENRAVG=0 ELSE ENRAVG=ENR/ERC
7740 IF CRC=0 OR CNR=0 THEN CNRAVG=0 ELSE CNRAVG=CNR/CRC
7750 IF PRC=0 OR PNR=0 THEN PNRAVG=0 ELSE PNRAVG=PNR/PRC
7760 IF URC=0 OR UNR=0 THEN UNRAVG=0 ELSE UNRAVG=UNR/URC
```

Figure 3. BASIC Program Segment to Total Responses

Each record has eight fields (D$-K$) to store the quantity of each response to each question. Figure 4 shows that segment of the program for general and combined responses.

After we tested the program, we compiled it to run faster and to allow us to call it from PDS as SVYGRAPH.EXE.

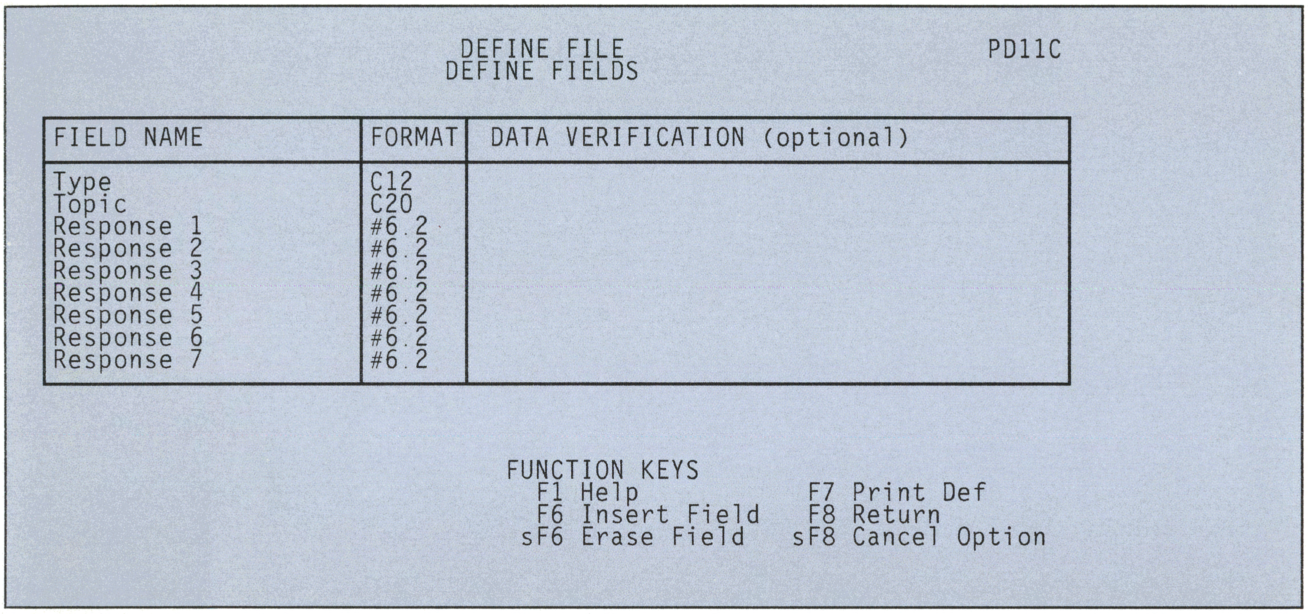
Defining the File for PDS

After we had a program to build a file, we needed to let PDS know about the file. We selected the Define File option from the PDS Files menu.

We named the file SURVEY (as in the program), and defined fields for PDS. We created SURVEY as a direct file rather than an indexed file, so we had no key field to define.

```
7830 REM File Results
7840 OPEN "C:\LIB1\SURVEY" AS #2 LEN=96
7841 FIELD #2, 12 AS B$,20 AS C$,8 AS D$,8 AS E$,8 AS F$,8 AS G$,8 AS H$,
     8 AS I$,8 AS J$,8 AS K$
7842 LSET B$="Cross-survey"
7843 LSET C$="Total responses      "
7844 LSET D$=MKD$(CDBL(DC)):LSET E$=MKD$(CDBL(EC)):LSET F$=MKD$(CDBL(CC))
     :LSET G$=MKD$(CDBL(PC)):LSET H$=MKD$(CDBL(UC)):LSET I$=MKD$(CDBL(MC))
     :LSET J$=MKD$(CDBL(0)):LSET K$=MKD$(CDBL(0))
7845 PUT #2
7846 LSET C$="Average readership   "
7847 LSET D$=MKD$(CDBL(DNRAVG)):LSET E$=MKD$(CDBL(ENRAVG))
     :LSET F$=MKD$(CDBL(CNRAVG)):LSET G$=MKD$(CDBL(PNRAVG))
     :LSET H$=MKD$(CDBL(UNRAVG)):LSET I$=MKD$(CDBL(MNRAVG))
     :LSET J$=MKD$(CDBL(0)):LSET K$=MKD$(CDBL(0))
7848 PUT #2
7849 LSET C$="Price per issue      "
7850 LSET D$=MKD$(CDBL(DAIP)):LSET E$=MKD$(CDBL(EAIP))
     :LSET F$=MKD$(CDBL(CAIP)):LSET G$=MKD$(CDBL(PAIP))
     :LSET H$=MKD$(CDBL(UAIP)):LSET I$=MKD$(CDBL(MAIP))
     :LSET J$=MKD$(CDBL(0)):LSET K$=MKD$(CDBL(0))
7851 PUT #2
```

Figure 4. BASIC Program Segment to Create PDS File



```
                              DEFINE FILE                          PD11C
                             DEFINE FIELDS

  FIELD NAME                | FORMAT | DATA VERIFICATION (optional)
  --------------------------|--------|------------------------------------
  Type                      | C12    |
  Topic                     | C20    |
  Response 1                | #6.2   |
  Response 2                | #6.2   |
  Response 3                | #6.2   |
  Response 4                | #6.2   |
  Response 5                | #6.2   |
  Response 6                | #6.2   |
  Response 7                | #6.2   |

                               FUNCTION KEYS
                                 F1 Help              F7 Print Def
                                 F6 Insert Field      F8 Return
                                sF6 Erase Field      sF8 Cancel Option
```

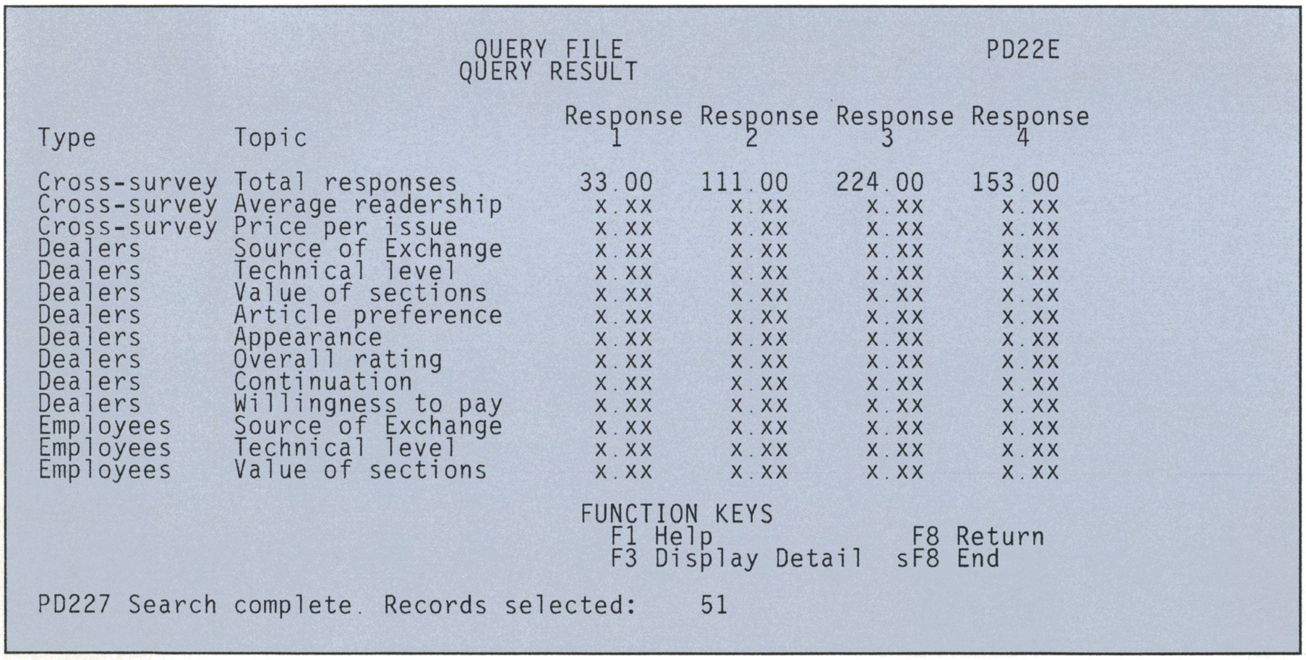
Figure 5. PDS Files Field Definition Screen

To define fields, we gave each field a name and a format. Later, we referred to fields by the names we established here. Figure 5 shows the PDS field definition screen.

We described character fields with the letter C and their length. A pound-sign (#) designates a BASIC double-precision variable. The numbers that follow define the digits allowed before and after the decimal for entry and display purposes.

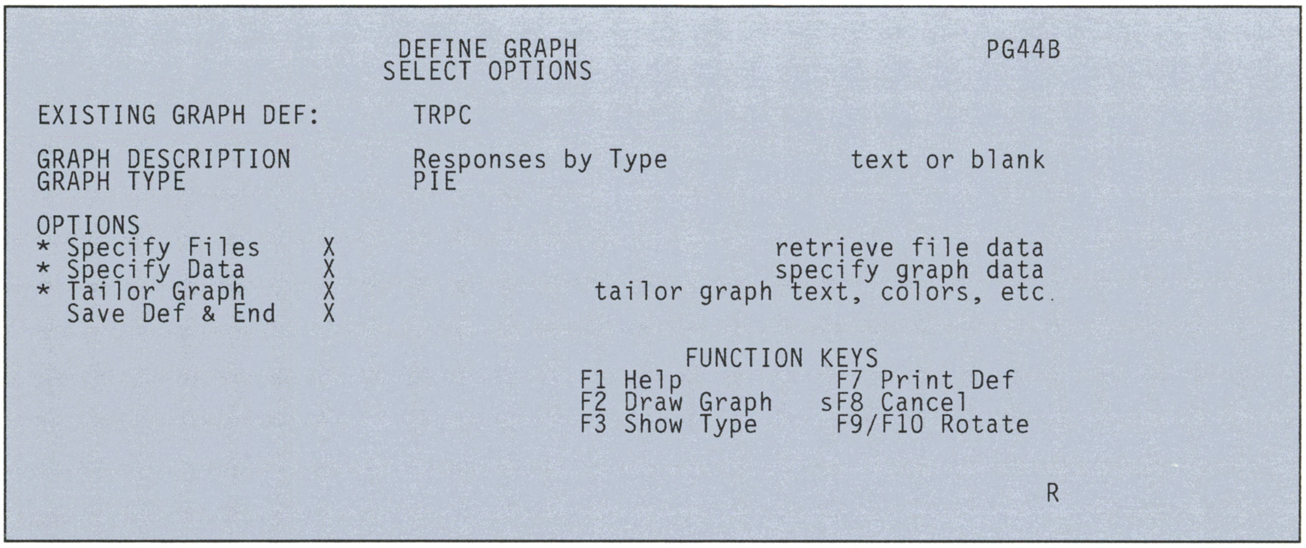
Data verification allows us to define a list of acceptable values, or a range of values. We didn't need this option, because we were automatically creating the file.

What PDS saw when it looked at the file is shown in Figure 6. (Not all records or responses are shown.)



```
                              QUERY FILE                           PD22E
                             QUERY RESULT

                                      Response Response Response Response
Type          Topic                       1        2        3        4

Cross-survey  Total responses           33.00   111.00   224.00   153.00
Cross-survey  Average readership         x.xx     x.xx     x.xx     x.xx
Cross-survey  Price per issue            x.xx     x.xx     x.xx     x.xx
Dealers       Source of Exchange         x.xx     x.xx     x.xx     x.xx
Dealers       Technical level            x.xx     x.xx     x.xx     x.xx
Dealers       Value of sections          x.xx     x.xx     x.xx     x.xx
Dealers       Article preference         x.xx     x.xx     x.xx     x.xx
Dealers       Appearance                 x.xx     x.xx     x.xx     x.xx
Dealers       Overall rating             x.xx     x.xx     x.xx     x.xx
Dealers       Continuation               x.xx     x.xx     x.xx     x.xx
Dealers       Willingness to pay         x.xx     x.xx     x.xx     x.xx
Employees     Source of Exchange         x.xx     x.xx     x.xx     x.xx
Employees     Technical level            x.xx     x.xx     x.xx     x.xx
Employees     Value of sections          x.xx     x.xx     x.xx     x.xx
                                    FUNCTION KEYS
                                      F1 Help               F8 Return
                                      F3 Display Detail    sF8 End

PD227 Search complete. Records selected:    51
```

Figure 6. PDS Files Query Screen



```
                         DEFINE GRAPH                          PG44B
                        SELECT OPTIONS

EXISTING GRAPH DEF:          TRPC

GRAPH DESCRIPTION            Responses by Type            text or blank
GRAPH TYPE                   PIE

OPTIONS
* Specify Files     X                                retrieve file data
* Specify Data      X                                specify graph data
* Tailor Graph      X                    tailor graph text, colors, etc.
  Save Def & End    X

                                           FUNCTION KEYS
                                    F1 Help            F7 Print Def
                                    F2 Draw Graph     sF8 Cancel
                                    F3 Show Type       F9/F10 Rotate

                                                                  R
```

Figure 7. PDS Graphs Option Selection Screen

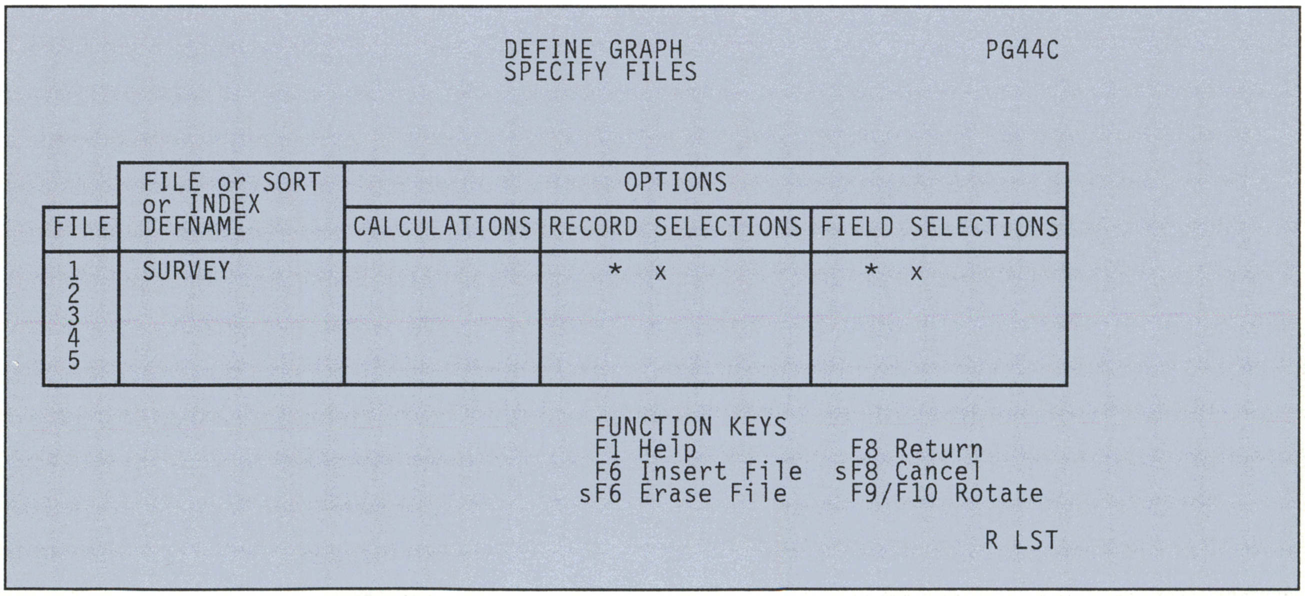
Defining Graphs

With a working file in place, we began describing what kind of graphs we wanted, and where PDS could find the information. PDS Graphs is capable of producing several different kinds of graphs:

- Bar
- Floating bar
- Line
- Line bar
- Marked line
- Overlapped bar
- Pie
- Scattergram
- Stacked bar
- Summed surface
- Surface
- Text

To show the kind of audience *Exchange* has, we chose a pie chart.

Producing graphs with PDS includes defining a graph, then running it. Defining a graph involves defining any files involved, specifying data labels for the chart, and tailoring the results to look the way you want.

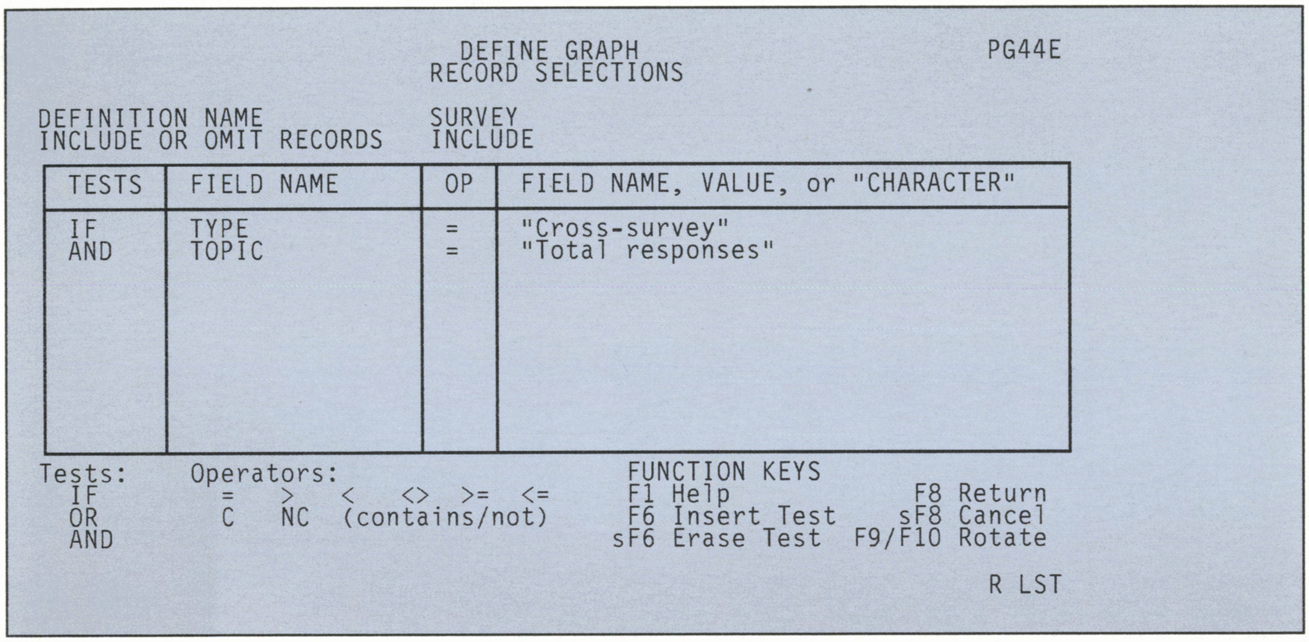


```
                               DEFINE GRAPH                    PG44C
                               SPECIFY FILES
```

FILE	FILE or SORT or INDEX DEFNAME	OPTIONS: CALCULATIONS	OPTIONS: RECORD SELECTIONS	OPTIONS: FIELD SELECTIONS
1	SURVEY		* x	* x
2				
3				
4				
5				

```
                                     FUNCTION KEYS
                                     F1 Help            F8 Return
                                     F6 Insert File    sF8 Cancel
                                    sF6 Erase File      F9/F10 Rotate

                                                                 R LST
```

Figure 8. PDS Graphs File Specification Screen



```
                         DEFINE GRAPH                          PG44E
                       RECORD SELECTIONS

DEFINITION NAME            SURVEY
INCLUDE OR OMIT RECORDS    INCLUDE
 ___________________________________________________________________
| TESTS | FIELD NAME     | OP | FIELD NAME, VALUE, or "CHARACTER"    |
|-------|----------------|----|--------------------------------------|
| IF    | TYPE           | =  | "Cross-survey"                       |
| AND   | TOPIC          | =  | "Total responses"                    |
|       |                |    |                                      |
|_______|________________|____|______________________________________|
Tests:   Operators:                   FUNCTION KEYS
  IF       =   >   <   <>  >=  <=     F1 Help              F8 Return
  OR       C   NC  (contains/not)     F6 Insert Test      sF8 Cancel
  AND                                sF6 Erase Test   F9/F10 Rotate

                                                           R LST
```

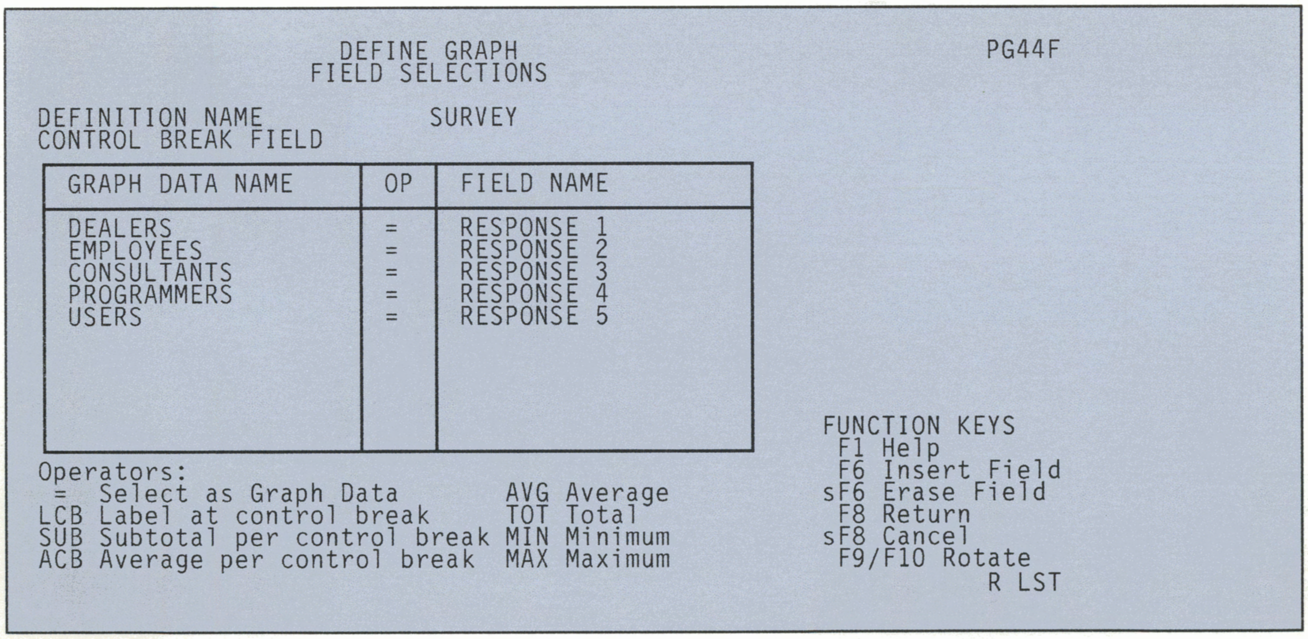
Figure 9. PDS Graphs Record Selection Screen

As Figure 7 shows, PDS lets you select individual options. We chose all the options, since we wanted data from a file, but with our own names for the data. The tailoring option let us see the graph and make cosmetic changes before committing anything to paper.

PDS Graphs lets you select data from up to five files. We had only one file, but were interested in selecting a specific record, and only certain fields within that record.

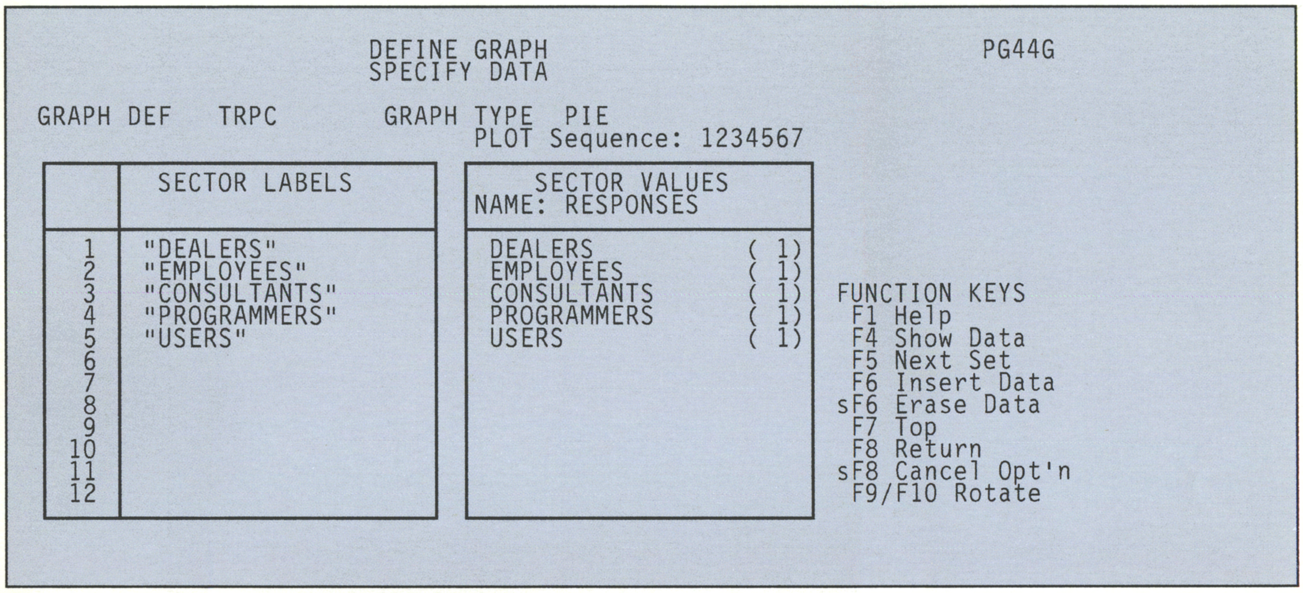
Figure 8 shows the files specification screen of PDS Graphs. We used the record and field selection options, but not calculations.

Selecting an option in PDS leads you to one or more secondary screens. Figure 9 shows the record selection screen.



```
                  DEFINE GRAPH                                 PG44F
                FIELD SELECTIONS

DEFINITION NAME            SURVEY
CONTROL BREAK FIELD
 _____________________________________________
| GRAPH DATA NAME      | OP | FIELD NAME       |
|----------------------|----|------------------|
| DEALERS              | =  | RESPONSE 1       |
| EMPLOYEES            | =  | RESPONSE 2       |
| CONSULTANTS          | =  | RESPONSE 3       |
| PROGRAMMERS          | =  | RESPONSE 4       |
| USERS                | =  | RESPONSE 5       |
|                      |    |                  |     FUNCTION KEYS
|______________________|____|__________________|      F1 Help
Operators:                                            F6 Insert Field
 =  Select as Graph Data        AVG Average          sF6 Erase Field
LCB Label at control break      TOT Total             F8 Return
SUB Subtotal per control break  MIN Minimum          sF8 Cancel
ACB Average per control break   MAX Maximum           F9/F10 Rotate
                                                                R LST
```

Figure 10. PDS Graphs Field Selection Screen

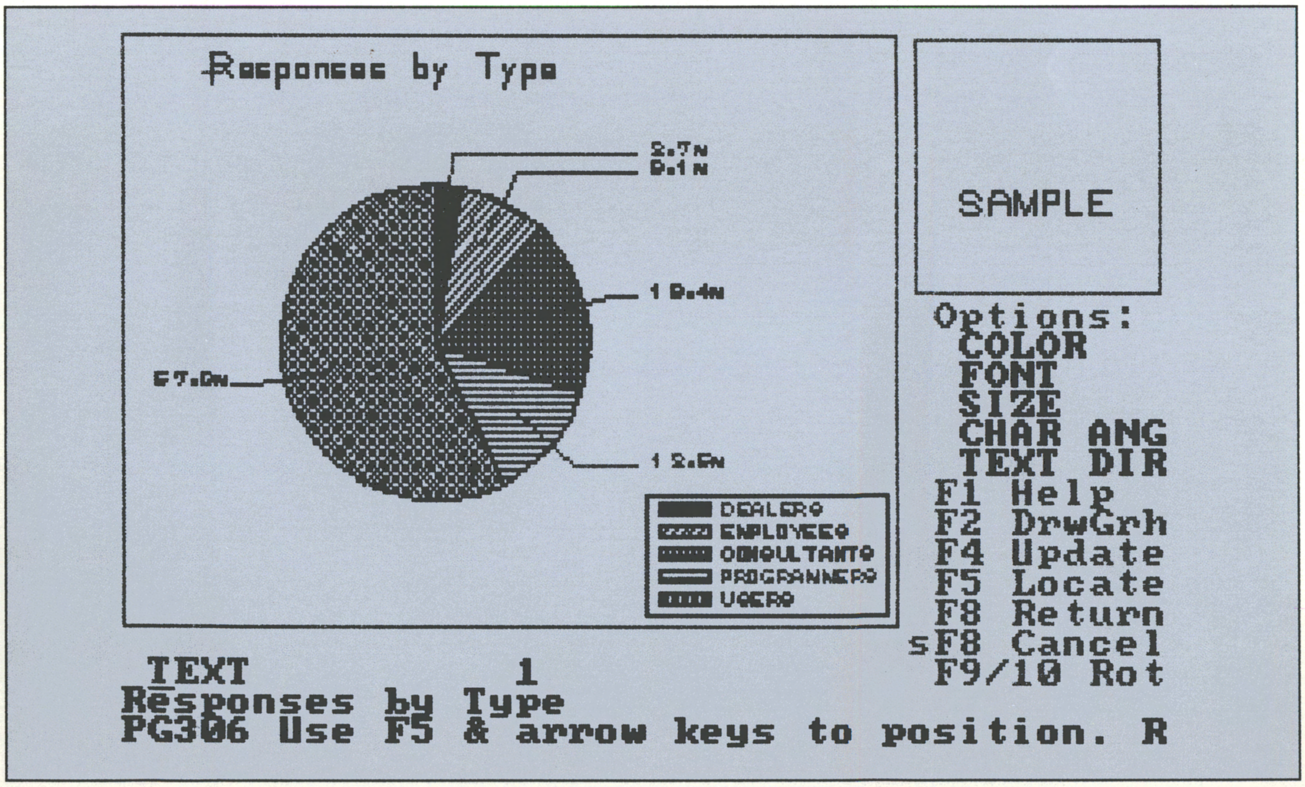


```
                    DEFINE GRAPH                                  PG44G
                    SPECIFY DATA
 GRAPH DEF   TRPC        GRAPH TYPE  PIE
                              PLOT Sequence: 1234567
┌────┬─────────────────────┐ ┌──────────────────────┐
│    │   SECTOR LABELS     │ │    SECTOR VALUES     │
│    │                     │ │NAME: RESPONSES       │
├────┼─────────────────────┤ ├──────────────────────┤
│  1 │ "DEALERS"           │ │ DEALERS          ( 1)│
│  2 │ "EMPLOYEES"         │ │ EMPLOYEES        ( 1)│
│  3 │ "CONSULTANTS"       │ │ CONSULTANTS      ( 1)│ FUNCTION KEYS
│  4 │ "PROGRAMMERS"       │ │ PROGRAMMERS      ( 1)│  F1 Help
│  5 │ "USERS"             │ │ USERS            ( 1)│  F4 Show Data
│  6 │                     │ │                      │  F5 Next Set
│  7 │                     │ │                      │  F6 Insert Data
│  8 │                     │ │                      │ sF6 Erase Data
│  9 │                     │ │                      │  F7 Top
│ 10 │                     │ │                      │  F8 Return
│ 11 │                     │ │                      │ sF8 Cancel Opt'n
│ 12 │                     │ │                      │  F9/F10 Rotate
└────┴─────────────────────┘ └──────────────────────┘
```

Figure 11. PDS Graphs Data Specification Screen

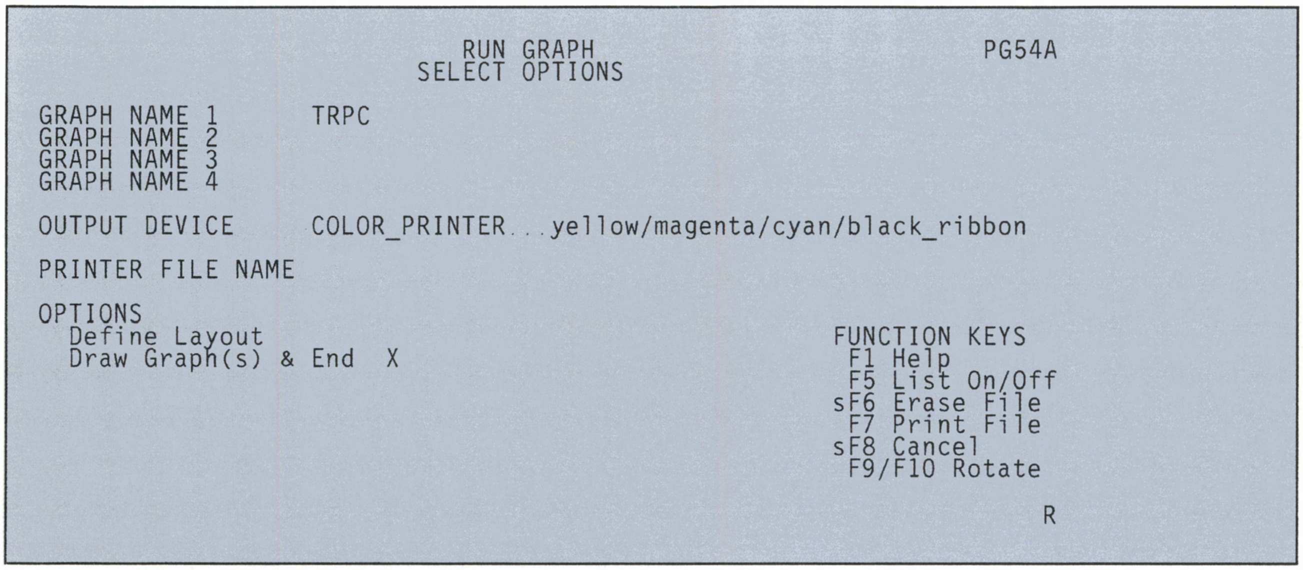
The first three records of our file contained general information (cross-survey). We were specifically interested in the total number of responses of each group. Using the field names we defined in the Define Field task, we selected only the total responses record of our cross-survey audience type.

While we have eight possible numeric responses, we only defined and counted five audience types. We used the field selection screen (Figure 10) to indicate the first five fields to PDS and tell what each field represented. Notice that we could have subtotaled or averaged responses across several records.



```
Options:
  COLOR
  FONT
  SIZE
  CHAR ANG
  TEXT DIR
 F1 Help
 F2 DrwGrh
 F4 Update
 F5 Locate
 F8 Return
sF8 Cancel
 F9/10 Rot
 TEXT          1
Responses by Type
PG306 Use F5 & arrow keys to position. R
```

Figure 12. PDS Graphs Tailoring Screen



```
                              RUN GRAPH                           PG54A
                           SELECT OPTIONS

GRAPH NAME 1       TRPC
GRAPH NAME 2
GRAPH NAME 3
GRAPH NAME 4

OUTPUT DEVICE      COLOR_PRINTER...yellow/magenta/cyan/black_ribbon

PRINTER FILE NAME

OPTIONS
  Define Layout                                        FUNCTION KEYS
  Draw Graph(s) & End  X                                F1 Help
                                                        F5 List On/Off
                                                       sF6 Erase File
                                                        F7 Print File
                                                       sF8 Cancel
                                                        F9/F10 Rotate

                                                                    R
```

Figure 13. PDS Graphs Run Screen

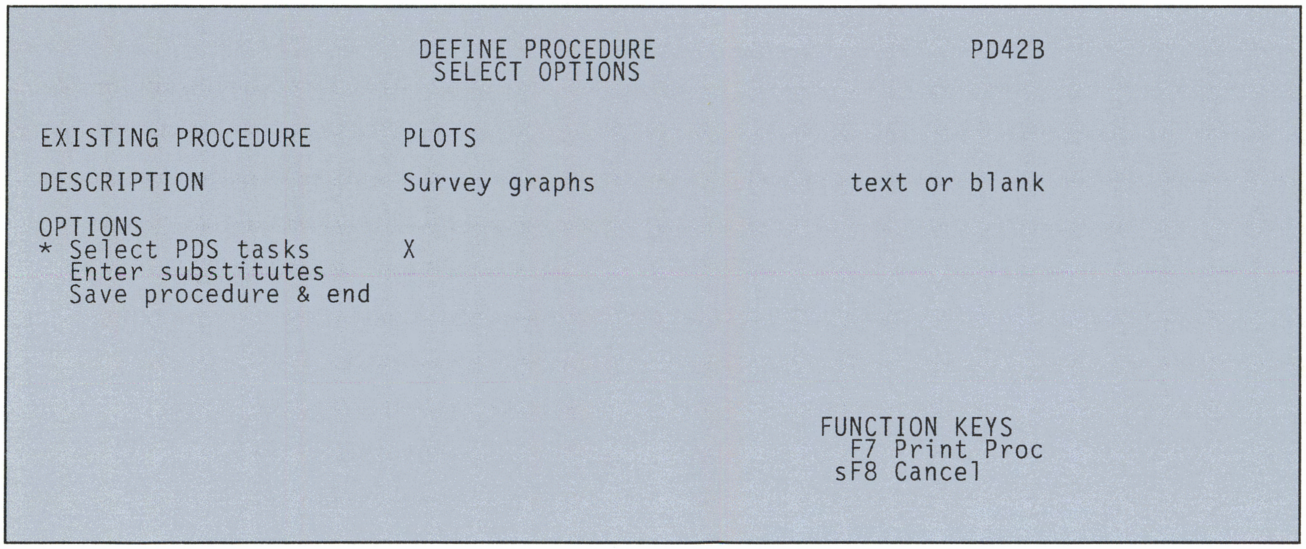
The data option of PDS Graphs lets you enter sector values for the pie chart. PDS then calculate percentages based on those values, and draws the graph. In Figure 11, we referred PDS to the field values from our file. The number in parentheses is the number of the record PDS found that met our specifications. We also gave the names for each sector of the pie on this screen. PDS used these names to build the legend in the lower right-hand corner of the graph.

We can use multiple data sets in our graph, each set represented by a new screen. While this is not useful in a pie chart, it allows multiple line graphs, overlapped bar charts, or stacked bar charts.

With all of the information described for it, PDS presented us with a sample screen showing the graph (Figure 12). Here we added text, change colors, and moved the legend. PDS stored our preferences for actually running the graph on a printer or plotter.

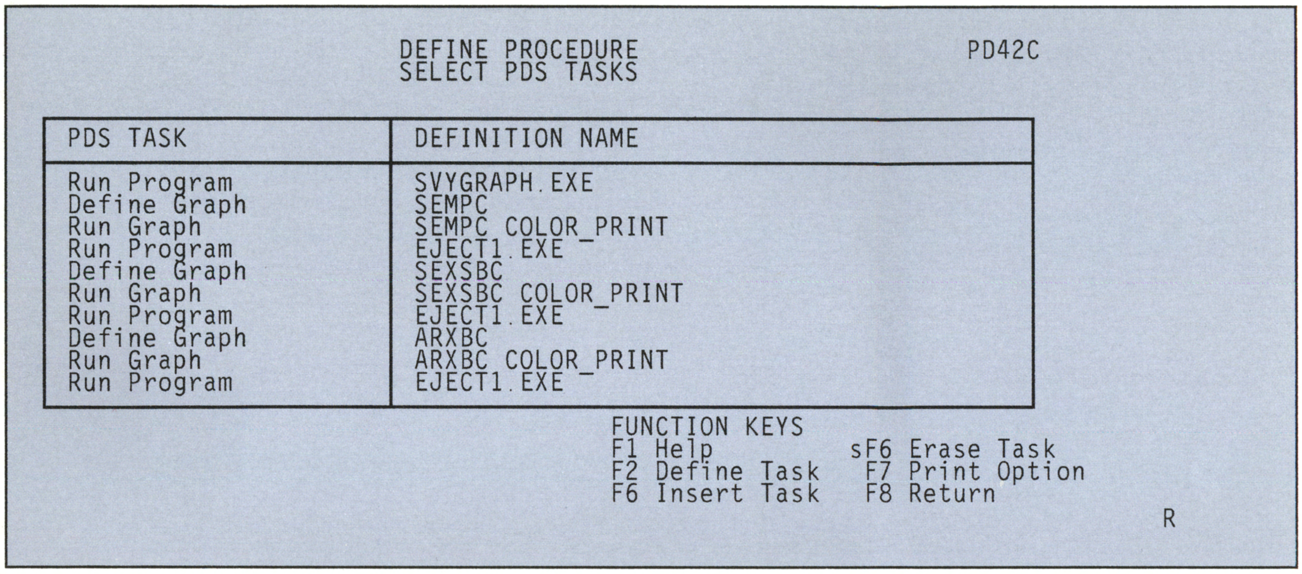
Running a Graph

Another reason we chose PDS Graphs was because it let us combine up to four graphs on a page. We found this useful when comparing the same responses from the different audience types. PDS automatically positions the graphs in the most efficient layout for the output device you've chosen. You may set the layout manually if you want.



```
                           DEFINE PROCEDURE                  PD42B
                            SELECT OPTIONS

EXISTING PROCEDURE     PLOTS

DESCRIPTION            Survey graphs                text or blank

OPTIONS
* Select PDS tasks     X
  Enter substitutes
  Save procedure & end



                                                 FUNCTION KEYS
                                                   F7 Print Proc
                                                  sF8 Cancel
```

Figure 14. PDS Procedure Option Screen



```
                    DEFINE PROCEDURE                          PD42C
                    SELECT PDS TASKS

 PDS TASK               DEFINITION NAME
 Run Program            SVYGRAPH.EXE
 Define Graph           SEMPC
 Run Graph              SEMPC COLOR_PRINT
 Run Program            EJECT1.EXE
 Define Graph           SEXSBC
 Run Graph              SEXSBC COLOR_PRINT
 Run Program            EJECT1.EXE
 Define Graph           ARXBC
 Run Graph              ARXBC COLOR_PRINT
 Run Program            EJECT1.EXE

                                   FUNCTION KEYS
                                   F1 Help          sF6 Erase Task
                                   F2 Define Task    F7 Print Option
                                   F6 Insert Task    F8 Return
                                                                 R
```

Figure 15. PDS Procedure Task Selection Screen

In Figure 13, we selected the color printer for output. You can select a graphics printer, the display, a plotter, or an output file.

The resulting graph (Figure 16) shows that PC users make up the bulk of our audience, followed by consultants, then programmers.

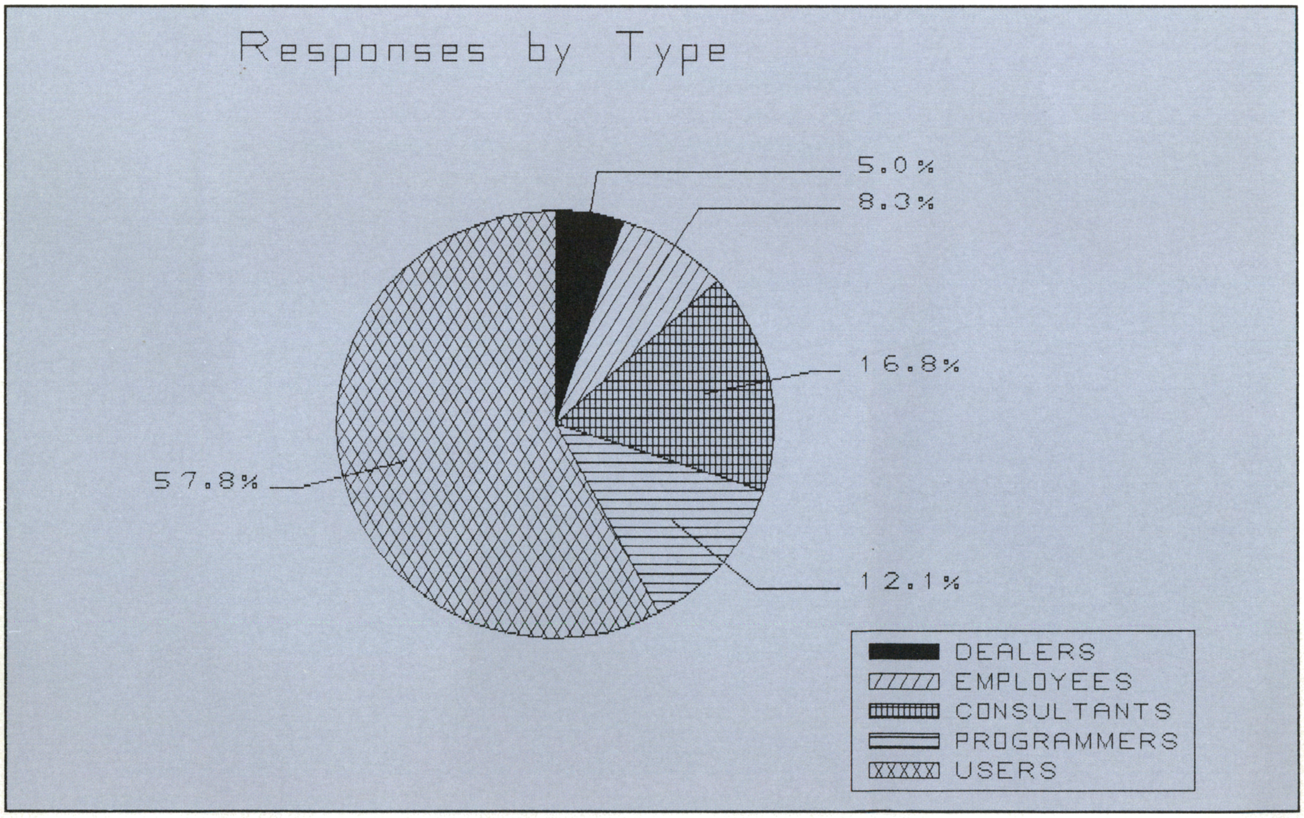


Figure 16. Readership

Automating Plotting

PDS let us define procedures that run tasks automatically. This was perfect for us. We called our procedure PLOTS (Figure 14) and included the tasks we wanted performed.

First, we needed to update the PDS data file based on the latest survey responses. We ran the SVYGRAPH program to update the file. For each graph, we used the Define Graph task to update the graph's data and the Run Graph task to print it. When called from a procedure, these tasks run unattended.

There is one addition to the tasks shown on the screen (Figure 15): PDS Graphs thinks of output devices as sheet-fed (plotters), rather than continuous-form (printers). This can mean a graph printing over a perforation.

To solve this, we compiled a one-line BASIC program called EJECT1 that had the following statement:

```
10 LPRINT CHR$(12);
```

This sent a page eject command to the printer.

Once defined, running a procedure is as easy as calling it by name. Once you define a program, graph, file, or procedure, PDS retains the name. Whenever PDS asks you to supply one of these names, you can use F9 and F10 to rotate through a list of existing choices.

The biggest benefit to using PDS is that you can start it and walk away. This gave us time to review your opinions and keep *Exchange* responsive to your needs.

PCjr Program Compatibility

Steve Mark
Atlanta IBM Employees PC Club

It has often been my experience that programs that are not supposed to function properly on the PC*jr* actually run just fine. The purpose of this article is to share, with other PC*jr* owners, the factors I have found that do and do not affect PC*jr* compatibility.

I will try to keep this discussion very non-technical. I don't know much about interrupt levels, assembly language, or soldering irons, and have no burning desire to learn now. I promise to get no deeper than an occasional reference to the CONFIG.SYS file and some Internal Modem command codes.

I also promise that everything in this article is based on my *personal* experience unless otherwise noted. There will be no "I think I heard somewhere that . . . "

My present system has 640K with two diskette drives. I started in September 1985 with a 128K, single drive PC*jr*. A month later, I bought a PC*jr* Internal Modem. Then came a 128K Microsoft Junior Booster, a Quadram Expansion Chassis that included a second drive, 384K, and a parallel port. A Hayes 1200 bps external modem and an IBM Proprinter finished the configuration (and my bank account).

Most reasons that programs supposedly will not run on the PC*jr* can be categorized as follows:

1. Configuration
2. Communications issues
   - The PC*jr* Internal Modem
   - Comm port addressing
   - DMA
3. DMA
4. The Video Buffer
5. It actually won't work on a PC*jr*.

This article discusses each of the above considerations and has a section that contains some simple techniques I have used to get the best performance from my PC*jr*.

Configuration

One of the earliest sources of misinformation regarding PC*jr* compatibility was the original maximum configuration. Many programs were said not to run on the PC*jr* simply because the programs required two diskette drives and/or more than 128K. To confuse matters more, many dealers and software vendors seemed unaware that several accessory manufacturers made it possible to add a second diskette drive and/or a fixed disk to the PC*jr*.

Some software manufacturers, however, came out with PC*jr* versions of their packages. These versions usually included one or two cartridges and were designed to run on a 128K, single-drive machine. The use of the ROM cartridges resulted in reasonably good performance, but prevented the PC*jr* version of these programs from running on anything other than a PC*jr*. The trade-off was that the user gained performance, but lost upward compatibility.

Because there were special PC*jr* versions of these programs, and because these versions would not run on other PCs, people believed that the PC version would not run on the PC*jr*. Wrong! In most cases, the "regular" version will run just fine on PC*jr*s that meet the package's configuration requirements.

Vendors created versions of their software (e.g., IBM's Planning Assistant) that utilized an overlay structure to fit into 128K, although with substantial performance degradation. Many PC*jr* users who have upgraded their machines to 256K and more are still suffering unnecessarily from the poor performance of the PC*jr* version of their software because they do not realize that the original constraint was memory, not the "*jr*" on their machines' nameplates.

The PC*jr* has one unique memory consideration. The Color/Graphics Adapter on a PC contains a 16K video buffer. The PC*jr*'s display adapter does not include memory for this buffer. Therefore, when you boot the PC*jr*, DOS allocates 16K of the system's memory for a video buffer. (The positioning of this buffer is the reason you need drivers such as PCJRMEM.COM in order to recognize memory beyond 128K, but that is beyond the scope of this article.) The result is that a program that has less than 16K to spare when running on a PC will not run on the same size PC*jr*. Remember that the words "requires a 256K system" usually mean "will not fit in a 128K system."

Communications

The PC*jr* Internal Modem

Where do I start? This little jewel has caused more confusion and consternation than all the tax simplification measures that the U.S. Congress could ever dream of!

Most of the problems in getting communications programs to run on the PC*jr* are really problems getting the programs to send the proper commands to the PC*jr* Internal Modem. The modem does *not* accept the Hayes (AT) command set. Most popular communication programs (QMODEM, PC-TALK, CROSSTALK, etc.)

are set up to issue the Hayes command set, with some method of specifying a different command set if necessary. In browsing through bulletin board systems around the country, it appears to me that many people have problems using non-Hayes compatible modems regardless of whether the attached PC is a PC*jr*. The PC*jr*'s internal modem just seems to get more criticism because it is concentrated in one environment.

Some programs (including the IBM PC Videotex Connection and the terminal program on the PC*jr* Sampler diskette) support the Internal Modem's command set and will dial, communicate, and hang up correctly. However, they often will not properly do things such as changing data format from 8-N-1 to 7-E-1. Other programs such as QMODEM allow the user to specify the control sequence for dialing, etc., but also will not issue the commands to change format. The result is that the modem will not dial, or else the received data looks like garbage.

It is usually very simple (at least it is with QMODEM and PC-TALK) to manually issue modem commands from the "terminal" screen of your program. Therefore, the best advice I can give you is to keep the PC*jr*'s command reference handy. There is one in the small supplement to the *PCjr Guide to Operations* that comes with the modem. There is a better one in the *PCjr Technical Reference*. It's not really as onerous as it sounds, though. The only time you should need to enter commands manually is when changing format or entering transparency mode.

Now comes the fun part. If all you want to do is exchange messages and download files, you can skip this discussion. However, if you want to endear yourself to your favorite SYSOP and be a good BBS citizen, you will occasionally want to upload a file or two. The PC*jr*'s internal modem makes that a real adventure. To understand why, you need to realize that the modem attempts to execute commands even if it finds them in the middle of a data stream that you are sending to another computer. If you try to upload a binary file (like a .COM or .EXE file), sooner or later the data will include a bit pattern that the modem thinks is its command character (Ctrl-N). It will not only fail to send that character down the line, but will try to execute the "command" it thinks comes behind it.

Without going any deeper into what happens when the modem receives what it thinks is an invalid command, or delving into how to put the modem into transparency mode, it should be obvious that the problem we are discussing is a function of the modem being used and not the fact that the system is a PC*jr*. Try all the tricks you can find documented on many bulletin board systems until you find one that works.

Communications Port Addressing

This is one problem you *won't* have if you use the internal modem. Some people, cannot get their communications programs to recognize external modems. If you have this problem, then you may be facing the COM1 versus COM2 addressing mystery.

When the PC*jr* Internal Modem is installed, it is COM1, and the external serial port on the back of the system is COM2. When there is no modem in the internal slot, the external port becomes COM1. So far, so good. The problem is that, regardless of whether an internal modem is installed, the base address and interrupt level of the external port remain the ones normally used for COM2.

If your communications program is well-behaved and does not try to bypass the system BIOS, none of this should cause any problem. Just tell the program that your external modem is COM1. QMODEM*jr* (version 1.07) works fine this way. Other programs seem to use the base address and interrupt level associated with whatever comm port you have specified for your modem. To use these programs (e.g., QMODEM 2.0 and 2.2), just tell them your modem is at COM2. That is how these three programs worked on my system after I removed the internal modem.

But I have heard of different people who got different results, supposedly using the same programs! In fact, on several bulletin board systems there are routines available that claim to swap COM1 and COM2 so you can use an external modem. The SYSOP of the IBM*jr* forum on CompuServe tells me the problem appears to be a function of the compiler used to compile the program and/or any external port drivers that the compiler uses.

If you think you have this problem, you have two choices. The first is to find an internal modem and plug it into your machine. This will straighten out the addresses so you can use your external modem as COM2. The second is to get one of the programs that logically swaps the addresses back to where they belong. These programs have names like SWAPCOM or COMSWAP, and can be found on many PC*jr*-oriented bulletin board systems.

There has been an interesting development on this issue. The QINSTALL program for QMODEM 2.2 lets the user specify the base address and interrupt level for each comm port. The defaults are provided by the program, so all you have to do is switch the specifications for COM1 and COM2. Because I was unable to recreate the problem on my system, I could not test the effectiveness of this new feature. In theory, however, it should solve the problem.

Direct Memory Access

Direct Memory Access (DMA) is a standard feature on all IBM PCs except the PC*jr*. Simply stated, DMA allows the processor to overlap disk I/O operations with other work. Some manufacturers offer a means of adding DMA to the PC*jr*.

With regard to communications, DMA allows the system, when downloading a file, to continue receiving new data while the data just received is being written to diskette. Without DMA, a few incoming characters will be dropped on the floor each time the system writes out a buffer full of data. When your download completes, you will have, at best, a text file with some missing letters, and, at worst, an unexecutable program. Uploading is not affected, because the system will not try to send data while it is reading from the diskette.

If you have very little extra memory beyond that required to run your communication program, then the lack of DMA is indeed a problem. But if you have extra headroom, the solution is very simple: do your downloading to a RAM disk.

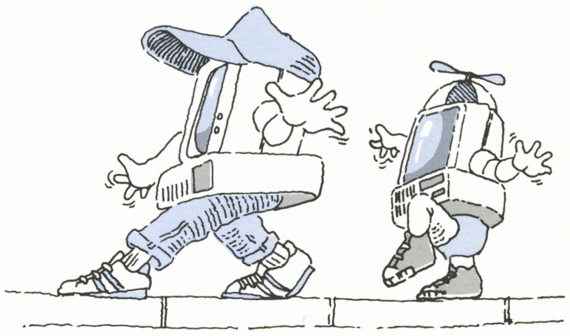
The documentation for QMODEM 2.2 says, "A PC*jr* must contain a DMA chip to successfully use the transfer protocols." I say this is not true. I normally run QMODEM with a 200K RAM disk. If I plan to download more than 200K of files, I simply install a larger RAM disk. I have successfully up/downloaded files to bulletin board systems, CompuServe, and point-to-point. After the transmission is completed, copy your new files to a real diskette immediately to prevent losing them in a power outage.

I know that several PC*jr* users have experienced system hangs that require a reboot to clear. This would make one quite nervous about trusting a RAM disk to receive a lengthy file. Let me try to set you at ease. There is a diskette containing patches to DOS 2.10. These patches fix a bug in the way DOS handles (or fails to handle) certain keyboard interrupts that are unique to the PC*jr*. I installed the patches (a very simple process) over six months ago and have not experienced any lock-ups since that time. You should be able to get the patch diskette from your dealer or another PC*jr* user. Ask around, it's worth the search.

DMA in General

The lack of DMA affects the performance of programs that use diskette drive(s). However, except for the communications concerns covered above, this shortcoming rarely affects whether a program will indeed run on a PC*jr*.

Some programs will simply not run correctly, or at all, without DMA. I also understand that some copy protection schemes use the DMA processor. If true, these programs obviously will not run on a standard PC*jr*. (I do not have a DMA chip in my PC*jr* and so have had no first-hand experience with these programs.)



If the software you want to run requires DMA, all it takes to run it on your PC*jr* is money. At least one manufacturer offers a DMA chip for the PC*jr*. The hitch is that the chips usually come bundled in expansion units which also include second drives, parallel ports, and other goodies. That's fine if you are just starting to expand your PC*jr*, but if you don't want (or already have) a second drive, it's an expensive DMA chip.

The Video Buffer

Here again, I am at (and sometimes beyond) the limits of my personal knowledge, but I'll give it a try anyway.

The video buffer is where the hardware gets the information it needs to put characters and pictures on the screen. Your program has some options as to how (or if) it will put information into the buffer. If the program uses DOS or normal BIOS to write to the buffer, then there should be no problem. Fortunately, most programs are well-behaved in this regard. If your program tries to address the buffer directly, or (worse) tries to write directly to the display, bypassing the buffer, then you've got a problem.

There are two differences between the way the video buffer is handled on the PC and on the PC*jr*. First, as I mentioned earlier, the video buffer is in main memory on the PC*jr*, rather than on the display adapter as is the case on other PCs. This means it is at a different physical address. If your program tries to write directly to the address where it thinks the buffer is, it won't work. This is especially true if your PC*jr* has been expanded beyond 128K, because the memory management software (PCJRMEM.COM or equivalent) moves the video buffer so DOS can find the expansion memory. I have not run across many programs that fall into this category, but I suppose that is small comfort to those of you whose favorite game does. I do not know of any method of getting such programs to run on the PC*jr*.

The second problem is another that just takes money to solve. In addition to being in a different place, the PC*jr*'s video buffer is mapped differently than the PC's. This means that a position in the PC*jr*'s buffer corresponds to a different place on the screen than the same position in the buffer on the PC's Color/Graphics Adapter. A program that tries to manipulate data directly within the buffer will run, but the screen will look very strange. I ran across a game once whose title screen was garbage. I took a guess, pressed "any key," and the rest of the game ran fine.

As I said, all it takes to solve this problem is money. The mysterious little PC/PC*jr* switch on the back of second drive units is there for precisely this purpose. When you power up your machine with the switch in the PC position, and execute the PCVIDEO routine that comes with the hardware, the mapping of the video buffer is changed to match the PC's. After that, my game had a real title screen. If you really want to, you can even run a PC*jr* using DOS 1.10 this way.

Some Programs Will Not Work on a PCjr

The vast majority of software written for the IBM PC will run on an adequately configured PC*jr*. In addition, we have discussed several ways to make programs that supposedly will not run on the PC*jr* work as well.

The simplest way to speed up your PCjr is to add memory.

There are programs, though, that cannot be made to execute correctly on our PC*jr* machines. Most of these programs write directly to the hardware interface for the diskette drive or display. They do this to optimize performance, or to bypass some limitation of the interface supplied by the system BIOS. Some games and graphics programs write directly to the display to do fancy manipulation of the screen images and improve performance. Unless there is a PC*jr* version of these programs, you're probably out of luck.

A few programs rely on timing to run correctly. And finally, there are those programs that require some piece of hardware (like a math coprocessor) that cannot be attached to the PC*jr*.

All told, these problems represent a very tiny portion of the mountains of software available for the PC*jr*.

Performance Tips

The simplest, most effective thing you can do to speed up your PC*jr* is to add memory. There are three ways you can use additional memory to achieve a significant (up to 50 percent in some (cases) increase in performance.

The first was discussed earlier. Some programs such as Writing Assistant and Planning Assistant come in two versions. One of these versions is structured to run in a 128K machine. It generally makes heavy use of overlays and goes out to diskette each time you invoke a new function. It works in 128K, but I hope you like the sound of your diskette drive grinding away. The other version, a resident version, loads completely into memory and only needs to access the diskette for data files. An added advantage is that after the program is loaded, you can usually remove the program diskette and insert a separate data diskette. This gives you much more room for files on a single-drive system.

The second performance improvement you will gain from adding memory takes us back to our old friend, the video buffer. In order to keep the current image on the screen, the image needs to be refreshed every few microseconds. If you remember, the PC*jr*'s video buffer is in main memory. Because of this, the cycles it takes to do this video refresh are taken from other jobs that are executing in main memory. (I'm not sure, but I believe one out of every three cycles is used for this purpose.) That is why the PC*jr* seems to process slower than the PC even though it has the same clock and 8088 processor. The good news is that this impacts only the first 128K of memory. Programs that are loaded into expansion memory are not affected.

There are two ways to force all of your programs to load into expansion memory. The simplest is to use the "/C" parameter with PCJRMEM.COM in your CONFIG.SYS file. This causes DOS to fill the system's main memory with I/O buffers, and force all user programs to be loaded above 128K. It also allows the use of the PC*jr*'s enhanced video modes, but that's another story.

The other way to fill up the first 128K is to define a RAM disk of at least 90K. This combined with DOS and the video buffer will fill the main memory, and your programs will be loaded into the expansion memory.

Does it really help? I have a program called THATSALL.EXE that plays the well-known cartoon theme and writes "That's all Folks" across the screen. It takes about 26 seconds to run in main memory, but in expansion memory, it takes about 17 seconds, the same as on a PC or XT. That's a 35 percent improvement.

The third way to take advantage of additional memory takes us back to my old favorite, the RAM disk. Let's go back to those programs that had to overlay themselves to fit into 128K. Sometimes the full version is a separate product that will cost you either an upgrade fee or the full price for the product. If you don't need the additional features that may be available in the full product, you can still get near-resident performance from the PC*jr* version.

Just load the program and its required modules onto your RAM disk, and execute it from there. It will still go through its overlaying process, but will do so at the speed of memory rather than the diskette drive. With a little experimenting, it's not hard to determine which files must be copied to the RAM disk to make this trick work. (Note: this technique does not apply to overlaid programs.) I use this technique to make the Personal Computer Picture Graphics program run almost as fast on my PC*jr* at home as it does on the XT with fixed disk at work.

The PCjr is a much more useful machine than it is given credit for.

Some programs, including the Assistant Series, allow you to specify a work drive for the program to use when sorting and doing various other tasks. By using your RAM disk for this purpose, many jobs will go a lot faster.

Are you tired of having to swap back to your DOS diskette every time you want to use an external command such as DISKCOPY, CHKDSK, PRINT, FORMAT, etc.? Why not just copy those programs that you use often onto your RAM disk when you boot the system? Then they will be right there on drive C for you to use whenever you need them. It's almost like having a small fixed disk.

The one caution to remember about a RAM disk is that when you reboot, or if a power fluctuation causes the system to do a power on/reset, the contents of your RAM disk are lost. Therefore, you should not put any non-recoverable data on a RAM disk. I'm a little gutsy, so when I use Filing Assistant, I copy my file to the RAM disk, update it there, and then copy it back. *(Editor's note: Don't be gutsy in South Florida, where lightning often causes power outages.)* I figure that, at worst, I may have to reenter the session's updates if something happens before I can copy the file back to diskette. In the eleven months I have been operating this way, I have never had a problem. Not only does it seem as if I were using a fixed disk, but I've also saved a lot of wear and tear on my diskette drives. If you are using Reporting Assistant, the effect is truly amazing.

Summary

I hope this has been of some help to you. I have tried not to go too far away from the intended subject. For example, there has been no discussion of the PC*jr*'s advantages over the PC (video, music, and a smaller footprint), or the fact that the PC*jr* comes with things that you have to add to a PC (like a display adapter, serial port, game port, etc.). My objective has been simply to show that the PC*jr* is a much more useful machine than it is generally given credit for.

BASIC Floating Point Numbers

Art Seddon
Northeast Indiana PC Users Group

Introduction

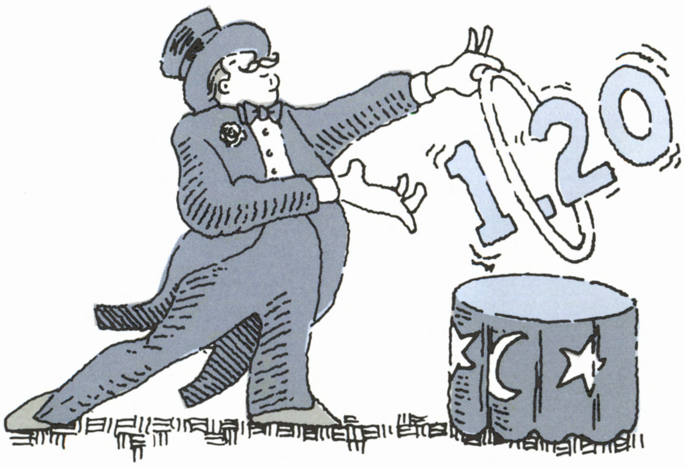
If you've ever wanted to write a number-crunching Assembler subroutine for use with an IBM BASIC program, you will have difficulty finding a reference (or someone) to tell you how IBM BASIC formats single-and double-precision numbers. Most references for Assembler use integers for examples of numbers passing from BASIC to Assembler. Other references offer schematics of the formats, but these lack details for coding Assembler math routines. So, the following is my own explanation deduced from trying many examples. My reasoning may be flawed, but I have been able to write Assembler programs that work!

Format of Floating-Point Numbers for IBM BASIC

Overview

Below is an explanation of the two formats used for floating point numbers in IBM BASIC.

1. *Exponent* (base two). One byte.
   a. If the exponent byte is zero, the floating-point number is zero.
   b. If the most significant bit is a one, the exponent is positive; if a zero, the exponent is negative.
   c. Subtract 81 hex from exponent byte to obtain its value.
2. *Mantissa.* Three bytes for single precision; 7 bytes for double.
   a. The leading bit is the sign bit. A 0 is a positive number, a 1 is a negative number.
   b. The mantissa is normalized so the leading bit is a 1. Then this bit is discarded and replaced with the sign bit; to decode, reverse the process.
   c. The binary point follows the implied first bit.



Discussion of the Floating-Point Number Structure

In IBM BASIC, a single-precision number is stored in memory as 8 bytes; a double-precision number is stored as 12 bytes. These bytes are arranged in RAM as shown below:

Single-precision:
04 53 58 00 19 04 1E 81 (hex)
= 1.2345 (decimal)

Double-precision:
08 5A 00 00 00 00 00 00 5B 72 01 74 (hex)
= 1.234*E-4 (decimal)

Following are descriptions of the bytes:

1. The first byte is the number of bytes in the actual number.
2. The next two bytes are ASCII characters (2 max); the name of the floating point number.
   - In the single-precision example, 53 58 (hex) translates to SX.
   - In the double-precision example, 5A (hex) translates to Z.
3. A byte inserted so that 4 bytes (2 words) precede the actual number.
4. The next 4 (or 8) bytes give the actual FP number.

BASIC passes the address of the first byte in the actual number to the assembler when it calls the subroutine.

For the single-precision example, the last four bytes describe the number. In RAM, these four bytes are arranged:

```
19 04 1E 81
```

Rearrange these RAM bytes into the following logic sequence:

```
81 1E 04 19
```

Now we translate these into the actual number (hex, binary, or decimal, as the reader desires). Recall that Assembler and DEBUG translate RAM into hexadecimal. The first byte is the exponent to the base two. The next three bytes are the mantissa which, when multiplied by the exponent factor, gives the appropriate number.

Let's consider the exponent and mantissa separately.

Exponent

There are two complicating factors which influence the translation to the exponent. First, byte 00 is reserved to identify the number zero. Thus, it is only necessary to examine one byte to recognize a zero number.

Second, in order to handle both positive and negative exponents, the actual exponent is offset by 128 decimal; therefore, the actual exponent can vary from +/- 127 for base two; or +/- 38 for base ten. One will see immediately that the 128 decimal offset is an 8 (hex) in the first character of the byte or a 1 in the most significant binary bit. By checking this bit, the sign of the exponent is obvious.

In order to translate the exponent byte, it then becomes necessary to subtract a hex 80 or discard the leading bit after treating a 1 as a positive exponent and a 0 as a negative exponent.

Since the least significant bit has a 0 reserved to identify the number zero, a two represents an exponent of one, etc. Therefore, the exponent byte should have a hex 81 subtracted from it (1000 0001 binary or 129 decimal) to obtain the actual exponent and its appropriate sign. For this example, the exponent factor is 2^0, or one.

Mantissa

The mantissa consists of 3 bytes for a single-precision number (7 bytes for double-precision). The most significant bit is the sign bit. If this bit is a zero, the mantissa is positive. If it is a one, the mantissa is negative. Furthermore, the mantissa is "normalized" so that a 1 appears in the leading bit. Then it is discarded and replaced by the sign bit. In this manner, maximum use of available bits is achieved. The binary point is assumed to follow the implied leading bit.

Thus the mantissa has a units bit and 23 fractional bits. A double-precision number has 55 fractional bits. Remember the number zero sets all bytes in the mantissa as well as the exponent byte to zero.

In this example, the 3 mantissa bytes are 1E 04 19 (hex) which translates into binary as:

```
0001 1110 0000 0100 0001 1001
```

Replacing the sign bit (a 0 means a positive number) with the implied 1 (normalized) and adding the binary point gives a mantissa of:

```
1.001 1110 0000 0100 0001 1001
```

which translates into a decimal mantissa of +1.2345. Since the exponent factor was 1, the resulting number is also +1.2345.

The translation of the double-precision example is left for you to try.

New Products

The following is a summary of new product announcements made by IBM. Detailed information about each product is available through our Electronic Bulletin Board System (EBBS). Details for using this system are on the inside front cover.

Hardware

- **IBM 6184 Color Plotter** — an economical large-format multi-pen drafting plotter.
- **IBM 3812 Pageprinter TEMPEST Feature** — filters and cables that allow the IBM 3812 Pageprinter to meet TEMPEST standards for prevention of electronic eavesdropping.
- **IBM 3118 Scanner Model 020** — a scanner with a 30-sheet automatic document feed.
- **IBM 1MB Memory Module Kit** — doubles the capacity of the IBM (PC/AT) Memory Expansion Adapter to 6MB per adapter.

Software

- **IBM Virtual Machine/Personal Computer Version 2.01** — provides new attachment alternatives, device support, and performance improvements.

Program for Base Conversion

Clarke M. Gilbert
Westchester (New York) PC Users Group

The following is a simple BASICA program that will convert numbers in various bases to decimal numbers and vice-versa. This program works with positive integers from 1 to 32767, which is 15 binary bits. It is self-explanatory. *(Editor's note: In the November/December 1986 issue of* Exchange, *the article "Hexadecimal and Other Numeric Curses" describes converting between number bases. This program is an appropriate supplement to that article.)*

```
001   'save "b:baseconv.bas",a
005   ON ERROR GOTO 30
010   OPEN "b:numbases.txt"FOR APPEND AS #1
020   DIM D%(20)
030   PRINT "THIS PROGRAM WILL HANDLE BASES 2 - 9 AND 16 ONLY"
040   PRINT "NUMBERS MUST BE POSITIVE AND LESS THAN 32768"
045   PRINT "OR YOU WILL GET OVERFLOW ERROR"
050   PRINT "32767 = 77777 OCTAL OR 7FFF HEX "
060   PRINT "TO CHANGE BASE INPUT 0 "
070   INPUT "enter base # ";B%
080   IF B% = 16 THEN 110
090   IF B%<2 OR B%>9 THEN 30
100   Y%=0
110   PRINT "TO CONVERT FROM BASE TO DECIMAL"
115   INPUT "ENTER 1, ELSE RETURN";Y%
120   IF Y% = 1 THEN 300
130   INPUT "ENTER DECIMAL # <32768 ";A%
140   IF A%=0 THEN 70
150   I%=0
160   N%=A%
170   C%=N%\B%
180   I%=I%+1
190   D%(I%)=N%-C%*B%
200   N%=C%
210   IF N%>0 THEN GOTO 170
220   S$=""
230   FOR J% = I% TO 1 STEP -1
240   STR1$ = RIGHT$(STR$(D%(J%)),1)
250   IF D%(J%)>9 THEN STR1$=CHR$(D%(J%)+55)
260   S$ = S$+STR1$
270   NEXT J%
280   PRINT "DECIMAL ";A%;" = ";S$;" IN BASE ";B%
290   GOTO 130
300   INPUT "ENTER NUMBER IN SELECTED BASE ";A$
310   IF A$ = "0" THEN 70
320   L%=LEN(A$)
330   N%=0
340   FOR J% = L% TO 1 STEP -1
350   K%=L%-J%+1
360   M$=MID$(A$,K%,1):M%=ASC(M$)-48
370   IF M%>48 THEN M%=M%-32
380   IF M%>16 THEN M%=M%-7
390   N% = N% + M%*B%¬(J%-1)
400   NEXT J%
410   PRINT A$;" IN BASE ";B%;" = ";N%;" IN DECIMAL"
420   GOTO 300
430   END
```

Support from Your Authorized IBM PC Dealer

David Both
IBM Corporation

Has this happened to you? You've been struggling with the report your boss requested this morning, but things have not been going well. You have managed to overcome several problems, but now you're stuck, and the report is due this afternoon. The documentation, which helped you solve earlier problems, is totally silent about your latest problem—neither the index nor the table of contents gives the slightest clue about where to look for information.

Where do you turn for help? You could go to your co-workers or friends; you could approach members of your local PC user group. You also could go to your Authorized IBM Personal Computer Dealer.

Your Authorized IBM PC Dealer can give you accurate, timely information. Your dealer has had experience with many kinds of problems you may encounter as a computer user. Your dealer can call upon the experience of in-store sales and service personnel, and possibly outside consultants as well. In turn, these people often dig into the documentation and come up with answers.

However, if the dealer, staff or consultants do not have direct experience with a particular problem, most dealers have access to additional resources that can help provide answers.

When You Call Your Dealer

Before you contact your Authorized IBM Personal Computer dealer, you should gather certain information about your computer system and the software you are using. Your dealer will want to know which version of DOS you are using and the contents of your CONFIG.SYS and AUTOEXEC.BAT files. This information will give him a complete picture of your operating environment.

You'll also need to provide details about what you were doing at the time the error occurred, for example: which program you were using, which operation you were performing when the problem occurred, and which (if any) programs were running as RAM-resident utilities.

Armed with this information, you contact your authorized dealer and explain the problem you have encountered. Perhaps the dealer has experience with the software you are using, but has not encountered your particular problem. Therefore, the dealer tells you that he will contact IBM for help, and will get back with you.

When Your Dealer Contacts IBM

Your dealer has two ways to contact IBM for help. The first option—the one that will give the quickest response—is to use IBM's Customer Support System (CSS).

The Customer Support System gives your authorized dealer several on-line services connected directly with IBM. CSS, which is provided to authorized dealers free of charge, allows your dealer to search for the answer to your problem in a comprehensive data base. This data base contains information about every IBM Personal Computer product that has ever been sold through authorized dealers. The CSS data base is updated daily by the staff of the IBM National Support Center in Atlanta, Georgia.

CSS also allows your dealer to ask the National Support Center a specific question and to receive a response—all electronically. This permits faster service than usual. With CSS, dealers also can receive the latest news and product announcements from IBM. CSS provides product demonstrations for most currently marketed IBM PC software. As evidence of its value, over 1,700 users recently performed over 60,000 transactions in one month using IBM's CSS.

While CSS is available to all Authorized IBM PC Dealers, it has been optional. However, when authorized dealers' annual contracts come up for renewal with IBM, it is now mandatory that they be connected to CSS.

The second option available to your authorized dealer is to call the IBM National Support Center using a toll-free number. An operator will verify that the caller is an Authorized IBM Personal Computer Dealer. Then the operator will take information about the subject of the call, and the call will be placed in a queue. However, if you indicate to your authorized dealer that your situation requires immediate attention, or if your dealer's representative is at your location awaiting response to the problem, the dealer's call is flagged and placed at the top of the queue.

When your dealer's call reaches the top of the queue, a Marketing Support Representative (MSR) calls your dealer to discuss your problem. In many cases, the MSR already knows the answer. IBM's MSRs are well-trained and have experience with many of the products they support, ranging from the PC*jr* to the Personal Computer AT, and from DOS and programming languages to application software. Although it is not a job requirement, most MSRs have purchased IBM Personal Computers and use them at home, thereby gaining additional experience.

If the MSR does not know the answer to your problem, he or she can turn to a number of resources. One such resource is the combined experience of all MSRs in the National Support Center. Another resource is the CSS data base. A third resource is the National Support Center's data base. This is essentially the same data base that your authorized dealer can access via CSS, but in addition it includes items that have not yet been checked for technical accuracy. Finally, if necessary, the MSR can contact the hardware or software development team that has ultimate responsibility for the product.

Whichever method your dealer uses to obtain the information you need, he will call you back with a response as soon as possible. What kind of answer should you expect? In some cases, the response will indicate that you need to install updates to your software. Or you may be doing something incorrectly. Some combinations of hardware and/or software are incompatible. Or the problem may be known, but no solution exists at present.

Whatever the answer, you should be able to proceed, knowing that your Authorized IBM PC Dealer has provided you with the best available information.

What is an IBM Authorized Dealer?

Karen Porterfield
IBM Corporation

What does being an Authorized IBM Personal Computer Dealer mean? First, it means that IBM has thoroughly checked out the dealer—business records, financial background, business ethics—and has found him or her to be a reseller of the highest quality.

Being an Authorized IBM PC Dealer also means that the store's personnel have been educated about IBM's products and sales and service techniques. Authorized dealers are required to take certain IBM training classes and also can attend other IBM-held classes if they wish. In addition, by taking advantage of various IBM sales incentive programs, authorized dealers can avail themselves of further training materials such as educational videotapes, student workbooks and class leader guides.

Being an Authorized IBM PC Dealer means that IBM sales account teams have a day-to-day interface with authorized dealers, keeping them informed of technological advancements and the latest programs. (A new 800 number has been installed so that dealers can order products directly from IBM. If the item is in stock, it can be sent out overnight via express mail delivery.) IBM representatives periodically monitor dealers—often by making on-site visits—to ensure that the dealers continue to meet IBM's conditions for being and staying authorized to sell IBM products. Dealers are authorized for a period of one year at a time. However, they can lose their authorization at any time if they fail to meet IBM's standards.

When you buy hardware from an Authorized IBM PC Dealer, you are covered by IBM's warranty on the product(s). In addition, if you need to have service work done, you can be assured that the work is being done by IBM-trained technicians. If, for some reason, you don't feel that your Authorized IBM PC Dealer has performed well or treated you satisfactorily, you can turn to IBM for resolution of the problem.

How can you tell if a dealer is an Authorized IBM PC Dealer? Every dealer authorized by IBM is given a special decal to display in his or her store verifying that fact. This decal is usually displayed on the door or front window of the dealer's store. The store's service bay area is another place the decal may be found. If you do not see the decal, ask the manager or salesperson to show it to you.

Dealer authorization contracts are for a period of one year only, and may be terminated at any time for different reasons, such as failure by the dealer to live up to service requirements. Since this is the case, how can you be sure that a decal signifies a *current* Authorized IBM PC Dealer? Once a dealer is no longer authorized, IBM asks him or her to remove the decal from display—which in many cases means actually scraping the decal off a surface. An IBM sales representative then visits the store to make sure that the dealer has complied with this procedure.

Results of Survey About Exchange

During November, we surveyed readers for their opinions about *Exchange*. We were very pleased with the responses and would like to share the results with you.

First, I thank everyone who responded so conscientiously and constructively. We were especially pleased to see written comments in about half the responses—a much higher percentage of comments than we expected. From the almost 2,000 responses you sent us, we learned many things.

The average number of readers per copy of *Exchange* is just under three. This means almost 120,000 people read *Exchange*.

Seventy-one percent of you think the technical level of *Exchange* is about right; 15% think it is too technical; and 14% think it's too general. We were buoyed by these statistics. It's difficult to satisfy a wide spectrum of reader sophistication, as your responses confirmed: programmers want more technical material, whereas new users want less technical, more user-oriented articles. So, when a large percentage of you said the technical level of *Exchange* is just right, and the rest of the data fell into an almost-perfect bell curve, we felt pleased with our efforts.

A number of your written comments said you think the articles in *Exchange* are well-written and easy to understand. As an editor, I particularly appreciate these comments. My staff and I strive to make our articles understandable and readable; we're glad you have noticed our efforts.

Articles about software were most in demand. You made it clear you want more articles about DOS features, program utilities, and how to get things done and become more productive. Programmers and experienced users want more articles about programming languages.

In written comments, one clear message came through: don't devote so much space to announcements about new products. Many of you told us you read about IBM announcements in other publications. On the other hand, a lot of you told us you like our material about new products. So we'll compromise: we'll continue to carry new product announcements, but we'll cut back considerably on the details. We are putting the details on our Electronic Bulletin Board System for those of you who want them.

Eighty percent of you like the mix of user-written and IBM-written articles. The rest of you prefer user-written articles, but a number of you want to hear more from IBM. We'll continue to address all these needs.

Ninety-four percent of you feel the appearance of *Exchange* is either attractive and friendly or professional and businesslike. In fact, the distribution is almost evenly split between the two choices. These are further statistics that tell us we're doing the right thing.

Overall, 83% of you rate *Exchange* excellent or very good; 14% rate it average; and only 3% are dissatisfied. Needless to say, we're extremely pleased to receive such overall high marks. These marks certainly encourage us to continue what we're doing.

Responses about continuing *Exchange* are equally positive: 83% would miss it if discontinued, or look forward to reading it; 15% enjoy it, but have other sources; and only 2% are dissatisfied.

Forty percent of you say you would be willing to pay for *Exchange*; 60% are not willing. Given the fact that *Exchange* is free to active members of registered user groups, we found it interesting—and we appreciate—that 40 % think *Exchange* is worth paying for.

A substantial number of you say you do not get every issue of *Exchange*, and you asked us to provide individual subscriptions. This is a good suggestion and we will certainly consider it for the future. For now, if you are an IBM customer and you have a customer number, or if you are an IBM employee, you can get a subscription to *Exchange* and/or order available back copies through the IBM Distribution Center in Mechanicsburg, Pennsylvania. There is a fee for ordering *Exchange* in this manner. I realize this option is limited to IBM customers and employees and is not generally available. However, it is the best we can do at this time.

A great number of you said the thing you did not like about *Exchange* is that it no longer comes out monthly. We sympathize—we'd like to publish monthly, but our budget doesn't permit it at this time.

We'll work as hard as we can to keep *Exchange* as good as it is, and to keep it coming regularly. After all, we want future survey results to be as good as the results you just sent us!

Again, thank you all for responding. Have a happy and productive 1987!

Mike Engelberg
Editor

" You should run CHKDSK often on your hard disk (without the /F option) to catch map damage problems right away. (page 1)

" The PC*jr* is extremely compatible with its big brothers. (page 8)

" We selected PDS Graphs because it let us use input from a PDS Data file, and we could set up procedures to let it run unattended. (page 9)

" The simplest way to speed up your PC*jr* is to add memory. (page 22)

" The PC*jr* is a much more useful machine than it is given credit for. (page 23)

" In IBM BASIC, a single-precision number is stored in memory as 8 bytes. (page 24)

" The Customer Support System gives your authorized dealer several on-line services connected directly with IBM. (page 27)

" When you buy hardware from an Authorized IBM PC Dealer, you are covered by IBM's warranty on the product(s). (page 28)

G320-0854-00