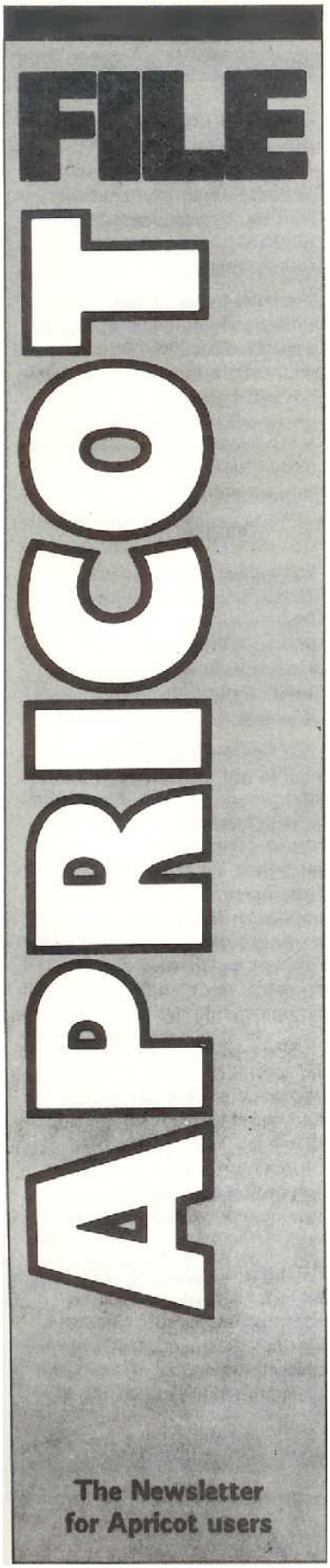
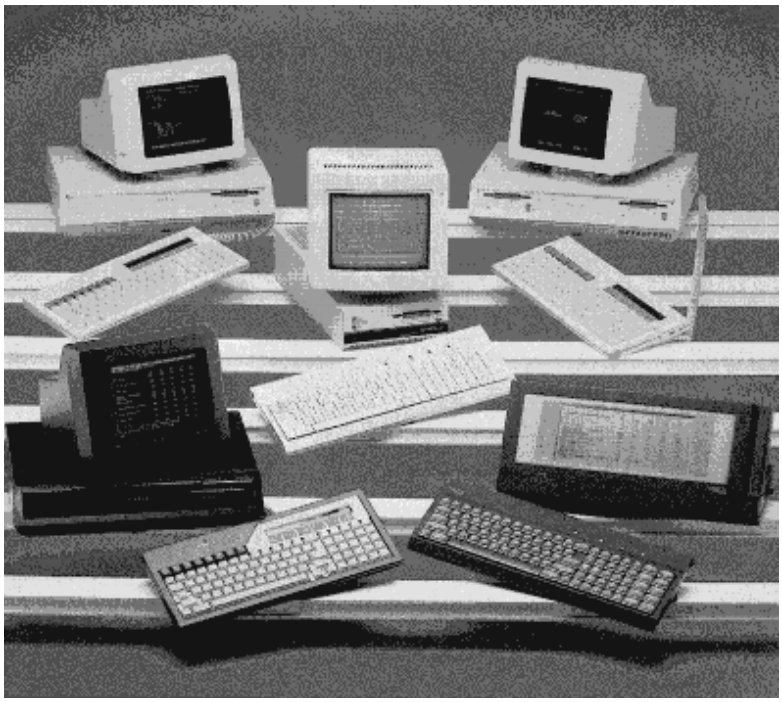


Apricot File 3.11



Scans courtesy of www.actapricot.org

Copyrighted content republished with thanks and permission from:

> Dennis Jarrett of TP Group
> Stuart Bunt
> Eddie Bromhead
> Mark Whitehorn

Many additional thanks to Eddie Bromhead for scanning all of Release 3, 4 and extra inserts.

Release 3.11

FILE APRICOT

The Newsletter for Apricot users

August 1988

Apricot update:

PS/2-compatible computers, new deals on software and support

Apricot has launched a new 386-based product line under the name CHI (no, we don't know what that signifies, either) which apparently features full compatibility with IBM's PS/2.

As we anticipated in Release 3.10, the entry-level machine uses the new Intel 80386SX chip — a hybrid 16/32 bit CPU that allows a price of £1,800 for the CHI 300, a notional system with a single floppy disk. The other machine announced, the CHI 600i, uses a standard 80386.

The major surprise for Apricot's dealers was the inclusion of the IBM-standard Micro Channel Architecture on both machines; there are also built-in drivers for ETHERNET and VGA graphics. The good news seems to be that the motherboards are well-sorted, for a change — not as many patch-wires as at the Xen-i launch.

Deals with Kalamazoo and DG

Apricot has signed a deal with **Kalamazoo** that adds Apricot computers to Kalamazoo's product portfolio — Kalamazoo is committed to taking £5.5 million worth of kit over the next two years, primarily to sell to bigger customers as turnkey systems — in return for the possibility that Apricot will sell Kalamazoo software. Kalamazoo will also be taking on all of Apricot's internal sales training.

The two companies are not dissimilar, in fact; they are located 20 minutes away from each other, both started with a bias towards accounting systems, and they share a slight air of resentment about the way money and attention in the computer business are both centered in the South East.

Other Apricot news includes a "strategic alliance" with **Data General**, which basically includes implementation of Apricot's QUASAR portfolio and trust management software on DG's MV range of 32-bit minis, particularly in harness with DG's CEO 'electronic office' system. Also promised is a 'technology partnership' for possible joint ventures on future products.

QUASAR is selling well in the City, primarily on DEC minis (Apricot Financial Systems has long been a DEC OEM); Apricot recently ported the software on to IBM minis and its own Sequent-based VX machines.

Apricot acts

From *The Lawyer* magazine for 2 June 1988: "Apricot Computers plc is suing NKD Architects and Malcolm C Nicholls. Plaintiffs want exemplary or aggravated damages for infringement of copyright and/or injurious falsehood and/or false attribution of authorship and/or libel and/or slander of goods. Plaintiffs say defendants issued a false advertisement, purportedly on behalf of Apricot. They also want injunctions, all copies of the ad delivered up, plus interest".

In this issue

Disinformation: On environmental impact 2
Notebook: File o' Facts 3 7
The Workshop: Troubleshooting modem connections 8
Free Small Ads 13
The Workshop: A Turbo Pascal Primer — Part Eight 14
Reflection: In praise of Apricots 18
Readers' Writes 21
Notebook: Using printer Escape codes 22
Writers in this issue 25
News and views 26

The Disinformation Column

On environmental impact

By David Langford

The question was posed in last issue's column of *Unanswered Queries*, and I made the mistake of trying to answer it. Days of my life vanished without trace and numerous computers crashed numerous times before I came up with a definite maybe. All right, then: how *do* you modify the MS-DOS 'Environment' information from within a program?

By way of background, Roger Gann's *Workshop* in APRICOT FILE Release 2.9 discusses all the 'orthodox' things you can do with the environment. In this small but mystic block of memory, MS-DOS keeps items like one's PATH and PROMPT information, about which not a lot need be said: both tend to be set via AUTOEXEC.BAT and to require no further attention. (I'll just shyly mention that I've never seen the attraction of stupefyingly long PROMPT strings laden with escape sequences to display reams of inverse-video information while slowing DOS response time to sub-glacier levels. A plain PROMPT pg to tell me the current directory path is as much as I can bear.)

In the unused remainder of the environment, we're told, one can deploy handy user-defined variables which in theory could transfer information between programs and batch files. In practice, Microsoft implemented the whole thing so weirdly as to make it almost unusable.

Environmental facts and fallacies

To clarify what we're talking about: the MS-DOS command

SET LANGFORD=ANSIBLE

will create an environment variable called LANGFORD and give it the text 'value' ANSIBLE. Typing SET alone will now show the line LANGFORD=ANSIBLE as part of the environment information. In a batch file, ECHO %LANGFORD% will henceforth display the word ANSIBLE, and lines like %LANGFORD% or GOTO %LANGFORD% would respectively try to invoke a program called ANSIBLE.COM (or ANSIBLE.EXE), or jump to the batch file label :ANSIBLE.

The LANGFORD variable could be given new values with further SET commands, or cancelled with SET LANGFORD= alone. Until it's cancelled, the variable stays around for the benefit of other batch files and programs. It sounds as though the environment offers a fair measure of added flexibility.

Yes, you can read the environment information from within a program. Yes, you can even write a changed version back to where it came from. You can set a whole new environment for a program which is run by your own.

The snag is that MS-DOS never lets you get your sticky fingers on the master environment. No matter what you do with the copy provided to your program, the changes vanish when the program halts. Run the same program a second time, and a fresh copy of the master environment — *not* including your changes — is what's provided. Only SET commands, at the DOS prompt or in batch file lines, can legitimately alter the master.

Aha! you think. In assembler, Turbo Pascal 4.0, and many other languages, you can execute COMMAND.COM from within the program to run batch files or DOS internal commands: COMMAND /C DIR to give a directory listing, for example. Why not something like ...

COMMAND /C SET LANGFORD=ANSIBLE

... to define an environment variable as above?

Alas, running COMMAND sets up a new master environment block for exactly as long as COMMAND is active; the variable is added to that, and instantly lost when the former environment is restored on return from your invocation of COMMAND.

I've written some demonstration routines in Turbo Pascal 3 (whose *Exec* function, by the way, is distinctly crude and only seems to work with other Turbo 3 programs), and will be scattering them in boxes through what follows. None is a complete program in its own right; but they'll be easy to put together or to translate into assembler. My initial researches were based on the few facts about the environment which have been made available to programmers.

1. The environment information is stored as a series of ASCIZ strings, being ASCII text strings whose ends are marked by 'zero' characters. (Basic CHR$(0), Turbo *Chr(0)* or simply #0.) The whole block of information ends with a double-zero. If we show the zero characters as [0], a typical environment might be held as PATH=A:\[0]PROMPT=pg[0][0].

2. The information always begins at a RAM paragraph boundary, ie a memory address divisible by 16 (10 hexadecimal). Its address can thus always be specified as a 16-bit segment address with zero offset.

3. The segment address of the copy of the environment supplied to your program is provided as just such a 16-bit word, located at offset 2C hex in the 'Program Segment Prefix'. The PSP itself is a large

ragbag of assorted information. In COM programs as produced by Turbo 3, it occupies the memory before the program start address at 100 hex in the code segment — *Cseg*. In EXE files you officially locate it by using the MS-DOS INT 21H call, function 62H. This returns the PSP segment address in BX, but only in MS-DOS 3.0 or higher. Immediately after loading, EXE programs are also supposed to contain the PSP segment in the DS and ES segments; and Turbo 4 obligingly provides it in a predefined Word variable called *PrefixSeg*.

4. The Apricot environment space is 256 bytes long, unless you're using an imported COMMAND.COM from elsewhere. Obviously some very nasty things might happen if while trying to change it one wrote beyond the available space and perhaps corrupted MS-DOS, the command interpreter or the upper echelons of the police force.

Since we have to start somewhere, let's begin with Turbo code to read and display the copied environment. I'll also define a few variables in advance for later use.

```
Const
  EnvStrLen  = 255;
Type
  EnvString  = String[EnvStrLen];
  MiniString = String[40];
Var
  I: Byte; {All-purpose}
  Environment: EnvString;  {Used throughout}
  EnvSeg: Integer;  {Current environment
                     address}
Procedure ReadEnv;
  {Assumes that you've set EnvSeg}
  begin
    I:= 0;
    Environment:='';  {null string}
    Repeat
      Environment:=
      Environment+Chr(Mem[EnvSeg:I]);
      I:= I+1
    Until (I>EnvStrLen)
       or (MemW[EnvSeg:I]=0)
       or (Environment[I] in
       [#1..#31,#127..#255]);
   Environment:= Environment+#0;
  end;
Procedure DisplayEnv;
  begin
    For I:= 1 to Length(Environment) do
     If Environment[I]=#0 then Writeln else
     Write(Environment[I]);
    Writeln;
  end;
```

This allows you to look at the environment copied to your own program, by use of the following three lines:

```
EnvSeg:= MemW[Cseg:$2C];
ReadEnv;
DisplayEnv;
```

The *MemW* 'access memory' call here reads the environment address from a PSP assumed to be in *Cseg*; in Turbo 4 you should substitute *PrefixSeg* for *Cseg*. The use of *MemW* in the 'until' condition of *ReadEnv* spots the double-zero which concludes the environment text. Note that the code has been written so that your copy of the environment ends with a single rather than a double zero. This will later make it easy to add strings to its end.

Henceforth, your program can extract any information it needs from the string *Environment*. If the hypothetical variable LANGFORD is of interest, the function call ...

Pos('LANGFORD=',Environment)

... will report its position in the string (zero if it's not present) and allow its value to be read via the Turbo string procedures.

Obviously it's easy enough to modify the local environment copy, remembering that this will only affect programs which are actually run by your own program via Execute or some similar function. Procedures for doing this follow.

```
Procedure AddToEnv(NewStr:MiniString);
  begin
    If Length(Environment+NewStr)<EnvStrLen
    then
      Environment:= Environment+NewStr+#0;
  end;
Procedure ZapEnvVariable(VarID:MiniString);
  Var
    Len: Byte;
  begin
    I:= Pos(VarID,Environment);
    If I>0 then begin
      Len:= 0;
      Repeat
      Len:= Len+1
      until Environment[I+Len]=#0;
    Delete(Environment,I,Len+1);
    end;
 end;
Procedure WriteEnv;
{Assumes EnvSeg set.}
  begin
    For I:=1 to Length(Environment) do
     Mem[EnvSeg:I-1]:= Ord(Environment[I]);
    Mem[EnvSeg:Length(Environment)]:=0;
  end;
```

Armed with these three, and having already read the environment as above, we can write back a perverted version by some such means as this:

```
ZapEnvVariable('LANGFORD=');
AddToEnv('LANGFORD='+Chr(64+Random(60)));
WriteEnv;
```

... which first performs the much-needed task of removing any Langfords which may already be

cluttering up the place, and then inserts LANGFORD=X (where just for fun X is a random character) into the environment. *AddToEnv* contains the required length check. At present, of course, the effect is only local and temporary.

To make it permanent and of use to subsequent programs and batch files, we must get hold of the master environment which has to be out there somewhere in memory. Microsoft have made this as difficult as they possibly can: I know of no documented or undocumented MS-DOS function which returns the address of this block of information, although there quite probably is one.

Secrets of the universe not revealed

A digression: many things exist unadmitted in MS-DOS. I was a bit peeved by Allen L Wyatt's *Using Assembly Language* (published 1987 by Que, UK price £19.95), which your editor provided for review here, because Wyatt slavishly lists well-known undocumented functions and interrupts as "reserved ... not publicized" without ever mentioning the extremely useful things they do.

For example, interrupt 21H, function 34H returns (in ES:BX) the address of a one-byte flag whose value is zero if and only if MS-DOS is not currently busy (indicating that it's now safe for memory-resident programs to use DOS facilities). Interrupt 28H is called when MS-DOS has just stopped being busy — another boon to writers of resident utilities. Interrupt 29H displays the character in AL and is the fastest screen output route which is unofficially compatible with all MS-DOS machines (though on some IBMs the call can corrupt the BX register, necessitating a PUSH BX ... POP BX safety-net). This is the sort of arcane wisdom one hopes for in books by allegedly experienced programmers.

Wyatt is reasonably good on listing well-documented information that I already have — like IBM BIOS routines and video memory layout, the extra function calls in MS-DOS 3, and the instruction sets of not only the 8086/8088 but also the 80286, 80386, 8087 and 80287. Indeed, this kind of reference material occupies about 649 of his 745 pages. Conversely, the chapter 'Interfacing with Pascal', about which I could write volumes, runs to just 13 pages and skips the really dodgy questions.

Throughout, Wyatt's idea of "using assembly language" is confined solely to the writing of assembler routines which plug into high-level languages. Despite what might reasonably have been assumed from the title, you literally are not told how to construct the simplest stand-alone assembler program. Tut tut.

Naughty practices

Meanwhile, back in the department of the environment, you will be agog with suspense. Can one indeed meddle with the master environment?

Well, yes, you can. But the method is ideologically most unsound, and if Microsoft ever heard about it they'd come and wave magnets over all your disks.

After lengthy trawling through undocumented DOS functions, I tried a brute-force search which ve-e-ery slowly climbed from the bottom of memory in search of something starting at a paragraph boundary and looking exactly like my search program's local environment. (If the local environment is an exact copy of the master, obviously the reverse must apply.) This was the point at which I found that the master environment, being associated with the command interpreter, was not located down in the bowels of MS-DOS but on the wrong side of my 448K RAMdisk. Hours of fun, you bet.

The search was ultimately successful, but suggested new complexities. The first copy of the environment might not be the current master, since mightn't a later COMMAND/P have 'taken over' the mastery? The last copy — the one that was highest in memory — would be the local environment of the search program, of course. Somewhere before that came the Turbo compiler's own PSP and environment (not present if the search program were compiled and run from MS-DOS rather than by Turbo in memory). Before that would be the fossil remains of various memory-resident utilities, each presumably carrying another confusing copy of the master environment ... There are rather too many possibilities. Here's the thoroughly bodged-up routine which ensued.

```
Procedure FindMainEnv;
  Var
    LocalEnvironment: EnvString;
  Const
    Countdown: Byte = 1;  {Test version: 1 for
                           COM file, 2 in Turbo}
  begin
    EnvSeg:= MemW[CSeg:$2C];
    ReadEnv;
    DisplayEnv;  {Optional. Remove for most
                  uses.}
    LocalEnvironment:= Environment;
    Repeat
      EnvSeg:= EnvSeg-1;
      ReadEnv;
       If (Environment=LocalEnvironment) then
       Countdown:= Countdown-1;
       until Countdown=0
  end;
```

This version works backwards from the *EnvSeg* address of the program containing it. (I tried alternative methods, such as calling MS-DOS function 35H to read the INT 21H segment address and working up from there ... only on the IBM I had to head *down* from there!) If there are no memory-resident programs to confuse the issue, the first copy of the environment found will be the

master; and *EnvSeg* will be left appropriately set so that the routines previously described can be used without modification to mess with this master.

The all-important fudge factor is the variable *Countdown*, which during development I set to 2 in order to skip over the environment of the Turbo Pascal compiler (I was using Turbo to run the test program in memory). Judicious experimental tinkering with *Countdown* could 'tune' the routine to skip an appropriate number of memory-resident items or whatever, thus adjusting it rather too precisely to the current system configuration. See below for a better approach.

Beware: the more *FindMainEnv* has to skip, the longer its delays will be. The speed is made as high as possible by the final exit condition of the *Repeat* loop in *ReadEnv* (above). This is unnecessary when you're dealing with a 'real' environment, but here it helps screen out duff segments by stopping the loop as soon as we read a character not normally found in the environment.

A bit of a bodge, as I said, and you wouldn't want all your programs to go through such a potentially sluggish process. But consider this. Suppose our program to seek the master environment were run first thing in AUTOEXEC.BAT, right after COMMAND.COM. Its search would be quick and certain of success. (Oops ... be sure to SET something in the environment first, even if it's only a dummy variable like LANGFORD=ANSIBLE. The *FindMainEnv* routine needs a bit of text to scan for.) Suppose the program then celebrated its success by writing the address of the master environment segment *into the master environment*, as ENV=5000 for instance. Every subsequent copy of the environment would then contain this information: your other programs could read the master address straight from the local environment. Here's how.

```
Procedure RecordEnvSeg;
{Assumes EnvSeg addresses master environment}
  Var
    TmpString: MiniString;
  begin
    Str(EnvSeg,TmpString);
    ZapEnvVariable('ENV=');
    AddToEnv('ENV='+TmpString);
    WriteEnv
  end;
```

Now we're really in business. A program containing this routine and the appropriate selection from those above (*Countdown* set to 1) would record the all-important master environment segment address for retrieval by future programs. The appropriate sequence of calls would be simply:

```
FindMainEnv;
RecordEnvSeg;
```

... and henceforth every new environment copy contains the ENV variable with the master address. Subsequent programs can now access the master environment by setting *EnvSeg:=ReadMasterSeg*, where *ReadMasterSeg* is a function defined as follows:

```
Function ReadMasterSeg:Integer;
{Assumes prior use of RecordEnvSeg by some
program}
  Var
    Len: Byte;
    Num,Code: Integer;
  begin
    EnvSeg:= MemW[Cseg:$2C];
    ReadEnv;
    I:= Pos('ENV=',Environment);
      If I=0 then ReadMasterSeg:= 0
     else begin
      I:= I+4;
      Len:=0;
      Repeat
       Len:= Len+1
       until Environment[I+Len]=#0;
      Val(Copy(Environment,I,Len),Num,Code);
      If Code=0 then ReadMasterSeg:= Num else
      ReadMasterSeg:=0
     end
  end;
```

Note that the function has been set up to return a zero value if no ENV=nnnn variable can be found in the environment, or if Turbo's *Val* procedure can't convert the 'contents' to an integer. (The I:=I+4 is there to prevent the ENV= part of the string from attempted numerical conversion; remember to change the '4' if you decide to use a name longer or shorter than ENV.)

In practice, potential confusions can be eliminated by running your *FindMainEnv/RecordEnvSeg* 'setup' program from a batch file, immediately preceded by SET ENV=*. This will alter the master environment and thus the local environment picked up by our program, but *not* any other discarded environment locked into memory by, say, a resident utility. *FindMainEnv* is thus sure to locate the true main environment only, while *RecordEnvSeg* will replace the dummy ENV=* variable with the proper value. No unnecessary clutter is left in the precious environment space.

All these routines are lifted straight from a demo program which does in fact work. Yes, it's a bit tortuous; but it *is* an answer.

Other paths to glory

Can we do without program-defined environment variables altogether? Probably. For simple passing of information back to batch files, much can be achieved using the ERRORLEVEL function. (Whose name, in many contexts, is misleading.)

Suppose you want your Turbo program YOURPROG.COM to decide, as it exits, which of two

other programs in the calling batch file should now be run. Use the optional form of the *Halt* procedure which lets you set the 'error level' as you leave Turbo: *Halt(0)* or *Halt(1)*, say.

Here's a familiar section of batch file which responds to the setting:

```
YOURPROG
IF ERRORLEVEL=1 GOTO PROG2
PROG1
GOTO FINISH
:PROG2
PROG2
:FINISH
```

Assuming the YOURPROG source code ends with *Halt(Err)* where *Err* is an Integer, the value of *Err* determines the batch file ERRORLEVEL. If *Err* is 1 or more, the batch file will jump to the label :PROG2 and run PROG2; if zero, it will run PROG1 and then skip over the PROG2 command.

Note that doing it the other way around with IF ERRORLEVEL=0 won't work properly: ERRORLEVEL=*n* returns a 'true' value if the 'error level' is equal to *or greater than* the number *n*.

Trying to use this facility for multiple conditions becomes boring, with a long series of tests for descending ERRORLEVEL values. Tinkering with the environment does at least short-circuit this:

```
SET YOURCMD=
YOURPROG
%YOURCMD%
```

If the environment variable YOURCMD here isn't given a value by YOURPROG, nothing will happen; but in theory it can be set to the name of any program, batch file or internal MS-DOS command by using the brute-force methods I've described.

In assembler, by the way, the recommended way to set ERRORLEVEL as you terminate a program is to put your desired value into the AL half-register and call INT 21H, function 4CH. Thus, to set a value of 3:

```
MOV AL,3
MOV AH,4CH
INT 21H
```

Or you could streamline the process by loading AL and AH together:

```
MOV AX,4C03H
INT 21H
```

Quickie: diagnostic aids

As you might imagine, when developing the environment routines I plastered them with display calls which reported what was going on, which memory segment was being scanned, etc. For the sake of sanity, compatibility with reference books, and avoidance of Turbo 3's confusing negative-integer display of hex values over 7FFF, I used the following little hex-string routines. Always handy things to have around:

```
Type
  ByteStr= String[2];
  WordStr= String[4];

Function HexByte(Num:Byte):ByteStr;
  Const
    HexCh: String[16] = '0123456789ABCDEF';
  begin
    HexByte:= HexCh[1+(Num shr 4)] +
              HexCh[1+(Num and 15)];
  end;

Function Hex(Num:Integer):WordStr;
  begin
    Hex:= HexByte(Hi(Num)) + HexByte(Lo(Num));
  end;
```

A note for the inexperienced: we extract the low half of a byte by 'masking'. The '15' is 00001111 in binary, and the AND operator zaps all the bits of *Num* which aren't set in this 'mask' constant. Getting the high half of the byte essentially means dividing by 16: the *Shr 4* is merely the most elegant way of doing this. Turbo provides the *Hi* and *Lo* functions to spare us the equivalent task of breaking an integer into its high and low bytes.

Thus a inserted statement like *Writeln(Hex(EnvSeg))* was useful to keep track of the *FindMainEnv* search routine's painful crawl through vast trackless deserts of inhospitable RAM. In fact, while testing, the exact statement I used was *Write(^M,Hex(EnvSeg))*: the return character *^M* in the absence of a linefeed causes each new string to start at the beginning of the same line and to overwrite the previous one — less irritating than an endlessly scrolling screen.

Tailpiece

Recently I gazed upon the most awesome piece of ergonomic design yet to emerge from the wonderful world of computing — the Apricot plc Disk Mailer! Let us give credit where credit is due. As we all know, the 3½in disk is actually a mite larger than its nominal size — about 3 17/32in by 3 11/16, I make it. Apricot presumably thought long and hard about this when designing their mailer, which consists of a couple of polystyrene slabs with embossed Apricot logos and a shallow disk recess on each side of each slab. The recess measures 3 1/4 by 3 3/8in. Please oblige me by meditating on this beautiful thought. ◆

File o' Facts

Cries and whispers

Notes from the working week (3)

By Eddie Bromhead

C Itoh Printer Blues

I finally got fed up with the way Microsoft Fortran, WordStar and my C Itoh (ACTWriter) printer interpreted tabs in my source code all in different ways; and, provoked by the CPM print formatter on a DEALS FOR READERS disk, I decided to write my own.

All it needed to do was to set up tabs on the printer, then copy the file line by line with appropriate gaps every page for the perforations. (Actually, this five-minute job turned into a marathon, but that's another story.)

But when reading my trusty printer manual, I discovered an amazing bodge-up. The lead-in ESCape sequence for setting tabs is ESC (and they are removed by ESC) — followed in both cases by column numbers at which tabs are to be set or cancelled. The poor old typesetter handling the printer manual had been awfully confused by what appeared to be mismatched parentheses, so he added a few himself or he interpreted likely symbols as being poor copy in the first instance.

To set tabs at columns 10, 20 and 30, you would send ...

```
LPRINT CHR$(27) + "(010,020,030."
```

And to switch off the one at column 20:

```
LPRINT CHR$(27) + ")020."
```

Easy, isn't it? Now just compare this gem from the ACTWriter manual (and this is only part of it!):

> Note:
> - Each three digit N2-Nn must be separated from a preceding N1–Nn-1 by a command. (ASCII or hexadicimal 2C)
> - Receipt of the ESC (command clears all horizontal tabs previously set.)
> - A period (ASCII or hexadecimal code 2E) terminates the ESC (and ESC) commands.
>
> Example: If "ESC (006,018,036)" is sent to the printer. Tabs will be set in print locations 18 and 36.

In English, separate the three-digit column numbers by commas (*not* by commands!) and finish the list with a full stop. ESC 0 clears all tabs previously set, or they can be selectively cleared with ESC); or a new ESC (command will effectively clear existing tab settings — it must be followed by the whole list of tab settings, because consecutive ESC (commands aren't additive.

In the manual's example, what happened to the tab set at column 6? Why didn't the printer at C Itoh's works crash when the example command was terminated a bracket rather than a full stop?

The moral of this story is that although I've had that printer for 4½ years, I still haven't used a fraction of its facilities. It's the same with the Apricot(s), the plotter(s), the software ... And to think that I thought I'd been hard at it!

MS-DOS 3.20

MS-DOS 3.20 for the Portable? It installed itself fine on my Xen HD (but then so did an F2's MS-DOS 2.11). I wanted 3.20 for the system utilities, things like ...

APPEND.COM A sort of PATH command for data — prefixes all data files with a pathname. Exists in 3.10 too.

ASSIGN.COM Calls drives by different names. Useful if your software demands to install itself on drive C and your hard disk is A.

ATTRIB.EXE We know the principle as CHMOD: sets a file to read-only (and/or back again).

AVERIFY.EXE Verify archived data restoration.

JOIN.EXE Adds part of one device's directory structure to another's. All a bit academic to me!

REPLACE.EXE Selective copying.

SHARE.EXE Only if you're in a network.

SUBST.EXE This removes the need for DPATH, SmartPath, and all the rest. You can make \WORDSTAR\DOCUMENT\ be drive E and then life gets oodles simpler with programs that don't understand subdirectories.

TREE.EXE Yes, it's in 3.10 too. Tells you the tree structure of subdirectories on a disk. TREEDIR from DEALS FOR READERS is better.

WHAT.EXE Elements of DEBUG: searches for plain text in EXE and COM files, and reports it. Doesn't work (when you want it to!).

XCOPY.EXE Selective copying. (This is in the bit I didn't write in Release 3.9!)

(Actually, all I wanted was SUBST!) What about a manual? There's a good one with the Apricot PC/S (well, better than you got with a real Apricot). Or use Chris DeVoney's MS-DOS book. Or get Amstrad's manuals — though not for WHAT or AVERIFY. *(We also like, use, recommend and sell Van Wolverton's Running MS-DOS — Ed.)*

Suddenly all my subdirectories appear with a date and time. Is this normal?

LCOPY in GWBASIC

Q. Where do you find LCOPY in GWBASIC?

A. In the version supplied on DEALS FOR READERS disk AP040, MS-DOS 3.20 for the Portable.

Q. Where do you find the documentation?

A. Um, er ... ◆

Troubleshooting modem connections

By Simon Potter

We're currently preparing a 'Best of COMMS FILE' book that should provide an excellent all-purpose introduction to comms on the computer by mixing articles from our COMMS FILE newsletter (revised and updated as appropriate) with new material. In the latter category comes this guide to sorting out problems with modem connections.

In the litany of computer-related problems, connecting and using a modem rates up there with the tribulations of attaching a printer. Sometimes it's a doddle: you plug the thing in and it works first time. Such blessed examples are sadly rare.

This piece is written in generic terms to apply to *any* user of modems, and as a result includes some information that won't be strictly relevant to the Apricot fraternity — in particular, the notes about COM ports and Hayes-compatibility. But there should be enough here of relevance to any Apricot user who is considering adventures in the telephone trade ...

■■■ Preliminaries

There's always a good reason why a modem connection fails to work. It's not always obvious, however, just because there are so many possible causes. In essence, things to check are:

- the phone line
- connection to the phone line
- the modem itself
- connection between the modem and the computer
- the serial RS-232 socket (or 'port') on the computer
- whether the RS-232 port has been set up correctly for the computer
- the parameters set by comms software
- the number you're calling
- log-on procedures for the service you're calling

This article offers a fairly rigorous checklist approach that should identify the hiccup. But first, take note of these suggestions:

Check the obvious things first (not least because they're the easiest aspects to check). This includes power supply, the connecting cables, and whether the phone line is working.

Try substitution before disassembly. If you're sure that the software and comms port parameters are set up correctly, try to get hold of a modem and/or a cable that you *know* work correctly. When things still don't work, you can be sure that the problem lies inside your computer — in the RS-232 connection or in your comms software. Otherwise, it's your modem and/or cable.

Take notes as you go — if you do have to phone a specialist (like the modem supplier or a service engineer) it will be helpful to know exactly what the problem is and what you've tried yourself. Things to log include any sign of activity at the modem (flickering LED lamps etc), error messages that appear on the screen, wisps of smoke, etc.

As a last resort, **read the manuals** for the modem and the comms software. That's offered as the last-ditch option because modem documentation in particular can be notoriously unhelpful, but things are generally improving under the pressure from market forces. Anyhow, the answer might just be buried somewhere in there ...

And now to the specific fault-identification procedures:

■■■ Problem: The screen is blank

1. **Is the telephone socket operational?** Try plugging a phone directly into the socket; do you get a dial tone? If yes, try dialling someone (like '123' for the speaking clock). And try dialling your own number too — an engaged tone means it's working. If the phone socket isn't working, call British Telecom service engineers; and, if possible, try your modem on a different socket and/or a different line. Otherwise ...

2. **Check that the cable between the modem and the telephone socket is seating properly at both ends.** To improve the connection, try wiping the business ends of the plugs (the connector strips embedded in the plastic) with a solvent cleaner — *not* a general-purpose domestic product like Mr Sheen; use an alcohol-based cleaner like computer cleaning fluids or a cassette deck tape head cleaner. There's only way that the standard 'new-style' plug can fit the socket, and a plastic spring-clip should hold it firmly.

3. For internal modems: **check that you have followed the installation instructions to the letter.** Remove the modem card, wipe the edge connector with the same

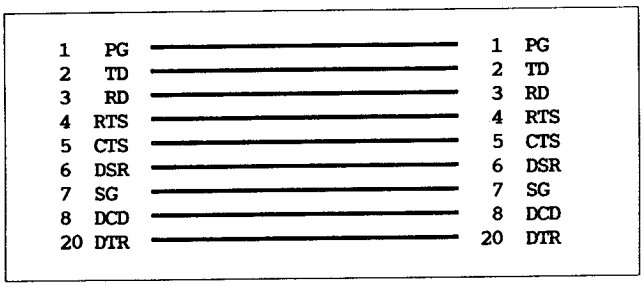
kind of cleaning fluid, and replace it firmly in its slot to ensure that it is seated correctly. If that doesn't work, try the modem in another slot (if one is available, and if the installation instructions allow you to use a different slot). Go to step 9.

4. For external modems: **make sure you have the modem cable connected to the right socket on the computer.** In particular, users of IBM-compatibles should be sure that the RS-232 modem cable isn't attached to the *printer* port — some machines use a *parallel* connector for printers that looks very like the standard D-shaped *serial* connector for RS-232 devices. Otherwise ...

5. Now **check that the** RS-232 cable is connected correctly at both ends (at the modem and at the computer's RS-232 port). Most RS-232 plugs can be screwed or clipped tightly into their sockets; do that, because it ensures the best possible connection.

6. **Is there any activity from the modem's** LED indicators when you try to make the connection? You should at least have a power-on light (the absence of which suggests one avenue to explore before going any further); most modems also run an automatic self-test on start-up, and if that is completed successfully at least one LED should light up to indicate 'ready to send/receive'. Check the handbook for this; if the relevant light isn't lit, the problem probably lies within the modem. Otherwise ...

7. **Check that you have the correct** RS-232 **cable** — does it apply to your make and model of computer? And to your make and model of modem? Most modems and most computers require a standard connector, but some variation is possible — particularly if you are not using the modem with an IBM-compatible machine (people with BBC Model Bs, for instance, get used to mating a five-pin DIN plug with a DB-25 connector). Modem vendors usually supply a 'generic' cable, but the modem's handbook should indicate what other varieties of cable the manufacturer can furnish. If in doubt, check with the modem vendor. And if a standard cable is required, these are available off the shelf from many suppliers (including TP Group's DEALS FOR READERS service) — or make one up yourself, as follows:



8. For external modems: if you still have problems, **try some substitution** if you can.
   - Will some other device work on the RS-232 port with a different cable? If you have a printer, plotter or something else that normally attaches via the RS-232 interface, try that in place of the modem and its cable. If another device works when the modem didn't, you know the port is functioning ok and the problem lies elsewhere (probably — but not necessarily — in the modem or its cable).
   - Will the modem work with a different cable? Beg, borrow or buy another modem cable and try it with the same modem on the same RS-232 port. If that works, the problem lies with the modem; if it doesn't work, on the other hand, the cable could still be inappropriate to your particular setup.

9. For any modem: still no joy? **Check that the correct** COM **port is being used.** In most cases there won't be a physical choice — your computer will have only one RS-232 socket. But some computers have two or more, and all modern computers have built-in facilities for identifying several COM ports and directing their attention accordingly. If your computer has been told to use port number 2 when there isn't a second COM port fitted, you might not be told that the operating system is trying to address this phantom. If you do have a second COM port, you obviously won't get too far if the modem is plugged into COM 2 when your software is trying to send and receive at COM 1. And do make sure that you don't have two devices set up as using the same COM port — if two such devices are permanently plugged in, it's likely that a printer will be on COM 1 and your modem on COM 2. Telling the operating system that both devices are on the same port can produce some weird effects ...

   Most comms software includes a 'utilities' or 'parameters' facility with which you can specify the COM port to be used for modem communications.

Once you have set this up and saved the setting, it should apply every time you utilise the program subsequently. A few comms packages don't have this flexibility, however, and they invariably expect the modem to be on a particular port. In that case you'll have to check the software manual to make sure you're plugged into the correct one.

In fact, specifying the COM port is not usually a major problem. A more likely source of your troubles will lie in the comms parameters being applied to that port. So ...

10. **Check that the comms parameters are set appropriately.** The key determinant is what's required (and expected) by the service with which you are trying to connect. The relevant parameters are ...
   - *line speed* or 'data rate'. This one is also restricted by the capabilities of your modem, of course; a 300 bps modem can't be set up to work at 1200 bps, and a modem which can't work at the V23 'split speed' can't be set up for 1200/75 bps comms
   - *parity* can be ODD, EVEN or NONE. This refers to a simplified method of checking for errors in a transmission, and for it to work both ends of the link must be set up in the same way
   - *number of data bits* can be 7 or 8. Again, both ends must agree on the format.
   - *number of stop bits* can be 1 or 2.

11. **Check for switches and straps.** Some modems include external (and/or possibly internal) DIP switches or jumper leads to define certain operational characteristics. For example the Dataflex Stradcom, an internal modem for IBM-compatibles, has a bank of four on-board switches to select the COM port on which the modem is installed; the Tandata Tm722 (an external modem) has a bank of four DIP switches on the rear panel to select aspects of RS-232 operation.

12. **Make sure your comms software matches your modem.** The first consideration is Hayes-compatibility, meaning that the software will be trying to use Hayes-type commands to control modem; *such software will not work unless the modem too is Hayes-compatible*. Similarly, a Hayes-compatible modem probably won't work with a comms package that *doesn't* use Hayes-type control codes. Though a few such modems can handle either Hayes-compatible or ordinary modems, it's safest to stick with these tenets:
   - Hayes-compatible software needs a Hayes-compatible modem
   - a Hayes-compatible modem needs Hayes-compatible software
   - other comms software needs a non-Hayes modem
   - a non-Hayes modem needs a non-Hayes comms package

13. If possible, **go for an even closer match.** Most comms packages come with a 'modem set-up' facility that enables you to specify the modem you're using with your software; but not all modems will be included in the list (there are just too many alternatives, with new products appearing all the time). If in doubt, try the lowest-common-denominator approach.

14. For Hayes-compatible modems: **Make sure that your modem is as Hayes-compatible as your comms software.** Some modem-makers have not followed the Hayes format for Hayes-type commands with absolute precision — not all the commands may be implemented, and the modem may reject the software's attempt to pass it commands if they aren't in the format it wants (a true Hayes modem, for instance, will accept Hayes-type commands if they arrive in upper or lower case: some other modems won't take Hayes control codes if they aren't in upper case — and so on.

15. **Do you have automatic 'line-sensing' on the modem?** Some modems can automatically detect the line speed and parity required by an incoming signal, and they adjust their own RS-232 ports accordingly. This may have side-effects that can only be overcome with specially-written software; in those circumstances the modem supplier will normally 'bundle' a suitable comms package with the modem. The implications of *that* could be headaches and frustration if you are not using the obligatory IBM-compatible on which the required comms package will run.

16. For auto-dialling modems: **can you hear the clicks as the modem dials the number?** If not, the likely reasons are that the modem isn't happy with your comms package — specifically, it isn't picking up the auto-dial instructions being sent by the software. This could mean that you need a different modem or (more economically) a different comms package.

Alternatively, you didn't enter the number correctly when setting up your

list of phone numbers. Did you include an unnecessary STD-code prefix? And check the format required by the comms package: have you used spaces or dashes which the software prohibits?

17. Still no contact, even though the dialling appeared correct? **Check that you have a carrier tone** (lift the handset and listen). If there isn't a tone, you called a wrong number; or the computer is not ready to receive you; or the remote service supplier hasn't been paying the phone bill.

18. For Hayes-compatible modems: did the dialling sound like a fast series of tones (such as you get when dialling on an all-digital phone system, like those in the States and elsewhere)? Or was it a more sedate collection of bips, corresponding to the old UK-style rotary telephone dial? The former is called 'tone' dialling, the latter is 'pulse' dialling. A phone system that works with one won't work with the other. At present, if you're connecting directly to British Telecom phone line you'll normally have to use *pulse* dialling (because the PSTN isn't yet all-digital). If you're going through a company switchboard you might have to use *tone* dialling (because most modern PBXs are digital, and automatically convert the internal digital tones into pulse signals for outgoing calls on the PSTN). So **check that the comms software is telling the modem to use the right kind of dialling**. Somewhere in the set-up sequence for the modem you'll find an entry called something like 'modem initialisation' that includes the Hayes command ATD. If it's actually ATDP, you're set up for pulse dialling; ATDT sets up for tone dialling. Alter this as appropriate.

19. Still no carrier tone? **Are you dialling out through a switchboard?** If yes, be warned that many digital exchange lines and PBXs will accept both pulse and tone dialling. If the modem decides to use tone dialling, there may be only a single click as the line relay closes prior to dialling. And make sure your auto-dialling numbers include the '9' or '0' (or whatever) to get an outside line. Note that some PBXs require a pause after the '9' to give them time to assign an external line to your call; check the delay between picking up the phone and getting a dial tone. If is more than about three seconds you will probably need to program some additional delay commands into the modem. Your comms software may not be able to provide that delay, in which case the auto-dialled number might just get lost in the system before you do get the outside line.

20. **Are you on a party line?** These have been known to produce considerable difficulties in making a connection, though to be fair the major problems seem to apply to keeping the connection going — the signal appears to be much fainter on party lines, with the result that midway through a session the modem might decide that it can't hear the carrier any more ...

21. **Does the modem have a Reset button?** If not, your comms software may allow you to send the modem a 'reset' instruction. Use one or both if you think that your experimentation might be starting to confuse things.

22. So you have a carrier tone and your system thinks it has done everything correctly to make contact — but the screen remains blank? **Maybe the remote modem needs to be 'woken up'** — to be told to receive an incoming call. Press the Return key a couple of times; if that doesn't work, try CONTROL-E. Still no response? Call the remote service on a voice line and ask what 'wake-up' character you should be sending.

■■■ Problem: duff contact

23. **If the screen fills with junk characters,** there are three likely possibilities.

- you're using the wrong *data format* for connection to that particular service. Try different combinations of setting for data bits (7 or 8), parity (EVEN, ODD, NONE) and stop bits (1 or 2).
- you're using a different *protocol* from what the remote service expects. Check whether the remote service expects you to be using straight ASCII (in which case concentrate on getting the number of stop bits correct) or XON/XOFF (your comms package should allow you to select the latter).
- there is *noise* on the line. Try picking up the telephone handset; can you hear a more or less pure carrier-tone whine? Extra fizzes and burbles will corrupt the transmission; so will a crossed line. The problem may just be inside your building (might be the proximity of portable cellphones, microwave ovens or other

appliances that generate radio-frequency signals — though in fact, stray radiation from a microwave oven is unlikely to interfere with a modem unless it is at health-hazard levels; but switching the oven on and off might cause trouble). More likely it's the PSTN; bad weather, roadworks or just overloaded circuits in an aging telephone exchange can all cause problems — circuits can be overloaded by illegal signal levels, typically from Bell-standard (non-BABT-approved) phones or modems; any exchange may be overloaded by more calls than it can handle; and old (Strowger) exchanges cause more than their fair share of clicks bangs and twangs (crossed lines rarely occur on non-Strowger exchanges). If the problem looks like the PSTN and if it occurs only occasionally, give up and try later ...

24. **If the screen occasionally fills with junk characters**, it could be intermittent noise on the line. Retry, or try again later. Alternatively, the remote service could be sending something that your computer can't display properly; this usually means graphics characters that your end is unable to convert and display correctly. You'll just have to live with this unless you can find a way of setting your system up to decode these graphics.

25. **If incoming characters are recognisable but meaningless**, chances are the text is intended for special conversion before it becomes legible. Perhaps it's encrypted for security purposes and you don't have the decryption facility; or maybe it's intended for an intelligent terminal, and your system can't emulate that particular device.

27. **If the characters you type don't appear on your screen at all**, you aren't getting what's called a 'local echo'. Most remote services 'echo' the transmissions they receive, meaning that each character sent to the host is immediately copied back to your screen as a visible record of what's been transmitted. If the host *doesn't* do this, you need to switch on a 'local echo' facility at your end; that way your comms software will send characters to the screen *as well as* to the modem — otherwise the screen remains blank. Your comms software should include an option that allows you to local echo (it may be shown as 'echo on').

26. **If characters appear twice**, you have 'local echo' switched on when the remote host is also echoing the transmissions received: each character sent to the host is copied back to your screen *and* your comms software is also sending characters to the screen. As above, your comms package should allow you to switch to 'local echo off'.

28. **If you can make contact but the remote service refuses to accept your log-on codes**, maybe you mistyped the identifiers — or maybe you've offended someone (by not paying your e-mail bill?) and have been bounced off.

Problems after connection

29. **If the screen fills with junk characters after you've successfully logged on**, the same considerations might apply as for 15 above. Or you might be trying to pick up a binary file (a spreadsheet or a program, typically) when your comms package is expecting straight text. To download anything but straight ASCII you'll need to select one of the file transfer protocols; most comms packages allow you to do this, offering ASCII as the default for most transmission but also including options like XMODEM or KERMIT as protocols for sending and receiving non-text transmissions.

30. **If incoming text appears with wide spaces between lines**, the remote service is sending a Return code at the end of a line and your system has been set up to add a Return automatically as well — so you're getting two Returns per line. Your software should enable you to correct this; select 'Carriage Return from host' or whatever.

31. **If the incoming information appears on a single line which is constantly being overwritten**, the remote service is not sending a Return code at the end of a line (or you're not able to pick it up). Again, your comms package should let you specify something like 'no Carriage Returns coming from host'.

32. **If you suddenly lose contact midway through the transmission**, chances are that a burst of noise on the line has so corrupted the carrier tone that one or both ends of the link couldn't remain in communication. This could have been caused by weather, a crossed line, someone picking up an extension, or whatever. Your only option is to try again from scratch (but first check that you or the cat haven't inadvertently pulled out any of the cables). ◆

Free Small Ads

Apricots for sale

■■■ Xi20: £1,000. Also F2: £250. Contact Joyce Quarrie, 01-530 2145.

■■■ APRICOT PC — twin 720K drives, 12in green monitor, 768K RAM, Internal Modem, spare 256K RAM card, dust covers. All the usual software plus Ansible ASD 1 and 2 disks and ACOLYTE utilities disk, SuperCalc 2, COMMUNIQUE, PolyPrint and PolyPrint Plus, IANKEY Two fingers to Touch Typing course, Hitchhiker's Guide to the Galaxy, wide selection of public domain software. Careful home use and in perfect working order. Available from about end September, price £650. Call Mike Florey on 01-728 3215 (day) or 01-777 7534 (evening), or e-mail on 72:MAG90225.

■■■ APRICOT PORTABLE 256K complete with carry case, SuperWriter, SuperCalc, SuperPlanner and utilities: £250. Phone D Loebl, 01-642 5380.

■■■ APRICOT PC (XiFD). 768K RAM, 12in monitor, plus software. Offers? 0244 817292 (day), 0978 760387 (evening).

■■■ APRICOT Xi5 with single 3½in floppy, Super software, WorkMate (SideKick look-alike), Manager, WordStar. £750. 0329 46166.

■■■ 512KB PORTABLE. Very little used, with original software. Cash or exchange offers? C Sherwood, 021-233 2997 between 10am and 4pm.

Apricot extras

■■■ PHILIPS 14in green-screen monitor: £70. Phone D Loebl, 01-642 5380.

■■■ INTERNAL MODEM and software: £100. 0329 46166.

■■■ 256KB EXPANSION MEMORY CARD for PC/Xi: only £65. Also ADVANCED USER'S GUIDE TO THE APRICOT £15. Robin Griffiths, 01-493 6771 ext 28 (daytime) or 09323 48148 (evening).

■■■ SOFTWARE FOR SALE: LOTUS 1-2-3 ver 1a for Apricot PC/Xi; with original registration intact, hardly used — £160. And QED+ integrated package (diary/calendar, project planner, address book, telephone list/dialler), hardly used — £75. Both prices include postage. Phone Robin Griffiths, 01-493 6771 ext 28 (daytime) or 09323 48148 (evening).

Other sales

■■■ EPSON LQ1500 printer with sheet feeder and continuous feed. Contact Joyce Quarrie, 01-530 2145.

■■■ SIRIUS: old-style, twin-floppy machine. Cash or exchange offers? C Sherwood, 021-233 2997 between 10am and 4pm.

■■■ NUMEROUS SIRIUS AND V9000 machines and associated extras are offered in VICTOR SIRIUS DEALS FOR READERS — call 01-254 2245 for your FREE copy.

■■■ MULTI-COLOURED DISK LABELS, 18 per packet — just right for your bulk disks. Three packets £1. Robin Griffiths, 01-493 6771 ext 28 (daytime) or 09323 48148 (evening).

Wants

■■■ 5¼in EXTERNAL DISK DRIVE and controller card for Xi. Also MOUSE, COLOUR card, BOOKS, SOFTWARE (WordPerfect, Symphony, Windows, etc). Contact Bo Danielsbacka, Valkamakatu 13 A3, 48200 Kotka, Finland.

■■■ APRICOT INTERNAL MODEM wanted. Phone (France) 0101 336 2851079, or e-mail Telecom Gold 72:MAG20660, or write M Leonard, Hotel Roc San Miguel, Soldeu, Andorra.

■■■ GSX PROGRAMMERS MANUAL and USERS GUIDE desperately wanted! Also any secondhand Borland programming languages (especially PASCAL or BASIC) — must run on an F1. Phone Tim on 01-444 9265 after 4pm.

■■■ APRICOT PC with monitor and twin 720K drives wanted — would consider an Xi. Call James Gray, 01-839 6651 (day).

■■■ APRICOT PORTABLE wanted, 512KB. Also SOFTWARE for FP and PC. Phone 0423 521439 or write E Holgate, 18 Montpellier Parade, Harrogate, HG1 2TG.

■■■ APRICOT PC TECHNICAL REFERENCE MANUAL wanted. Phone Simon Protheroe, 01-836 5454 ext 2161 (office hours) or 01-263 4489 (after 6pm).

■■■ APRICOT PC or 10MB XI wanted; also PRINTER if available. Fair price paid, immediate cash. Phone Mrs Morrison, 0636 704208.

Misc

■■■ KITCHENS: I work for a kitchen manufacturer and am able to buy kitchens heavily discounted because the packaging has been damaged — contents are perfect. Call 0484 545807 after 6pm.

■■■ HOUSE TO LET in the South of France at Agde, on the Mediterranean coast. Attractively converted 16th Century house with roof terrace: sleeps 4-5. Local British management. Phone 01-352 5925 evenings.

TP Group Services

■■■ DEAD COMPUTER? We have a life support system that can tackle anything.

■■■ WOBBLY SCREEN? May be a failing power supply. We can replace and/or rejuvenate.

■■■ FILES ACCIDENTALLY DELETED? We can (almost certainly) recover anything that has been erased on hard disk or floppies — individual files, whole directories, even a complete disk that someone has inadvertently reformatted.

■■■ DEAD DISKS? Hard disk not working? System not booting? We can fix that too.

■■■ PRINTER PROBLEMS? Printer not working at all? Not printing as well as you'd like? Or not using all the facilities the printer handbook claims? We can make any printer work with any computer.

■■■ INCOMPATIBLE FILES? So you have a disk from one computer that contains data you want to use on another machine? We can take any disk in any format and copy the data on to a disk in any other format.

■■■ TP GROUP — THE PROBLEM SOLVERS. CALL US ON 01-254 2245.

A Turbo Pascal Primer

Part Eight: Data manipulation

By Mark Whitehorn

When computers were first invented, it was a belief widely held that they would be used almost exclusively for data manipulation. This has led the writers of programming manuals to describe at great length how a particular language can deal with data. That would have been more sensible if the earlier expectations had been realised, with computers being used in the main for number-crunching; what we have found instead is that computers have an enormous number of uses, only some of which involve large-scale manipulation of data.

Having said that, I do feel that now is the time to look at the way in which Pascal handles data. Even if you believe that you'll never have to handle large blocks of data, you may well find the ideas interesting.

Arrays

We start off with an immediate problem. (As a digression, it is worth noting that most of the extra bits that you find in computer languages are there to deal with awkward problems — which can often be solved without recourse to the extra bit, but not as easily.)

The problem is this. Suppose you wish to store lots of pieces of data which are all linked in some way; an example could be the marks achieved by students in an examination. You might well decide that since students can achieve a mark such as '120.8', this sort of data was of type *real*. With say 200 students, you'd have to make the following declaration:

```
Var
StudentMark1,
StudentMark2,
StudentMark3,
```

... and so on, up to ...

```
StudentMark199,
StudentMark200 : Real;
```

Boring isn't it? Then you have the problem of manipulating these data with a control structure. This construction looks as if it might find the sum of all the students' marks:

```
For Count := 1 to 200 Do
  Total := Total + StudentMark(Count);
```

Unfortunately, it doesn't; instead the compiler decides that it has never heard of the variable *StudentMark*, since it will not substitute the value in *Count* into the variable name *StudentMark(Count)*.

But wait: racing to the rescue comes a data type which you can define. It is called an *array*. In the variable declaration section of the program you declare an array like so:

```
Var
StudentMark : array[1..200] of Real;
```

This tells the compiler that you wish to declare two hundred *Real* variables, which will have the names *StudentMark[1]*, *StudentMark[2]* ... and so on up to *StudentMark[200]*.

There are a couple of things to note here. The array can be made up of any pre-defined variable type. So we could have ...

```
StudentAge : array[1..200] of Integer;
```

Or ...

```
StudentComment : array[1..200] of String[42];
```

Also note that the square brackets are essential; and there must always be a square bracket after the variable name when you use it in the program. So in the above example, we have *not* declared any variables called *StudentMark* — instead we have two hundred variables, all with names like *StudentMark[57]*.

For a start you'll notice that using an array makes it easy to declare lots of related variables very quickly. The real power of declaring variables like this, however, lies in the fact that you can substitute a variable for the number which is found in the square brackets.

Ah, *now* you see the advantage. You can write things like this:

```
Total := 0;
For Count := 1 to 200 Do
  Total := Total + StudentMark[Count];
```

And lo, *Total* will hold the sum of the students' marks.

How an array is declared

When we declare an array type in this way, we have to tell the compiler about two other types.

One is the *base* type, which is the type of the information which we wish to store. In the example we are storing the students' examination results, which are *Reals*; so the base type here is *Real*.

The second is the *index* type, which defines how many bits of information we are going to store. We had 200 students; students are not divisible into little pieces, so the index type is of type *Integer* not type *Real*.

LAPTIMES.PAS

```
Var
Count, Counter : Integer;
Temp           :Real;
LapTime        :Array[1..8] of Array[1..56] of Real;
{ This can actually be shortened to ...
LapTime        : Array[1..8,1..56] of Real; }
{-------------------------------------------------------}
Procedure FillArray;
{This fills the first row of the array with zeros}
Begin;
 For Count := 1 to 56 Do
   LapTime[1,Count] := 0;
 {This fills the remaining seven rows with zeros.
 There are other ways of doing this,
 but Turbo allows us to do it this way,
 so I thought I would demonstrate.  Basically
 we are saying, make the second vertical column
 of numbers the same as the first, then make the third
 the same as the second and so on until all the columns
 are full.}
 For Count := 1 to 7 Do
   LapTime[Count +1] := Laptime[Count];
 {The two bits above can be replaced by
 FillChar(LapTime,SizeOf(LapTime),0);
 Although useful, FillChar should not be used to
 fill your arrays with anything except 0,
 unless you know what you are doing.
 The SizeOf function is useful.
 It returns the size of the array, so that the
 procedure called FillChar knows how much
 memory to fill with 0.}
End; {Of Procedure FillArray}
{-------------------------------------------------------}
Procedure WriteArray;
{Write out the whole array, just to prove it
is all full of zeros. }
Begin;
 For Count := 1 to 8 Do
   For Counter := 1 to 56 Do
     Begin;
      Temp := LapTime[Count,Counter];
      Writeln(Count:5,Counter:5,Temp:8:2);
     End;
End; {Of Procedure WriteArray;}
{-------------------------------------------------------}
Procedure TakeInValues;
{Take in values for all of the lap times.
If you actually run this program I suggest that you
reduce the figures from 8 and 56.  If you don't,
you will have to enter 448 values!}
Begin;
 For Count := 1 to 8 Do
   For Counter := 1 to 56 Do
     Begin;
      ClrScr;
      Writeln('Car number...',Count);
      Writeln('Lap number...',Counter);
      Writeln('Please enter the lap time.');
      ReadLn(Temp);
      LapTime[Count,Counter] := Temp;
     End;
End; {Of Procedure TakeInValues}
{-------------------------------------------------------}
Procedure QueryArray;
{Allow the array to be queried.}
Var
Car,Lap: Integer;
Begin;
 ClrScr;
 Writeln('Which number of Car ?');
 Readln(Car);
 Writeln('Which lap ?');
 Readln(Lap);
 Temp := LapTime[Car,Lap];
 Writeln('Lap speed was ...',Temp:5:2);
End; {Of Procedure QueryArray}
{-------------------------------------------------------}
Begin;
 FillArray;
 WriteArray;
 TakeInValues;
 QueryArray;
End.
```

Mostly of course you will use type *Integer* as the index type, but that's not always the case. The index type must be both *simple* and *scalar* — criteria met by type *Integer*, of course, but also by other types like *char*.

A quick example

Let's have a look at another example. You have entered your racing Bentley in a 56-lap race and wish to store the lap times in an array. You decide on the name of the variable you will declare (say *LapTime*), the sort of data which you wish to store (*real* numbers), and how many you want to store (say 56). Then declare the array accordingly:

```
LapTime : array[1..56] of Real;
```

A simple loop like this ...

```
For Count := 1 to 56 do
 Readln(LapTime[Count]);
```

... will enable your pit crew to enter the time (in seconds) for each lap as the race progresses. At the end of the race it will be a simple matter to calculate all sorts of things — average lap time, total time, and so on.

More complex arrays

That's it, really. If you understand what's gone before you will be able to utilise arrays to their best advantage: so don't think they are too complex — arrays are ridiculously easy to use and a great boon to the programmer.

This may well be as far as you need to take your understanding of arrays, but I can't resist talking about some of the more devious uses (since they are so much fun!).

It may come as little surprise to you (or conversely it may be a terrible shock) to realise that the base type of an array can itself be an array. Hmm ... pause for thought. This means that you can do this kind of thing:

```
Var
LapTime : Array[1..8] of Array[1..56] of Real;
```

This sets up a two-dimensional array of *Reals*. If this begins to sound complicated, don't worry; let's examine what this terms 'a two-dimensional array of Reals' means.

Suppose you want to record lap times not only of your own car but also for the other seven participants (an exotic mixture of Gilberns, supercharged Spitfires and D-Types). If you were working on paper you might draw eight vertical columns, one for each car; then you might have 56 horizontal lines across the paper numbered 1 to 56 (one for each lap). Now you have an 8x56 grid with 448 boxes or cells. Into each of these you can enter a lap time, locating the correct box by reference to the car number along the top and the lap number down the side.

At the end of the race someone might want to see the time car no 5 took on lap 23: you can look that up in box 5, 23.

The computerised version is easier to use. Once the declaration above has been made, we've declared a total of 448 real variables. They all have names like *LapTime[5,23]* and *LapTime[7,45]* (but not *LapTime[45,3]*). We could write a procedure like *TakeInValues* in LAPTIMES.PAS to enter all of the data. And when asked about car 5 in lap 23, we could use a procedure such as *QueryArray*.

This is a two-dimensional array of reals. In fact arrays like this can have as many dimensions as you like, but bear in mind that they can eat up a lot of computer memory — an array of 40x40x40 *Char* takes up 64KB!

Arrays and user-defined types

Still more interesting — particularly for game freaks — is an array of a *user-defined* type (I told you that user-defined variables were useful). Say we're designing a game which allows the player to move around on a board of 40x30 squares. These squares are normally empty, but sometimes they contain food, drink, a penguin, or treasure.

We could first define a type called, say, *Contents*:

```
Type
Contents = (empty, sandwich, salmon, steak,
wine, water, penguin, diamond, gold, silver,
crown);
```

Then we could set up an array of the squares of the board:

```
Var
Location : Array[1..40] of Array[1..30] of
Contents;
```

Now, each square of the board can only contain one of our chosen objects. The second demonstration program illustrates this in a (hopefully) enlightening manner.

Last words

Finally, the last article left you with the problem of how to avoid a data type running outside declared limits. Procedures like this are useful:

```
If Penguin = Emperor Then
 NextPenguin := Little
Else
 NextPenguin := Succ(Penguin);
```

As an exercise for this month, try expanding GAMEDEMO.PAS into a useable game. Lots of possibilities spring to mind — the penguins could move around in the array, the program could be made to warn of the proximity of a penguin, they array could be have three (or more) dimensions ... The limit, as they say, is your own imagination; but be warned, game-building can be addictive. ◆

GAMEDEMO.PAS

```
{ This program sets up an array of locations and
scatters a few goodies and Penguins around. Then the
player is located in the array at random, and
allowed to move around looking for things. The
program tells the player where they are, and which
goody they have found. The program terminates if
they press 'q' or find a Penguin. }

{-----------------------------------------------}

Type
Contents  =  (empty, sandwich, salmon, steak, wine,
             water, Penguin, diamond, gold, silver,
             crown);
Var
Ch :Char;
Count, Counter : Integer;
Location : Array[1..40,1..30] of Contents;
PlayerX,PlayerY : Integer;

{-----------------------------------------------}

Procedure WriteCont(Var Temp :Contents);

{ A procedure to enable output from Type contents }

Begin;
  Case Temp of
    empty      : Writeln('empty');
    sandwich   : Writeln('sandwich');
    salmon     : Writeln('salmon');
    steak      : Writeln('steak');
    wine       : Writeln('bottle of wine');
    water      : Writeln('jug of water');
    Penguin    : Writeln('Penguin');
    diamond    : Writeln('diamond');
    gold       : Writeln('sack of gold');
    silver     : Writeln('silver');
    crown      : Writeln('crown');
  End; {Of Case}
End; {Of Procedure WriteCont}

{-----------------------------------------------}

Procedure FillArray;
Begin;

{ This fills the array with 'empty' }
{ Now that I think about it, filling an array with
empty seems like a philosophically difficult
concept. Let's just ignore it. }
  For Count := 1 to 40 Do
    For Counter := 1 to 30 Do
      Location[Count,Counter] := empty;

{ This puts about 20 jugs of water around, about 20
sacks of gold and exactly 10 Penguins. (Why is it
'about 20', rather that exactly? I'll give you a
clue. It would be more accurate to write 'this puts
20 or fewer jugs...') }

  For Count := 1 to 20 Do
    Location[Random(40),Random(30)] := water;
  For Count := 1 to 20 Do
    Location[Random(40),Random(30)] := gold;
  For Count := 1 to 10 Do
    Location[Random(40),Random(30)] := Penguin;

  { Locate the player at random }

  PlayerX := Random(40);
  PlayerY := Random(30);

End; {Of Procedure FillArray;}

{-----------------------------------------------}

Procedure InitialScreen;
Begin;
  ClrScr;
  Writeln('Use the W A S and Z keys to move
  around.');
  GotoXY(5,20);
  Writeln('Press "q" to finish.');
End; {Of Procedure WriteScreen}

{-----------------------------------------------}

Procedure MovePlayer;
Begin;
  Read(Kbd,Ch);
  Case Ch of
    'A','a' :PlayerX := PlayerX - 1;
    'S','s' :PlayerX := PlayerX + 1;
    'W','w' :PlayerY := PlayerY - 1;
    'Z','z' :PlayerY := PlayerY + 1;
 End;
  { Wraps the player around in the array, so that
  he/she can never escape }
  If PlayerX < 1 then PlayerX := 40;
  If PlayerX > 40 then PlayerX := 1;
  If PlayerY < 1 then PlayerY := 30;
  If PlayerY > 30 then PlayerY := 1;
End; {Of Procedure MovePlayer}

{-----------------------------------------------}

Procedure UpdateScreen;
Begin;
  GotoXY(5,5);ClrEol;
  Write('You are at location
  X = ',PlayerX,' Y = ',PlayerY);
  GotoXY(5,8);ClrEol;
  If Contents(Location[PlayerX,PlayerY]) = empty
  Then
     Writeln('Sorry, this location is empty') Else
     Begin;
       Write('This location contains a ');
       WriteCont(Location[PlayerX,PlayerY]);
     End;
End; {Of Procedure UpdateScreen}

{-----------------------------------------------}

Begin;
  FillArray;
  InitialScreen;
  Repeat
    MovePlayer;
    UpdateScreen;
  Until (Ch = 'q') or (Ch = 'Q') or
  (Contents(Location[PlayerX,PlayerY]) = Penguin);
End.
```

Reflection

In praise of Apricots

or why an Apricot programmer hates the IBM PC

By Sten Ternström

In May 1985, I bought an Apricot PC and was very pleased with it. I still am; it has a compact, appealing exterior, an interesting hardware architecture, and a quiet fan. It seemed to be a sure winner over you-know-who.

Well, we now know what became of it; a sorry fate indeed for such an excellent machine. It continues to give me many happy hours of programming and use and has never failed (true!). Only when I started to program in the Microsoft WINDOWS environment did my Apricot PC begin to feel like a slow machine for the task. Having recently acquired a fast AT-compatible, I find myself so frustrated with the stupidities of the IBM architecture that I felt obliged to compile this belated reminder of the many merits of the original Apricot PC.

What follows is purely a programmer's point of view; I do realise that people who want to *use* programs rather than *write* them may have quite a different opinion.

Programmers working with the IBM PC quickly found that the machine's performance was quite poor. This very soon created a dead-lock situation: you have to program directly to the IBM BIOS or even to the hardware in order to achieve acceptable speeds. As a result, most of the interesting programs on the market are very IBM-specific. And this fact imposes a dictum of downward-compatibility for future hardware, in that it 'must' be compatible with the old, mediocre machine.

This is a very sorry situation to be in, from the technical point of view. In my opinion, IBM's new PS/2 is also an ugly beast. It had to be compatible with MS-DOS, which wouldn't be too bad; but it also had to be hardware-compatible with the earlier machines. Apricot's early claims to IBM-compatibility were not unreasonable if you took the view that only 'naughty' programs access the hardware directly. If the IBM PC had been a better machine to begin with, this might not have become such a problem — and Apricot might have done better, too.

I think Apricot's designers were quite sensible: they created a good technical design for the original PC and didn't worry about making the subsequent models 100 per cent hardware-compatible. The F Series, the Portable and the Xen were all innovative machines in their own more or less successful way. The cross-machine compatibility was (eventually) provided by the generic BIOS 3.1, which in principle allows an Apricot application program to run on any model (even with graphics) provided that the programmer adheres to the rules.

There was also an IBM BIOS emulator, IBM.COM, which in my opinion provided as much compatibility as could reasonably be expected. The main obstacle here was of course the different addresses and layouts of the video memory. It is ironic that the Apricot PC failed largely because of its technical superiority and the concomitant lack of 100 per cent hardware-compatibility.

So let me get down to the nitty-gritty details of things that I prefer in the Apricot PC, lest they be buried and forgotten without due respects. I shall also mention some characteristics that admittedly did make the Apricot solution less than perfect.

CPU and bus architecture

The Apricot PC has the Intel 8086 CPU chip rather than the 8088 used in the IBM PC. They are identical except that 8086 uses a 16-bit bus, which means that the Apricot can read and write to memory in chunks of 16 bits rather than just 8. This should lead to some improvement in performance, although not by as much as a factor of two.

In addition to the 8086, the Apricot has a special I/O co-processor chip — the Intel 8089. This gives the potential of much higher throughput on I/O operations and memory transfers, since the 8089 is specialised for such things. In fact the 8089 has two separate channels which essentially can make it do the work of two concurrent I/O processors.

Admittedly, the 8089 is not easy to program, since it has its own instruction set and therefore needs a different assembler from the 8086. The BIOS writers used this chip for doing floppy disk I/O, but unfortunately it was never called on to do anything else (so far as I know). For example, the 8089 would probably have done nicely for speeding up the elaborate but admittedly sluggish ESCape sequence interpreter of the screen driver.

No other microcomputer that I know of has an 8089; its main use appears to have been in industrial controllers.

Memory map

The 8086/8088 chips support a memory address space of up to 1,024K bytes. Some of this space must inevitably be sacrificed to basic resources such as the operating system, the BIOS code and the video map.

One strange decision that the IBM designers made back in 1981 was to reserve all memory above 640K for system use — for video memory and 'future developments — thus making it useless for application programs. Apparently 640K was thought at the time to be more than enough for anyone. The reserved upper 384KB are however almost unused: on a standard IBM PC the video

memory in text mode occupies typically 4K bytes, or 32K in graphics mode, usually starting at 768K (B0000 hex). The IBM BIOS only starts at 960K (F0000 hex) and the rest is empty land, laid to waste.

By contrast, the Apricot uses a two-part video map with a 32KB pointer table starting at F0000 hex and the fonts and/or graphics bits themselves residing in about 48K bytes starting at 2000 hex. There is nothing to disrupt the contiguity of memory at A0000 or B0000. With a full complement of memory installed, and subtracting for the space required by Apricot's RAM BIOS, this gives the Apricot user about 190K bytes more to play in. Technically this is a much more attractive solution than IBM's — but it's not compatible at the hardware level.

RAM BIOS

Most of the Apricot's BIOS is in RAM, making it much easier to upgrade the BIOS. The downside to this is of course the confusion over the many BIOS releases, each with new fixes and new bugs. I still prefer that to a ROM chip which *never* gets revised.

Boot sectors

Apricot boot diskettes include a boot sector which contains information on the current system configuration — like how many disk drives there are, what kind of display is installed, and so on. This is so much better than the PC/AT's horrible scheme of storing configuration info in a little non-volatile RAM, which (a) actually *is* volatile now and then; and (b) requires you to run a SETUP program when the configuration is lost (when the battery fails, for example, or when another program runs amok).

Dual-ported video memory

One common nuisance when doing graphics in the IBM's video memory is that it is not dual-ported. This means that when the 6845 video controller (the same chip is used in the Apricot) is accessing the video memory, the CPU cannot write or read there without causing garbage on the screen. This is not true for all IBM video displays, but it does apply to many of them — notably the CGA. Only when the electron beam is switched off for a horizontal or vertical retrace can you do anything with the IBM's video memory.

In practice this means you have to waste a lot of time polling the 6845 for a retrace before you can squeeze in an access. And once you get it, there is only time for transferring a few bytes at a time.

The Apricot, on the other hand, has dual-ported video memory — special hardware which arbitrates memory accesses between the CPU and the video controller. This means that you can access the video memory at any time without causing hash and without undue delays. In view of this, and given the 16-bit wide bus, I find it hard to understand why the screen output of the Apricot isn't faster than it actually is.

Orthogonal text attributes

In text mode, both the Apricot and the IBM can enhance the text with attributes such as underline, reverse, and bright. The Apricot lacks the 'blinking text' attribute; but otherwise it is superior in that the four attributes are orthogonal — that is, they can be displayed in any combination. The IBM hardware refuses to do some combinations, such as bright reversed, which is quite annoying when you try to produce rich displays.

The Apricot allows both underline and strikethrough, and the font designer can specify in which pixel row these should appear. Strikethrough is normally used in legal documents, but I am more scientifically inclined and prefer to have an overline attribute. On my Apricot, I can (though printers, of course, need extra cajoling to do overlining).

Multiple fonts in text mode

The Apricot also implements a clever scheme for supporting multiple fonts or character sets. In text mode, the hardware can display up to five different fonts simultaneously. The FONTEDIT utility is a bit cumbersome but perfectly usable.

IBM's poverty in this regard is further aggravated by the brain-damaged way that 'foreign' characters are allocated to the so-called 'extended ASCII' character set. Version 3.3 of PC-DOS tries to fix this using 'code page switching' which seems to be a sorry kludge.

Unfortunately the usage of multiple fonts on the Apricot was rather poorly supported in the utility software. The ESCape sequence handler could only do three fonts, and even that was poorly documented; the FONT utility could change only the system font. Only by poring over the PC's *Technical Reference Manual* at length did I learn how to display five fonts; and now I can, in my custom editor.

High resolution

IBM's 1988 'Apricot clone,' the PS/2 Model 30, still hasn't reached the 800x400 pixel screen resolution of the 1983 Apricot PC.

Disk drives

Apricot's use of 3½in diskettes was innovative and superior in all respects to the 5¼in disk — faster, smaller, more reliable, and providing more storage (at least with 720K drives). At long last IBM has caught up on this point. *(Well, almost; Apricot drives spin at 600rpm, twice as fast as IBM's — Ed.)*

Keyboard

Keyboards are much a question of taste; I really like Apricot's, MicroScreen and all (yes, I know, except for the membrane keys; but mine still work).

I suppose I have come to miss numbered function keys. But the great thing is that the Apricot lets you change what you do not like: you can have the CONTROL key in the middle of the board, if you must. For us Scandinavians, who really need to have umlauts and other exotic typography on ordinary keys, this is invaluable. The Apricot keyboard is mercifully 'soft' and the keyboard tables are not too hard to understand. The IBM's keyboard BIOS is a real mess by comparison.

The MicroScreen is cute but doesn't really come into its own without the RINSTALL utility from DEALS FOR READERS disk AP028 *(or* MS.EXE, *on AP027 and others — Ed)*. I often use the calculator, especially for intermediate sums when doing spreadsheet accounting. And there are dozens of other nice applications for the MicroScreen; for example, I wrote a little utility called SHOW8089 which flashes the membrane LEDs whenever the 8089 chip (see above) is actually doing something. Thrilling, what?

Mice, too, are rather personal, and again I like Apricot's trackerball which doesn't need any desk space at all — you can even hold it in your lap when there are papers and coffee mugs everywhere. More important, it doesn't need a serial port or an expansion slot, since the connector is already there on the back of the keyboard.

And after a few years, we are now seeing the clock/calendar batteries becoming exhausted. The Apricot owner buys a new battery, while the IBM owner must haul the machine to a workshop and have them solder a new one in ...

Expansion slots

I have now been infuriated with IBM clone add-on cards that *almost* fit the connector *and* the housing so many times that the Apricot expansion slots seem like heaven — real connectors that don't wear quickly, and small dimensions, which to me has an intrinsic appeal. Small is beautiful.

Another point is that the Apricot manual told third-party board vendors which port numbers and which interrupt to use, which should have reduced confusion. I am not sure how many add-on boards ever actually appeared — might APRICOT FILE make a list?

Interfaces

IBM ATs have a parallel printer port that looks very like an RS-232 connector. The Apricot's printer port has a Centronics connector, just like the one every parallel printer has. This feels right and eliminates connector confusion.

On the Apricot it is also possible to use the parallel port for *input* if you are prepared to do some low-level programming. This might be useful if you need two-way high-speed communication with some peripheral device. On the IBM, this facility has been (literally) short-circuited.

Apricot's BIOS implements a 2KB background cache and a number of useful calls for the parallel port. My favourite is 'Transmit string' which is very handy for bit map dumps and other voluminous output. MS-DOS can only print one character at a time and adds so much overhead to the printer output that the effective baud rate is not much higher than for a serial port. With the Apricot BIOS, you can really flush those bytes out at good speed, so that my matrix printer (a NEC P6) is kept busy. This facility is sorely lacking in IBM's BIOS.

The Apricot's RS-232 connector is female, which is less vulnerable to accidents and also feels more correct. The Apricot BIOS support for the serial port is very comprehensive; the IBM BIOS has no serial port support to speak of.

Documentation

APRICOT FILE people are going to hate me for this, but the *Technical Reference Manual* for the PC/Xi is really quite good. So is the *Technical Reference* on BIOS 3.1, although it was very late in appearing. *(Not so — we agree, though the former is a bit out of date — Ed.)*

Aesthetics

It still amazes me that a machine as unwieldy and ugly as the IBM PC can have sold in such numbers. Appearance was certainly an important consideration when I chose an Apricot.

Windows

When Microsoft WINDOWS arrived, it held the promise of survival for the Apricots. Apricot plc never seemed to realise this, or perhaps the real WINDOWS applications were too late in coming. Among many other things, WINDOWS really does provide hardware independence. For a programmer it is quite a thrill being able to run your fancy graphics program on an Apricot, then copy the .EXE file to a PC and run exactly the same program on a Hercules screen or an EGA display — WINDOWS takes care of the differences in hardware.

If there is anything I would beg of Apricot now, it would be to continue supporting upgrades of WINDOWS for the old PC and Xen machines. All it should take is rewriting a number of device drivers.

Last words

To those of you who still love your Apricot PC, let's all be thankful for TP Group's publishing efforts, and let's hang in there writing clever new programs. Without continued support for MS WINDOWS I think Turbo Pascal is the best way to go. Get yourself a copy of the official Apricot BIOS 3.1 *Technical Reference Manual* and the *Hardware Technical Reference*, and you will see that there is still a lot to be wrung from your machine. ◆

Readers' Writes

Your chance to have a say

BIOS 3.1.2 screendumping

In your *Q&A Clinic* of Release 3.8, you refer to the patched BIOS 3.1.2 ejecting the page after hardcopy is done in a screen dump. That's not the case with my version from AP026; I have had to patch it myself, using the location you identified *and* also location 1083 hex. There the following appears:

```
xxxx:1083  B00C  MOV AL,0C
xxxx:1085  E668  OUT 68,AL
```

The alteration required is to change 1083:

```
xxxx:1083  B000  MOV AL,00
```

Andrew Glennie
Christchurch, New Zealand

Thanks for pointing this out. Both patches are now incorporated on AP026.

Laser printing revisited

Steve Allen (APRICOT FILE *Reader's Writes*, Release 3.10) is right in some respects: the Epson emulation in the Brother HL8 (the same one as chosen by Apricot as its PCL laser offering) makes it highly suitable for use as a high-speed correspondence-quality printer. I'd also agree that anyone using multiple fonts etc in correspondence is a mug: most people (including me) haven't got the design flair to carry it off.

There is, however, a particular situation where you need those all those font styles and sizes in correspondence; and that's where an office supplies the secretarial support for a family of companies. The secretary can load a pre-designed letterhead for the appropriate company on whatever letter s/he has typed, and need not worry about what is in the printer's tray — it's always plain paper.

Having the multi-font facility in the Brother costs as much extra over the standard HP Laserjet II as a font cartridge, and gives you about as much added functionality. Of course, you still have two slots free. Two slots seems a lot (two slots on an Apricot seemed a lot, anyhow!) but I'm sure that they're soon filled. The Brother isn't cheap, certainly not in the sub-£1,200 minus — let alone the under-£1,000 area that appeared to be dawning this year.

Of course when you configure the Brother as an Epson, that means bit-image graphics characters in enormous variety. The problem with them is that they're jagged. For many purposes this doesn't matter, because they aren't *very* jagged — they come in about as good as the large headings on Apricot File articles, which have an effective resolution of about 75 dots per inch. I'd have been happy with that if it had been easy to obtain on a standard HP or Canon printer; but in my searches for a package, I came across Ventura Publisher — and that showed me what 150 dpi could do, let alone 300. Not only is 300 dpi so good that you have to look very carefully indeed to see the dots, but a 1,200 dpi typesetter gives you no improvement at all if your method of reproduction thereafter is photocopying on bond paper!

I never use the laser for correspondence. Polytechnic correspondence I don't type myself, and for private things I use my trusty 12 cpi (note *c* not *d*) typewriter-with-an-interface. What I need are extremely high-quality documents for reports, and text-based masters for slides or overhead projection transparencies. These both demand large character sizes. In the former case I don't want jagged characters and Ventura is the ideal print formatter. I don't even need WYSIWYG (though Ventura gives it) — just a means of knowing where the page breaks are. The paragraph tagging function is ideal, since I can tag headings and suchlike while still in WordStar.

I've learnt a lesson, too — don't mix text and figures, but paste up the graphics afterwards. That can save you nearly a day a page!

For the OHP masters, the 24pt size maximum in my version of Ventura is just too small: 48pt would be nearer the mark. I don't care about a degree of jaggedness in this, so I've tried GEM DRAW and PAINT, WINDOWS PAINT and Micrographx IN*A*VISION. Actually, WINDOWS WRITE would do on a Brother that's treated as an Epson; but then that printer's so much more expensive. Sadly, the GEM products aren't so easy to use as the WINDOWS ones. Equally sadly, the graphics drivers in WINDOWS won't print PAINT images properly unless you have extra RAM. It's an equally sad story with everything you touch.

It seems that there is a threshold at about £2,800 where everything becomes easy. You can spend that money on upgrading your HP-compatible or buy a POSTSCRIPT laser. POSTSCRIPT has to be the better buy, because you get a package that doesn't need disk space on your Apricot.

Eddie Bromhead
Camberley

Back-patting

This is to advise you that I will be retiring next Monday, and to express my appreciation of the very pleasant and friendly association with yourselves over the past three years or so. I am not quite sure how I am going to survive without APRICOT FILE, a truly excellent publication; the service from you has always been courteous and most helpful ...

John Hermiston
London

Notebook

Putting a bold face on things:

Using printer Escape codes — a guide for beginners

By Eddie Bromhead

Before I get a heap of criticism for writing something which everybody knows, let me say that I have come across dozens of highly intelligent individuals who use Apricots and other PCs every day without having a clue what ESCape codes are all about ...

Background: bits, bytes and ASCII

Data is stored and manipulated in your Apricot in the form of bytes. A byte is a sort of code, made up of eight 'bits'; and they allow enough unique bit combinations to represent 256 different codes.

We could number these from 0 to 255 inclusive; or hexadecimal notation could be used, in which the digits 0 to 9 are supplemented by letters A to F — so the code range in hex is from 00 to FF. The benefit of hexadecimal notation is that the byte range is represented by just two characters, although a lower-case 'h' is often put after the code to denote hex notation — EFh, for instance. *(At* APRICOT FILE *we also put hex numbers in a 'small capitals' typeface to distinguish them from decimal numbers — Ed.)*

You will often see the character codes referred to as 'ASCII codes'; ASCII is the American Standard Code for Information Interchange, and it defines the meanings and order of the first 128 byte codes — those with values from 0 to 127 (decimal), 00 to 7F (hex). For example, ASCII code 69 decimal, which is 45 in hexadecimal, is an upper-case 'E'.

Control codes

The first sixteen of the ASCII-code characters are reserved for 'control codes'. They instruct devices like the printer to do various things, and won't normally cause any actual printing. From there on up to code number 127, each byte represents a single printable character: numerals, upper and lower case letters, and punctuation. Bytes from 128 to 255 may be anything, or nothing: more on that later.

Some of the control codes are reserved for communications purposes, but others are relevant to printing. Control code operations vary from the entirely obvious to the relatively obscure. Take for example one of the non-printing characters: CR or 'carriage return'. (It was named in days of yore, when printers had a typewriter-like 'carriage' and a fixed-position printhead; on the average modern printer this would more likely be 'head return'). CR is number 13 in the ASCII code set.

On a dumb printer, sending CR between lines of printable characters and having it obeyed explicitly would just lead to consecutive lines overprinting each other. CR normally has to be part of a pair, CR plus LF ('line feed' — move the paper up one line). Of course today's smart printers can be set to perform CR-LF on receipt of CR, or CR-LF, or LF. This is probably set by flicking a switch in the innards of the printer.

In its own-format files, SuperWriter likes to use CR on its own (just to save one byte per line in the file — the byte that would otherwise be taken by LF). WordStar uses LF as a 'soft' end-of-line indicator and reserves CR as end-of-paragraph.

Not all printers use all of the control codes, though. And one at least is guaranteed to confuse — FF (form feed) causes a printer to throw a page, and it is *not* the same as FF hex!

Possibly the most useful are SI and SO (shift in and out, ASCII code numbers 14 and 15 — or would you prefer 0Eh and 0Fh?). Many dot matrix printers utilise these to select and deselect double-width printing, some use them for compressed print.

All these are conventions typically used by software packages; there are bound to be others.

Escape sequences

Modern intelligent printers have more than sixteen software-selectable functions; as a result, they need to be sent more than the sixteen control codes in the standard ASCII set.

In order to generate additional codes, a *de facto* standard has been adopted — the code for ESCAPE, 27 decimal or 1Bh, is followed by one or more bytes: and taken together, this collection forms an extra command. Most of the resulting 'Escape sequences' are formed by the addition of a single extra character. A few take two, three or more extra bytes; HP printers sometimes require *five*.

This idea of ESCape sequences has been carried from printers to screens, so that various display operations are generated by using the appropriate codes. The whole business of screen ESCape sequences was ably dealt with by David Langford in APRICOT FILE Releases 2.9 to 2.11 (when are you doing ANSI sequences David?).

Printer ESCapes, however, are another matter altogether. The problem is that there is no standard for them: each printer is different. Well, not entirely — all Epsons support a huge range of

common ESCape sequences. But try them on other printers and you come unstuck.

ESCape sequences can be grouped into five classes:

1. Simple use of the sixteen standard ASCII control codes
2. Selection of choices that can also be set by DIP switches on the printer
3. Choice of character set, character width, and the like
4. An equivalent of cursor control for moving about on the page by more (or less) than a character or line at a time
5. Bit-image graphics — dot-by-dot printing

In this note I don't intend to say much about bit-image graphics: why should I anyway, when we've got GSX? In the remaining categories, number 4 is a mite smart-alecky and will possibly only be required by programmers.

For everyone else, an intelligent use of DIP switches plus four or five ESCape sequences at most from groups 2 and 3 should fill most needs.

DIP switches

DIP switches aren't convenient. In most printers they're inaccessible, and the printer may need to be partly dismantled to get at them. Once set, the settings are easily forgotten; at least no-one is going to change them in a hurry (but don't rely on this if there is common access to the printer).

It varies from printer to printer, but the kind of things you can set via DIP switches include these:

- **communications protocols** for serial RS-232 printers — the sort of thing you use an Apricot ladder program or ACTIVITY or GEMTOOLS or WCONFIG or whatever to change (Later, IBM compatible, Apricots have the wonderful MODE command). Fortunately, most printers these days are parallel (Centronics).
- **national character set** to give UK users a Sterling sign instead of a dollar or sometimes a hash symbol. If you set it wrong, you may find other odd characters being printed from time to time.
- **'End of line'** — whether CR alone, LF alone, or CR-LF produce a printhead return and line feed.
- **Mapping of characters 128 to 255** — sometimes called **7- or 8-bit data** selection. Select 7-bit data, and you can happily send WordStar files to your printer: the 'attribute' that WordStar appends to each character byte is ignored, and the byte is treated as the equivalent character in the lower half of the ASCII set. Other choices include whether to treat the upper half 'control codes' (the sixteen codes from 256 onwards) as ordinary control codes, or to use them as block graphics characters (useful for box-drawing).

Parameters with ESC

The first family of real ESCape sequences deal with the selection of character pitch and font. Typically these ESCapes take ESC and one character, often a single capital letter: Epsons, for example, switch into 10 cpi after receiving ESC E. But be wary — there's no standard in these things. ESC E makes a C Itoh go into 12 cpi, and we all know what ESC E does to an Apricot screen *(for those who don't know,* ESC E *clears the display — Ed.)*

Similar simple ESCape codes control whether paper moves forward or backward under line feeds, turn on boldfacing and underlining (and off again), and suchlike.

Setting the height of a line for paper control during line feeds can sometimes be done with quite high accuracy. This demands sending an ESCape sequence with a 'lead-in' code and then some 'data'. Printers differ in how they expect the data. A C Itoh, for instance, sets the line spacing in increments of 1/144ths of an inch with ESC T followed by three 'data' characters. These are plain text numerals; for example, ESC T followed by ASCII codes for '024' sets line spacing to 24/144th or 1/6 inch. Decimal numbers start at ASCII code 30, so the full sequence would be ...

```
ESC T 30, 32, 34
```

(In fact there are simpler sequences for 1/6th and 1/10th inch intervals, which are 'standards' on C Itoh printers.)

Epsons are typical of the other end of the spectrum; they have several ways of sending data. The first is to use plain text numerals. The second involves sending a number encoded in a byte: for example to send '32', you sends the digit '2' — which is byte number 32. Epsons have different lead-in sequences in some cases to tell the printer what to expect; in others, the range of numeric options are so small that the interpretation is obvious to the printer.

Some ESCape sequences may require large numeric values as parameters — for instance, when bit image data is being specified. In that case the date bytes are encoded so that the first one is 'the number of 256s' and the second is the number of 'units'. That scheme is used for things that potentially have more than 256 values. 'Bit image data' is used to drive the individually pins in a dot-matrix printer's printhead; that's how printers generate screen dumps and other illustrations.

Using Escapes — setting up

One of the simplest ways to use ESCape sequences is to send whatever you want to the printer to set it up initially, and thereafter not to use ESCapes, but treat the printer as a teletype or electronic typewriter. I use EPSON.COM from DEALS FOR READERS disk AP002; this utility can be invoked with a number of parameters to summon Epson-specific functions, or it can be given a string of hex codes to send to the printer (in which case it can be used with *any* printer).

Always invoke EPSON from a batch file; then you don't need to remember the command sequence.

Using Escapes from a program

This is the way to get the most out of your printer, but it results in the most programming problems. Programs are simply not portable if they support a multiplicity of functions. If in doubt, stick to the Epson codes — at least their printers are common, or commonly emulated.

All sorts of fancy ruling characters and text effects can be used. My experience though is that you should use them only for data which appears in the same way on each page, as anything which is variable in length causes problems!

The exception to this is for people who are writing programs for generating forms on plain paper. It's never as fast as using preprinted stationery; but at least there isn't going to be a page registration problem — and plain paper is usually significantly cheaper.

Using Escapes from WP software

When you call up a facility like boldfacing in SuperWriter or WordStar, codes are put into your text file. These are interpreted by the print formatter element of the software when you call it up; and for that to work correctly, you need to 'install' the printer — to select the particular printer from a list of choices offered by the word processor's installation program.

In SuperWriter there is plenty of choice; but many of the printers are obscure and uncommon, and often they aren't installed very well. With WordStar you can custom install some options to get better fine control than in SuperWriter — but at the expense of much effort.

Whatever happens, you should get boldfacing and underlining correctly at the very least. Installation menus for Epson printers normally handle italics correctly. The same applies to sub- and superscripts (by rolling the carriage up and down, or by using the special Epson sub- and superscript characters).

Sadly the earlier cheap Epson NLQ printers don't do NLQ italics or sub/superscripts; and Superwriter's installation details don't understand the range of facilities offered on modern printers.

From SuperWriter you can also insert specific printer-control ESCape sequences in the text file by means of the \OUT command. One of the useful options to so insert is '08' (BS or backspace) to get the next character to overprint the previous one. The same principle can call up whatever effect you want.

The final option is a 'printer interception' program which catches certain characters, or character sequences, and substitutes something else. Ansible has a sophisticated example (APP, on Ansible's ACOLYTE disk — DEALS FOR READERS number AS035); and David Langford also seems to be the author of DONTINIT.COM, which intercepts the ESC @ sequences used by many commercial programs to return all the printer settings to their defaults — ruining any set-up you have done.

The upper end of ASCII

ASCII codes in the range 128 to 255 are generally used in one of three ways:

- duplicating the characters from 0 to 127. This may not seem useful; but WordStar works by setting an extra bit in the first character of every word, so that it knows how to adjust word spacing in that line at a later date. If your printer cares not whether the extra bit is set, copying a WordStar document file to the printer will still generate a readable printout.
- duplicating the 0-to-127 character set with a subtle variation — like the characters being in italic, for instance (Epson).
- additional characters, like Greek letters and symbols, might be put in this area (as with C Itoh).

Problems

Difficulties arise because printer manuals tend to be written for experts, not novices. *(Actually, most of ours seem to be written by particularly illiterate chimpanzees for someone on a different planet — Ed.)* Modern Epson printer manuals are among the best — the FX800/1000 handbook, for one, is a model of clarity. Sometimes a printer has facilities that you only find out about by accident or by *extremely* thorough reading; most manuals certainly don't help the browser. Nevertheless, there's no substitute for reading the manual, however bad it may be.

The second common type of problem arises because your DIP switch selection conflicts with the software you are trying to run. Once again, you'll have to read the manual. For example, one printer won't accept bit-image data because it's set to

accept only 7-bit data; another printer is smart enough to know that bit-image data is *always* in eight bits, and to accept it whatever the DIP-switch setting.

Last words

Finally, there's ignorance. "I didn't know it could do *that*" is a common cry. Let's run through the sort of things even the cheapest dot matrix printer can do these days:

- print at *different character pitches* — 10, 12, and 16 or 17 characters per inch are usual
- print *double-width* characters
- print in a *near-letter-quality* mode and/or with *proportional spacing*
- handle *double-strike* and *underlining* automatically
- *move the paper forward and/or backwards* in tiny increments, as little as 1/144 or 1/210th of an inch, and print *superscripts* and *subscripts* — often as reduced-size characters
- have completely programmable *tab settings*
- print *unidirectionally* or *bidirectionally*
- offer one or more *alternative fonts* — typically italics or Greek letters and mathematical symbols
- provide *block graphics* and special *ruling characters* for generating forms and borders
- have a *bit-image graphics capability* better than half the 300 dpi resolution of a laser printer.

Spend a little more and you can have alternative fonts in serif or sans serif styles, downloadable character sets, full letter-quality, double-height characters, fonts on plug-in cartridges, and yet more bells and whistles.

Even daisywheel printers use ESCape sequences. My interface-equipped 12 cpi electronic typewriter can be sent ESCape sequences to change character pitch, line spacing, and do half-line subscript and superscript rolls. More complex daisywheels have other functions accessible in this way, notably proportional spacing.

Virtually every single one of these facilities will be obtained by ESCape sequences. If all you do is know that they exist and what they do, even if you never need to call them up explicitly, you will find your printer easier to use. ◆

Contributors in this issue

Dennis Jarrett is Editor of the newsletter. His magazine credits include being founder-editor of *16 Bit Computing*, *Office Automation Report*, *Practical Computing* and *Which Computer*. Among his books are *The IBM PC Book*, *The Sirius/Victor 9000 Book*, *The Electronic Office*, and the brand-new *Fast Track: Information Technology for the Fast-Track Executive* (£17.95 inc P&P from DEALS FOR READERS).

Edward Bromhead is Reader in Geotechnical Engineering at Kingston Poly, where he uses a couple of Apricots to model the behaviour of soil slopes and groundwater (mainly using his own Fortran software and GSX). He specialises in earthworks (eg for motorways) and landslides (he predicted the collapse of Carsington dam in November 1983: it fell over eight months later).

Roger Gann works full-time running the DEALS FOR READERS section of the newsletter as well as contributing words, ideas, humour, hints, legal advice, spittle for mass-mailings, etc. He was formerly the principal writer on 16 BIT COMPUTING; before that he was a builder (and a pioneer Sirius user), a lawyer, and a kind of grandiose bank clerk.

David Langford is a full-time author and computer addict, whose latest book *EARTHDOOM!* — a spoof of disaster novels, written with John Grant and scorned by Kingsley Amis — is "by no means" about Apricot BIOS variants. (The inclusion of a permanently drunk character called Apricot should not be misinterpreted.) He is inordinately proud of being a guest of honour at the 45th World SF Convention last August.

Simon Potter is a business systems consultant with a speciality in word processors, file managers and spreadsheets. He says he'd like to like computers but can't find one that likes him enough to warrant that.

Sten Ternström works and plays in music acoustics research in a computer-intensive environment at the Royal Institute of Technology in Stockholm. He bought his Apricot PC last Spring, likes it, and pig-headedly seeks to expand the system while stocks last.

Mark Whitehorn started life as a Cytogenicist but the Siren Song of the computer was too strong and he is now a private consultant and Advisor of Microcomputers to a Scottish university. He says his hobbies include penguins, building 1960s racing cars on the cheap, and indoor underwater powerboat-snooker (of which only the last is apparently untrue).

News and views in and around the Apricot world

Computing at work

A US publication called the *Jobs Rated Almanac* compares **occupations** in terms of salary, stress, work environment, outlook, security, and physical demands. According to the *Almanac* the worst job titles on that mix of criteria are migrant farmworker, fisherman, seaman, roofer, construction worker, cowboy, dairy farmer, nurse's aid, binman, and taxi driver. At the other end, the very best job is 'actuary'; the computer business does pretty well, however, with systems analysts ranked at no.7.

Then comes the *1988 Businessman Survey*, the ninth in the annual series from **Research Services**. It covers two thousand businesspeople (men and women, in fact) and among the 'lifestyle' data suggests that computing is on the increase — over 80 per cent of the sample have access to a computer at work, compared with 74 per cent a year ago. By comparison with their male counterparts, it seems a larger proportion of women are using computers regularly at work.

Or are they? Coincidentally, the annual *IT Review* from **Price Waterhouse** confirms that more and more micros are on the desks of top management — but they aren't being switched on. "If they are not slick at using a PC they would rather avoid it as a time-consuming machine which could undermine their confidence and ability in front of staff". What IT revolution?

And the on-line newsletter *Newsbytes* reports that using computers probably won't help you much in your job. "Productivity growth in the US economy has collapsed in recent years, and the size of the collapse has the dimensions of a crisis" concludes a report from the prestigious **Brookings Institution**. And "there has been little sign of the payoff to productivity among the industries buying computer equipment". At a press conference one of the report's authors opined that many corporate managers lack computer skills and have failed to take advantage of what computers can do.

Also recently published by **Manpower** and **Lotus Development** is a survey of top secretaries. *Secretaries and IT — which way for the future?* suggests that more than half the secretaries surveyed spend at least 50 per cent on their time on a desktop computer: that in general skills have been added but only a minority have won higher pay as a result: and that substantial time-savings are identified by PC users — 59 per cent of respondents put the figure as at least an hour, which the survey's authors translate into "several hundred million pounds a year" across the whole secretarial workforce. Not surprisingly, word processing is far and away the most common use (over 90 per cent) with the next application — databases — scoring less than 50 per cent. But the respondents did use an average of 2.6 applications apiece, so WP isn't the only candidate.

Computers for accountants

Accountancy has always been one of the professions most likely to benefit from computerisation — all those ongoing client accounts, the mass of words that have to be generated, the range of client services (ledger-keeping, bureau payroll services and the like) that are typically offered. A survey by **Tudor Business Systems** (which clearly has an axe to grind, in that it sells primarily to accountants) suggests that the last five years has seen an explosion in computer awareness and use among accounting firms with five or more partners — Tudor's figures compare with a 1983 survey by *Certified Accountant* magazine. **Client services** and **accounts production** both feature in 90 per cent of respondents; five years ago they were at 47 and 73 per cent respectively. **Spreadsheets** and **WP** were both 87 per cent; and perhaps the important one is **practice management**, cited by 77 per cent of respondents. Over half the survey were using MS-DOS, though a quarter had UNIX or XENIX.

US computing makes money

The latest *Datamation Top 100* list from the US magazine suggests that the computer business in the States is in good shape. The 100 largest DP suppliers there clocked an **18.7 per cent** increase in **turnover** (to a total of nearly $209 billion); and **profits** rose **27 per cent** too, with only six of the top 100 losing money — compared with a dozen one year ago (the losers this time around were Data General, Convergent Technologies, Zenith, Gould, Xidex and Dataproducts).

The gains in earnings varied widely, though, with **IBM** and **Unisys** (in first and second places in the turnover table) managing increases of only 5.9 and 6.8 per cent respectively. British-owned firms are of course conspicuous by their limited appearance in the tables; STC, Atlantic, Racal, Ferranti and Amstrad do however feature, with **Amstrad** doing particularly well — 75th place thanks to a 74 per cent improvement in turnover.

More eye evidence

The University of California's School of Optometry has the first eye clinic in the States (possibly in the world) that is devoted to the study and treatment of vision problems related to computer display technology. It looks as though there may be a definite need for the clinic; a study of 153 patients suggested an above-average number of problems in focusing, eye fatigue and other vision problems, he said. All the patients had worked for an average of six hours a day on a VDU over more than four years. Interestingly, the clinic suggests that VDU work could be "changing the physiology of the eye".

Dowty gets CASE ...

In a minor skirmish in the modem market, the Canadian firm **Gandalf** has been beaten to acquisition of Britain's **CASE** by engineering giant **Dowty**, whose comms arm is one of the UK stars in the field. Dowty's bid valued CASE at **£82.5 million**, Gandalf stuck at £70 million. CASE, which is best-known for high-speed modems and multiplexors, has been losing money in the States for a couple of years; Dowty has one successful operation there already in Datatel and should be able to rationalise things.

... Miracle equals Miracom

Once **Miracle Technology** was a pioneer in producing small, cheap, working modems for everyman (and woman). That's all changing; over the past two years the company has decided its future — and the big bucks — lie upmarket. In truth, profit margins are being squeezed at the V21/V23 end of things, with smaller, leaner and keener modem-makers nibbling into the market. So Miracle has become **Miracom**, is describing itself as a system supplier to the corporate market, and is taking on the high-end modems from **US Robotics** as the next steps towards "multi-user solutions capable of utilising mainframe facilities".

Miracom has just obtained BABT approval for its interesting rebadging of US Robotics' **Courier HST** modem (there are some software mods for the UK phone system). At **£995** it's not cheap; but you do get a proprietary 9600bps system with a modified high-speed version of MNP level 5 error-correction and data compression to give a claimed throughput of 17,200bps.

Death of the mini?

A couple of weeks back IBM launched its AS/400, basically a family of go-faster up-market minis, and described the minicomputer market as "the fastest-growing segment" of the industry. But about the same time a report from consultants **Butler Cox** was published on *The Future of the Personal Workstation*; and this anticipates that within five years the minicomputer will disappear in medium-to-large installations, leaving just mainframes and interlinked workstations. Lack of standards, a confusion of products and "persistent problems" with communications and interworking are identified as current difficulties; 'ease of use' was nominated as the major concern among organisations surveyed. IBM's response to all this can be imagined; and DEC, the leader in the mini market, poured scorn on the report. Maybe it's all a question of semantics — the distinctions between 'supermicros', small mainframes and 'traditional' minicomputers are increasingly becoming blurred; perhaps a minicomputer is merely any computer from a company that regards itself as making minis ...

DDL dies

Well, we keep recommending a flutter on **MBS** shares apparently in the face of all the odds: the latest news suggests that it's a medium-term buy rather than a quick way to pay for your next DEALS FOR READERS order, though. MBS has killed off **DDL**, its IBM-based PC distributorship — an operation that contributed over **£40 million** to last year's group accounts by selling to dealers, and which was forecast to provide a quarter of this year's group turnover. DDL was IBM's biggest European distributor.

The official MBS press release said IBM's policy of appointing new distributors was overcrowding the market and also stated somewhat baldly that DDL had to go because gross margins from distribution are only a third of those it is earning from sales to the corporate marketplace. Competitors have been quoted as saying that DDL simply wasn't efficient enough, IBM is saying that MBS "misunderstood" the role of a distributor.

Anyhow, MBS is now saying it can realise some £15 million from the divestiture for the purposes of acquisition. But the City wants real profit improvements right now, not investment for a golden future: after all, MBS has raised a total of £48.6 million from the City in the past three years, including a £25 million rights issue only a few months ago presumably to pump into DDL. City analysts are saying that's a lot of money to fund a reduction in business — and the closure has left the group with an overdraft variously estimated at between £4 and £8 million.

Some of the more knowledgable observers feel that MBS has been pretty courageous in refusing to chase turnover at the expense of profit. The alternative view is that MBS has cocked up its dealings with the dealer base, that cutthroat margins in the distribution game mean it can't really survive on direct sales alone, and that things will get worse before they get better.

Our view? Stick with it. The short-term emergencies may have to be met by realising some of the cash tied up in MBS operations — which means selling off some the more peripheral activities. And a few top-management changes would seem to be on the cards. But a smaller MBS that majors on corporate-level systems should be well-placed for 1989 ... if the group can last that long.

Other distributor news

There's much ado in the distribution business all round just at the moment. Eager Epson-compatible manufacturer **Citizen** has dropped one distributor (**CPU Peripherals**) and lost another (**Northamber** — who resigned, but Citizen was going to dump it anyhow). Replacing them are **Computacenter** and a newly-formed Akhter Group operation called **Microsales Distribution**. These join four existing distributors with whom Citizen seems happy.

Meanwhile **Northamber**, which still has lots of lines to sell, included the news of a **£5.4 million rights issue** with its best-ever year-end results — turnover up by 89 per cent to **£67 million**, profits nearly doubled to **£5.2 million**. The rights issue is supposed to reduce borrowings and take advantage of early-settlement discounts on bills, but competitors have suggested that Northamber is feeling the same kind of constriction in the IBM marketplace that MBS blamed.

The other big quoted distributor, **P&P**, has just reported interim results that must look good to the City — profits 74 per cent higher at **£3.3 million**, turnover up 48 per cent to **£50 million**.

Our new address

As visitors to subterranean Kings Cross will have discovered, TP Group has moved to the tasteful wilds of up-and-coming Hackney, swapping a rented basement for a smaller but wholly-owned (well, Nat West owns it) office with multiple views of the sky. These are the details:

Unit 20
London Fields Studios
11-17 Exmouth Place
London E8 3RW

Telephone 01-254 2245
Telex 9312102507 TP G
Telecom Gold 72:MAG068

DTI buys Apricot

The **Department of Trade and Industry** is spending an initial £150,000 on an Apricot network to link 15 regional centres to the DTI's Technology Transfer database. Apricot's VX1000 LAN system and Harwell's STATUS text database software are being used; the value of the order isn't as important as the public-sector penetration it represents — Apricot can now claim to have 30 per cent of the desktop computer business in this marketplace.

DOS 4.0 arrives

The newly-release **MS-DOS version 4.0** fixes several of the irritations in its predecessors — more than 20 enhancements and new utilities have been incorporated. The key amendments are support for hard disk files of up to 512MB (the current limit is 32MB), expanded memory support to beat the 640K restriction on user memory, a built-in front-end 'shell' with pull-down menus for easy file and directory management ... And that's about it. IBM apparently did most of the work on the new DOS and licensed it back to Microsoft; so MS-DOS 4.0 should be available to end-users (at around $95, we hear) in the Autumn. At the launch, Microsoft was keen to kill the "unfounded speculation" that the launch of OS/2 "spelt the end of MS-DOS ... We expect MS-DOS, WINDOWS and OS/2 to co-exist for a very long time to come". MS-DOS 4.0 probably won't work on anything but IBM-compatibles, but we'll get hold of a copy and check it out for the Apricot.

Deeper disks

Verbatim has announced a floppy disk drive that fits a standard half-height slot and stores *12MB* per diskette. The target market is security backing-up, presumably as an alternative to tape streamers. Where will it all end?

Top database?

The PC User show last month included a 'Database Challenge', with two-person teams from eight database vendors invited to write a test system based on the membership records of the white-collar union APEX. Microsoft and Ashton-Tate both declined to take part — Ashton-Tate in particular pointed out that ongoing support was at least as important as meeting the spec — and clearly the ability of the team was as much at stake as the software's capabilities. Anyhow, joint winners were **Precision Software**'s new SUPERBASE 4 and DATAEASE by **Sapphire Systems**; those defeated included DELTA, Blyth's OMNIS QUARTZ, Borland's PARADOX and Migent's much-touted EMERALD BAY.

DATAEASE, which is still available for the old-style Apricot, is currently second only to DBASE III+ in the Romtec list of UK database package sales; and Sapphire was immediately able to announce a minor coup in displacing Compsoft's DELTA 4 for Thames TV's Olympic coverage.

Gates appeal

Bill Gates of **Microsoft** has been voted 'Best Executive of 1988' by the readers of a US mag called *Electronic Business*. Gates beat 12 other candidates in a survey which polled 50,000 readers.

Some evidence for his position comes from the record year-end results just reported by Microsoft — profits up 72 per cent at **$124 million**, turnover improving by the same proportion to nearly **$591 million**.

Poor service for small punters

Kalamazoo has just published the results of an interesting survey that highlights the problems faced by some 261 small-computer buyers (and indeed small computer buyers). Over half had installation hassles which meant their systems took a month or more before they were up and running; incompatibilities between hardware and software were the main cause. And once things were operational, back-up support was generally considered poor — on maintenance in particular, some 43 per cent of respondents expressed themselves dissatisfied. Sounds familiar?

More international Dialcom

British Telecom has clocked up two more licensees for its **Dialcom** electronic mail systems subsidiary, with **Malta** and **Spain** joining the existing 18. Subscribers on Dialcom systems — such as Britian's Telecom Gold — can e-mail each other, so this brings the total user base to over **280,000** subscribers worldwide.

On-disk applications

UCCA was never like this: students applying to the **University of Richmond** in Virginia can now be sent a disk containing all the forms of a conventional paper application as well as extras like the college football team's song. ◆

Scratch 'n' sniff

A Japanese building company has reported that keyboard work benefits substantially from an appropriate aroma. Data entry errors fell by 20 per cent with lavender in the air, 33 per cent with jasmine, and fully 54 per cent for lemon and eucalyptus. Roger remains convinced that Brut is the best option.