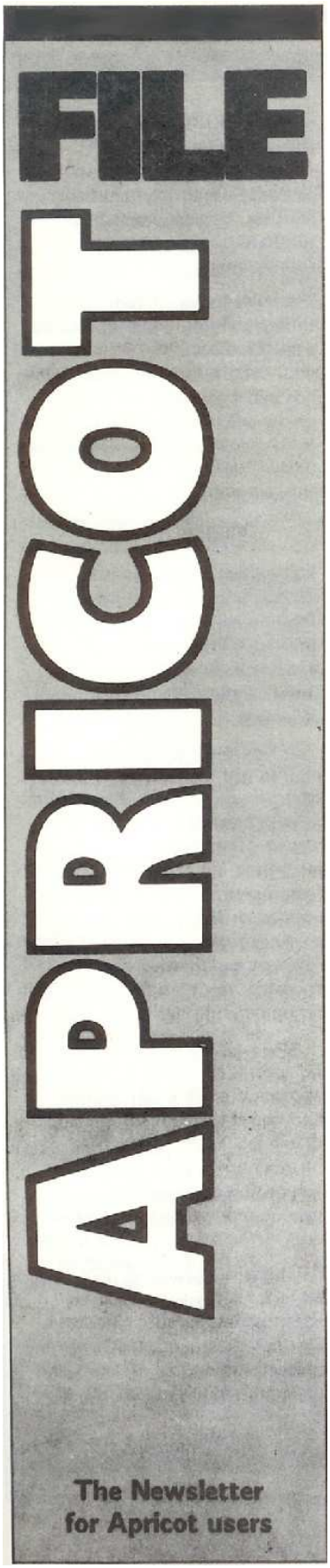
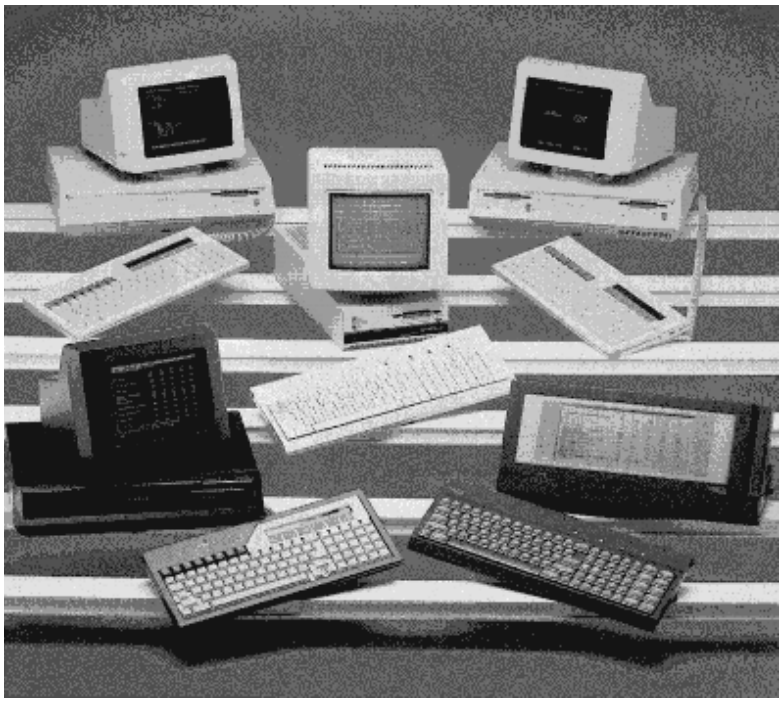


Apricot File 3.9



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 3.9

FILE

APRICOT

The Newsletter for Apricot users

June 1988

There's Gold in them thar bills

Charging problems on Telecom Gold

Prestel's price hike

Mercury moves

Telecom Gold appears to be making more money than it should from its customers. According to the FT's *FinTech 2 — Electronic Office* newsletter, you're billed if you try to access one of the on-line services and get the 'temporarily unavailable' message from the gateway connection. You'd think that since you haven't connected, you won't be charged; but Telecom Gold counts the failed access as billable time. So you're paying the minimum access fee, *plus* the connect-time charge, *plus* the per-character rate to receive the 'try again later' message (and maybe packet-switching charges too, if you're using PSS).

Charges for these gateway services range from 38p to £3 per minute "or part thereof". So a series of failed connections to say the Infocheck company credit service would mount up; *FinTech* reckons each failed attempt is worth at least £3.15 to Gold for this premium-priced service — £3 for the one-minute minimum, 6.5p for a minute's connect time, 4p character charge for one block, and one 4.4p unit for the phone time.

What's more, you pay for dead time whether or not you reach the database — you're charged during the connection and disconnection phase even though nothing is actually happening. In practice this can mean 20 to 30 seconds per access.

All this affects externals services, those that don't run on Gold's own computers. *FinTech* thinks services provided on Telecom Gold computers — such as the on-line version of its own newsletters — "probably do not suffer from the same defect".

The *FinTech* article highlights two other problems with Gold's charging. Rounding up is applied by all e-mail suppliers to all their services; but at £3 per minute, you might expect that the rounding-up is applied on a monthly basis. Instead it's applied to each and every call.

And some Gold users have been unpleasantly surprised to find 'storage charges' sneaking on to their bill. Any mail and/or Telexes left in your mailbox for more than 30 days are automatically stored elsewhere on the system; you have to be pretty conscientious about checking for and deleting these filed messages. Otherwise you might be paying for old and unwanted messages at a minimum of 20p per page per month.

CONTINUED ON PAGE 2

In this issue

Disinformation: On Patches of Turbulence 3
Writers in this issue 7
Opinion: We are all pirates now 8
The Workshop: A Turbo Pascal Primer — Part Six 10
The Workshop: Archive your files 13
The Q&A Clinic: Queries from subscribers 14
The Workshop: Practical experience with PC-File/R 17
Notebook: File o' Facts 22
The Workshop: Replace the MicroScreen 23
The Workshop: MS-Basic screen dump 24
Free Small Ads 27

None of this is illegal; as *FinTech* puts it, "Telecom Gold's conditions of use may well entitle it to levy charges such as those outlined". On the other hand, those conditions of use do impose a duty "to exercise reasonable skill and care". Telecom Gold says it's looking into the points *FinTech* raised.

Regular readers will know the name **Friedman Wagner-Dobler** (well, you wouldn't forget it, would you?) — he's a sharp technical guy who occasionally writes for us on computers and comms. It's him we have to thank for bringing the above to the attention of the world (and maybe even to Telecom Gold's management). Incidentally, we've just added his little paperback *aide memoire* on the C language to our DEALS FOR READERS lists along with an introductory tome on the subject.

Gateways going and coming

There's no connection with the foregoing, but the on-line version of VNU's *IDB* daily computer-industry newsletter will no longer be available as one of Telecom Gold's gateway services. VNU says it wasn't happy with the current arrangements, though industry sources suggest that a lack of profits might have had something to do with it.

On the other hand, Gold has added a service that probably *will* be susceptible to problematic charging. It's another travel-oriented gateway, a link to **British Telecom Travel Services** that joins the existing OAG airline routes and fares database; BTTS provides the ABC services, with information on scheduled flights, holidays, over 35,000 hotels in 170 countries, and local travel information on overseas destinations.

Prestel charges rise

As if the Telecom-related news wasn't bad enough, **Prestel** has announced big price increases on its services. With effect from 1 July, Prestel subscriptions rose from £66 to **£79.95** a year for home users; business users will now pay **£119.95** pa. And peak-time charges have increased from 6p a minute to **7p**.

What's more, peak-time rates will now apply on Saturdays between 8am and 6pm; and free off-peak usage has been scrapped — users will now have to pay 1p a minute.

At least **Micronet 800**, the microcomputing service that claims to be the largest information provider on BT's viewdata system, has won a small concession for its subscribers — no time charges for access between midnight and 8am. But most users will find that the new off-peak rates effectively double their outlay per hour; previously, subscribers were required to pay only the cost of a local phone call to their nearest Prestel node.

The reason for the price increases appears to be the strictures against cross-subsidy within privatised of BT — one profitable area of activities (particularly local telephone services) cannot be used to subsidise the development of another (like Prestel, which has been struggling to find both a rôle and a way of making real money — after nearly nine years, Prestel is still only just in profit).

Steve Gold on *Newsbytes*, an expert comms commentator, remarks that he expects users to leave in droves; and he "finds it difficult to see the system continuing as a viable business proposition ... 1 July 1988 could be a sad landmark in UK communications history".

OFTEL on Telecom

Neither Telecom Gold's charging nor Prestel's policies are within the remit of **Office of Telecommunications**, which is perhaps a shame. In OFTEL's fourth report on telecommunications the industry watchdog does nominate British Telecom's quality of service as one of the major causes for concern in 1987; in particular, only 74 per cent of faults were repaired by the end of the next working day compared to 90 per cent the previous year. OFTEL attributes this to the engineers' strike, the October hurricane and "unpredicted acceleration in demand".

OFTEL itself also experienced a dramatic increase in complaints — from 13,660 in 1986 to 24,186 last year. Over a third of these related to billing disputes; BT has agreed to improve its billing procedures and is still promising the long-awaited itemised billing that should help both sides.

On the question of BABT approval, the pilot scheme introduced last year whereby applicants can deal direct with test labs has definitely speeded things up — BABT says approval can be obtained within five weeks for some products. Overall, some 1,320 items were type-approved during 1987.

Mercury moves

By now you might be thinking that **Mercury Communications** looks a better and better alternative to Telecom. The latest of Mercury's links is a deal with the Deutsche Bundespost to operate a switched telephone service between the UK and the Federal Republic — the busiest traffic stream in Europe, accounting for a quarter of all communications between Britain and the Continent. A direct Mercury service will start next January. With the existing agreements, this means that Mercury could be used for more than 85 per cent of international calls.

And Compulink has moved from BT to Mercury for packet-switched access to its **CIX** Compulink Information eXchange, one of the more interesting paid-for bulletin boards with a computer-interest bias. CIX is now available nationally on Mercury 5000. That's Mercury Communications' public data network, the competitor to BT's PSS; Mercury 5000 can go faster (2400 bps) and Compulink has invested in Tricom modems to provide the dial-up ports — Tricoms feature MNP level 5 error-correction for high data reliability, and MNP level 5 also has a data compression system which should mean an effective data rate approaching 4800 bps. For a taste of CIX, call **01-390 8446** (voice) or **01-399 5252** (data).

The Mercury news isn't all good, however. The **Mercury Link 7500** e-mail service (*né* EasyLink) has slashed usage charges for its Telex facility — UK inland Telexes drop from 25p to 10p a minute — but has raised the monthly subscription rate from £12.95 to £20. The net effect should benefit volume users at the expense of the occasional Telexer. ◆

The Disinformation Column

On patches of Turbulence

By David Langford

I've been playing with Turbo Pascal 4.0. This major upgrade from Borland International isn't likely to become available for 'generic MS-DOS' machines like the Apricot. Certainly a universally-installable one would be hellishly boring to set up, since Turbo 4 has a vast apparatus of windows and pop-up menus which make heavy use of direct IBM video memory addressing. It would also be near-impossible to 'genericise' the impressive library of graphics facilities. (There's even a *PieSlice* procedure.)

Still, APRICOT FILE is taking on an increasingly funereal tone as article after article concerns itself with moving to IBM: if you move, Turbo 4 immediately becomes an attractive proposition. So what about all those old Apricot Turbo Pascal version 3 programs? After converting a lot of stuff to and fro, I can report.

What you get

If you're resigned to the resistless march of the IBM, Turbo 4 is certainly worth it. Both compilation and display handling speed increased dramatically when I recompiled old programs. The multi-window, multi-menu display is very pretty and has practical advantages: you can keep an eye on (and indeed zoom to a whole screenful of) the output produced by the last test run, without leaving the editor. You can escape from Turbo on a 'shell' basis, to do things in DOS without losing your place in the file. You can in fact leave Turbo, switch off the computer, go on holiday for six weeks, forget the name of the program you were editing, and still come straight back to the same position of the same Pascal file! The compiler/editor package saves this kind of information in 'pick files', and (unless you specify another name) reloads the file last edited.

Turbo 4 produces .EXE rather than .COM programs — but although a typical .EXE file is a few hundred bytes bigger than its .COM equivalent, a Turbo 4 program usually turns out several KB smaller than the Turbo 3 version. There's a fairly effective 'dead code removal' system which weeds out uncalled procedures, unused library functions and statements that never get executed. A program which does nothing whatever occupies about 10K in Turbo 3 (the size of the run-time library code); it takes exactly 1,360 bytes in Turbo 4. My own programs shrank by an average of 5 to 6KB when recompiled.

Finally, the manual is fatter and much improved: 654 pages, hardback, indexed, with all procedure and function descriptions in one big alphabetical chunk. It's easier to find things than in the old layout with string procedures here, file procedures there, procedure procedures elsewhere, and functions in a list of their own.

Upgrade problems and quirks

Borland supplies an UPGRADE.EXE converter for your old Turbo 3 .PAS source: it doesn't really do a full conversion job, but makes some basic changes and fills the source code files with hordes of bright comments. An example, following my usual *{$C-}* compiler directive:

```
{ Directives A, B, C, D, F, G, P, U, W,
and X are obsolete or changed in
meaning. }
```

I had some trouble with UPGRADE — it kept giving cut-down versions of 'converted' source files with included libraries. One library shrank to a single byte. In the end, it seemed easier to compile the old stuff with Turbo 4 proper, and make changes as advised by error messages.

Here's what I ran into, in no particular order:

- The first thing every converted program needs is a *Uses* statement, defining which standard library 'units' are to be drawn on. Most programs require *Uses CRT* (video routines); several of mine went on to *Uses CRT, DOS, Printer;* (self-explanatory).
- Turbo 4 allows advanced users to define their own precompiled 'units'; the former 64KB code limit has become a limit on unit size only, allowing mega-colossal .EXE files.
- Overlays — did you ever use them? — are now passé: you must convert to units.
- As hinted in the sample UPGRADE message above, the old *{$C-}* directive (speeds display output and turns on the type-ahead buffer for *KeyPressed* and keyboard reads) is no more. The memory-mapped display couldn't be faster; the type-ahead buffer is now always available.
- *IOresult*, the good old I/O error report function used in conjunction with the *{$I-}* and *{$I+}* directives, still returns zero if nothing has gone wrong. But if your program's response depends on which specific error code was reported, be warned that the I/O error code numbers have changed.
- The input statement *Read(Kbd,Key);* becomes *Key:= ReadKey;* and this accounted for many of my changes. One could prepare for the life to come by adding the *ReadKey* function to old Turbo 3 libraries, to be dropped when you move to 4:

```
Function ReadKey: Char;
Var
  Input: Char;
begin
  Read(Kbd,Input);
  ReadKey:= Input;
end;
```

- The *Kbd* device no longer exists in Turbo 4, though it and some other superseded keywords can be simulated by including *Turbo3* in your *Uses* statement. On the whole the manual discourages this: to get maximum efficiency from Turbo 4, you should go the whole way.
- The optional buffers for text files are no longer set in the *Var* definition (eg *Var ApricotFile: Text[$2000];*). There is a special new procedure called *SetTextBuf* which allows you to assign a named array of any size or type as being an input/output buffer for the file. *SetTextBuf* is best used after the file's *Assign* statement but before actual access. In case you were wondering, the point is that by allowing text to be transferred to and from disk in bigger chunks (several KB of buffer rather than the default 128 bytes of MS-DOS), file reads and writes can be speeded up. It mightn't help much if you use MicroCache or something similar — but remember that your programs might well be used by people who don't have MicroCache.
- If any Labels, those naughty wicked things, have been defined... Turbo 4 insists that you use them!
- Here's an odd and subtle one. Is there something wrong with the statement ...

```
For Index:= 1 to 10 do Writeln('Hello?');
```

  Not necessarily: but if the statement is in a procedure which itself is within a procedure, and if the Byte/Integer variable *Index* isn't global (defined in a main-program *Var* statement) or local to the procedure (defined in the inner procedure's own *Var* declarations), then you get an error message.
- How about ...

```
Index:= $FFFF;
```

  ... where *Index* is an Integer? Not allowed in Turbo 4, but there's an excellent compensation. The new variable type *Word* takes the range ø to $FFFF (decimal 0 to 65535) — unlike Integer, it doesn't waste half its range on negative values, and is thus handier for loop variables.
- In addition you get the new types *ShortInt* (byte-sized, range -128 to +127) and *LongInt* (four bytes, range -2147483648 to +2147483647). Several sensible changes to procedures have been made possible by *LongInt* in Turbo 4. For example, the *FileSize* function now returns a *LongInt* value, so *LongFileSize* has become obsolete. *MemAvail*, the function which reports how much unused memory is available on the 'Heap' (of which more below), now gives the answer in bytes as a *LongInt* rather than in 'paragraphs' of 16 bytes — coded thus to fit the result into a plain Integer.
- 'Inline' machine code and *MsDos/Intr* calls may well need revision: there have been changes to segment-handling, so that (for example) typed constants are no longer found in *Cseg*. 'Absolute *Cseg*' and 'Absolute *Dseg*' variables are *verboten*. Compensation? A predefined 'Registers' type in the 'DOS' library unit, and some nifty directory search procedures called *FindFirst* and *FindNext* which allow you to do wildcard directory scans without having to look up the MS-DOS calls and register loadings as of old.
- 'External' machine-coded procedures (which few people ever used because they're so poorly documented in Turbo 3) must be reassembled as standard-format .OBJ object-code files.
- Perhaps the only really weird item is the hyper-strict type checking. This hasn't bothered me directly, but the manual gives an alarming example using a couple of pointer-to-integer variables:

```
Var
  A: ^Integer;
  B: ^Integer;
```

  After this definition, *A:= B;* or vice-versa will cause a compile error because of type mismatch! ''Variables must have identically named types or be declared together in order to be assignment-compatible.'' The assignment *A:= B;* would work if the declaration looked like this:

```
Var
  A, B: ^Integer;
```

  ... or this:

```
Type
  IntegerPointer = ^Integer;
Var
  A: IntegerPointer;
  B: IntegerPointer;
```

Subtle stuff, eh? New Pascal programmers often scratch their heads a lot about the whole business of pointers: what is the (as it were) point? Let's have a digression on pointer variables, their care and feeding. The following section applies equally to the old Turbo and the new.

A heap of pointers

The segmented architecture of the 8086/8088 chip makes it handy to address a 64K data segment with 16-bit variable addresses; but the average MS-DOS machine has got oodles of spare memory beyond that puny amount. With a 32-bit 'pointer' address, we can store and read stuff anywhere in a megabyte.

Turbo gives us a special syntax for organising this vast waste of memory which isn't handily addressable in the usual 64K blocks. All the spare memory not being used by the operating system, the program code, the 'normal' data and the system stack is called the Heap. Pointers are used to

address an odd sort of variable found only in the Heap: 'dynamic' variables, which lead nebulous lives, can be created and destroyed at will, and (very Kafkaesque, this) don't have names.

Let's imagine that I'm writing a program which does something pretty clever with strings, a large number of them, too many to accommodate in the data segment but not so hugely many as to require the delays of disk storage. Let's say my strings are to be at most 40 characters long. I start programming like this:

```
Type
  MyString = String[40];
  MyPointer = ^MyRecord;
  MyRecord  = Record
    TextInfo: MyString;
    Next: MyPointer
    end;
Var
  FirstPtr,WorkPtr: MyPointer;
  WorkString: MyString;
  Index,
  DataCount: Integer;
    {Word or LongInt would be better here in
    Turbo 4}
```

I've now defined my string type, and a Heap pointer type which points to (that's the significance of the enigmatic ^) a *MyRecord* record, and the record type itself. Note that *MyPointer* can and must be defined in terms of *MyRecord* before the latter has itself been defined. The whole cunning of this lies in the fact that *MyRecord* includes a pointer of its own ...

In the program proper, as I read my strings in from the keyboard, a disk file or an RS-232 interfaced ouija board, they go one by one into the temporary variable *WorkString*. I'm going to store them in the Heap, first setting things up as follows:

```
New(FirstPtr);
NextPtr:= FirstPtr;
DataCount:= 0;
```

The first line creates a new, nameless variable in the Heap, and points *FirstPtr* in its direction. The Heap variable is of type *MyRecord* because that's the sort of variable type associated with *FirstPtr* and indeed all *MyPointer* variables. I want to keep a record of *FirstPtr*, since it's my key to the whole coming sequence of Heap variables — this is why the following code uses *NextPtr*, whose value is going to change a lot. We're ready to store stuff in the first Heap variable, since *NextPtr* points to it: the nameless variable has a nickname, consisting of its pointer's name followed by ^.

```
With NextPtr^ do begin
  TextInfo:= WorkString;
  New(NextPtr);
  Next:= NextPtr;
  DataCount:= DataCount+1
  end;
```

This chunk of code clearly belongs in a *Repeat/Until* loop which will execute it again and again until (a) a check on *MemAvail* reports that there's no room left for more Heap variables, or (b) we get to the end of the data. The *New* statement in the loop creates the pointer for the record which is to follow the present one, and we store this pointer in the *Next* field of the current record. (*DataCount* merely keeps track of how many records there are.)

Each record thus contains the pointer to the next, and we don't need a vast array of pointers. Instead, to access all those text strings on the heap, we just follow the arrows from record to record, starting at *FirstPtr*.

Let's now suppose that we desperately need to recall the 1,000th of those strings from the heap. We know where to find the first; *FirstPtr* points straight to it. We need to trace a path 999 pointers on, thus:

```
NextPtr:= FirstPtr;
For Index:= 1 to 999 do NextPtr:=
  NextPtr^.Next;
Writeln(NextPtr^.TextInfo);
```

This time I've used the alternative means of reading record fields: *Writeln(NextPtr^.TextInfo);* is identical in meaning to *With NextPtr^ do Writeln(TextInfo);*. In real life this would be expanded into a generalised procedure to fetch any record, with a test to ensure that the requested record isn't beyond the range 1 to *DataCount*.

All unnecessarily tortuous and slothful-seeming? It's certainly not the intuitively obvious way to handle large masses of data, but it's surprisingly powerful when you get used to it, and very much faster than making up temporary disk files. For simplicity I've used a worst-case scenario in which every record between the first and the one required must be looked at; there are other ways to organise this, such as a short-cut record of (say) every hundredth Heap pointer, allowing speedier searches.

What's more, the technique adjusts itself automatically to the available memory ... and you can always overflow into disk files when space runs out: the Turbo Database Toolbox seems to work that way. Turbo Pascal also allows you to wipe all or some of the nameless records from the Heap and start afresh, but I suspect that all or some of the people at the back of the audience are getting restless and hoping for an interval.

Apricot and IBM Turbo: keys

Back in the world of Turbo Pascal discrepancies ... The famous Apricot soft keyboard can actually be a bit of a pain when you're writing programs. You carefully research the escape sequences produced by the cursor keys, and (not forgetting to set the *{$C-}* directive) write a bit of code like this:

```
Read(Kbd,Key);
If (Key=#27) and KeyPressed then begin
  Read(Kbd,Key);
  Case Key of
    'A': Up;
    'B': Down;
    'C': Right;
    'D': Left;
   end
  end
  else ...
```

(The procedures *Up*, *Down* and so on are left as exercises for the student.) This works fine, until someone tries to run your program after (say) leaving Apricot WordPerfect via 'GO TO DOS'. Now the cursor keys are programmed to the deeply useless WordPerfect codes #23 to #26, alias ^W, ^X, ^Y and ^Z. Instant disaster! Configuring the keys from within your program, via the appropriate escape sequences (see the *Complete Escapes* series in APRICOT FILE Release 2.10), will make users say horrid things when after EXITing back to WordPerfect they discover that the cursors no longer work there.

A good solution on Apricots is to use the Black Hole RINSTALL.COM utility to make a keyboard loader which temporarily imposes the standard Apricot configuration. Incidentally, you can tell RINSTALL to load and run a program which isn't a .COM or .EXE file. Once, I deviously renamed my Turbo .COM file with the extension .OVL and gave the old .COM name to the RINSTALL-generated loader. Since the two files had the same main filename,they obviously belonged together (handy for users trying to organise their directories), while slightly more discerning customers were duly awestruck at my seeming power-user ability to handle overlays ...

This strays away from IBM Turbo, whose keyboard-read of 'function' keys formerly generated escape sequences curiously like the Apricot cursor keys. In fact the above fragment of Pascal would work in IBM Turbo 3 if the letters A to D were changed to H, P, M and K respectively. (The arbitrary-seeming letters correspond to keyboard scan codes.) The use of Escape was always a bit illogical, though, because you had to do that *KeyPressed* test to check that it wasn't just a press of the single-character ESC key itself. In Turbo 4, the lead-in character for special function keys is now zero ... so to update the mini-program again, we'd have to substitute #Ø for #27 in the *If* test.

Apricot and IBM Turbo: Escapes

The worst thing about converting an Apricot program is the need to write cumbersome substitutes for the screen-control escape sequences that are so familiar. You can take the ANSI route by restricting yourself to the ANSI screen control sequences: but there aren't enough of them, they aren't as powerful as one would like, and they're slower — more characters to be sent. The real argument against IBM ANSI sequences is that they're not built into the BIOS, like Apricot's. Your victim has to be persuaded to rewrite CONFIG.SYS and install the special ANSI.SYS device driver; at which point he is liable to respond "Why should I go to all this effort just to run your rotten old program?".

So, cumbersomeness. The single Apricot statement

```
Writeln(#27')Hello, '^['(Roger Gann'^[,
  ' this is your life!');
```

uses the ESC) 'dim display' and ESC ('bright display' commands to glorify Roger's name. I've used #27 (character 27 from the ASCII table) and ^[(CONTROL [) representations of Escape as a reminder that they're exact synonyms in Turbo. Over in the IBM world, even taking advantage of Turbo's power and non-standard screen commands, we end up with six whole lines:

```
TextColor(LightGray);   {"dim" display}
Write('Hello, ');
TextColor(White);       {"bright" display}
Write('Roger Gann');
TextColor(LightGray);   {"dim" again}
Writeln(', this is your life!);
```

This example was cunningly chosen, and not merely to remind you that Turbo insists on nauseating American spellings. It actually has the same effect of bright and dim text on the standard IBM PC monochrome and colour video adaptors. This is unusual. A text mode like underline can be done on the former, but has to be arbitrarily colour-coded on the latter; so the program gets complicated with BIOS interrupt calls which deduce what sort of machine the program is running on, and whether it has one of the above video systems or maybe the EGA or VGA or MCGA or Hercules video cards, and on the whole I rather think I need a large drink.

To be fair, Turbo 4 has some excellent example programs which help you get to grips with this problem. But it's all a long way from the happy land of Apricot, where the powerful Escape sequences for scores of screen control functions have spoilt us rotten. Down these mean streets a man must go ...

Metaphysics dept.

The last question: is Turbo 4 as it now exists of any use to the dying breed of Apricot users? Roger suggested, "What about the command line version?" This compiler, TPC.EXE, is thrown in for free with TURBO.EXE and the rest of the Turbo 4 package. It has no display or editing facilities: it just compiles. And like other unfancy IBM compilers and assemblers, it works perfectly well on an Apricot.

However, the memory-mapped IBM video output seems at first sight to imply that Turbo 4

programs won't show text (let alone graphics) on an old-style Apricot, IBM emulator or no. I toyed with the idea of working up a library Unit of my own which would enable me to use the improved power and speed of Turbo 4 to generate Apricot programs: one would say 'Uses *Dos*, *Printer*, *Apricot*;' and all the video output routines would produce Apricot BIOS output to the Control Device. It would work, in theory: the principle was tested with an awesome ten-line program which sent one character (one doesn't want to overdo things) to the screen, via an Apricot interrupt call. It compiled nicely on the IBM, compiled (with TPC.EXE) and worked nicely on all the Apricots. Suddenly, fascinating old horizons seemed to open up ...

But I'd been this way before, when (in Turbo 3) I wrote my own memory¬mapped video output library. The Pascal rules don't let mere users create routines as flexible as the standard *Write* and *Writeln*, with their variable number of parameters, built-in format controls, etc. One would be doing all one's display output through tortuous Heath-Robinson procedures, and longing for the straightforward (but unworkable on Apricots) Turbo 4 display routines.

Then, trudging again through the more obscure recesses of the Turbo 4 manual, I made a happy discovery. If you omit the *Crt* unit which handles the blazingly fast IBM memory-mapped output, Turbo 4 still supports *Write* and *Writeln*, letting their output route default back to normal 06-DOS_... slow but reliable and Apricot-compatible. In this way you can write Turbo 4 programs on the Apricot (using, say, the Turbo 3 editor), compile them on the Apricot, and run them. Much fun lies ahead: we can all get down to writing escape-sequence Apricot versions of lost *Crt* procedures like *GoToXY*. Apart from this long chore, the only snag is transferring TPC.EXE from an IBM to an Apricot disk.

Is it worth shelling out for Turbo 4 merely to get the 'free' poor¬relation compiler TPC.EXE? Only if you plan to run it on an IBM too, or if some of the added Turbo 4 features sound exciting enough to justify the change ... remembering that Apricots may benefit from new variable types, new procedures, and .EXE file generation, but can't use all the graphics stuff in *Crt*. Meanwhile, for now, Turbo 3 will in its unpolished way do most things you could need, especially if you don't mind delving into the *MsDos*, *Intr* and *InLine* procedures for twiddly bits (like directory searching, or fast handling of the unlimited-length strings I invented when the normal maximum of 255 characters became irksome). Turbo 4 remains a happy dream on the horizon; I promise you, it sweetens the bitter pill of capitulation to horrible old IBM 'standards'.

This column has been another of my many aberrations. Next time, it's Apricot all the way, and I will tell you the secret of how not to have to type that irritating 'Y' to load Turbo's error messages (or 'N' not to, as you prefer) ever again. Yes, in a return to our most popular theme, it's Tiny COM File time again. ◆

Contributors in this issue

Dennis Jarrett is Managing Editor of the TP Group newsletters and a frequent contributor.

Jeff Ashurst is a chartered engineer who believes that personal computing can improve both the efficiency and the quality of life. His specialist interest is in computer communications, and he writes on various aspects of the subject.

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.

Guy Sturges worked as a programmer for the largest software house in England before moving to Switzerland. He is now working in telecommunications, systems and programming in Geneva. Guy purchased an Apricot PC in October 1986 ("it was only then that they were a reasonable price") and is putting all the pieces together to make it work as it should.

Mark Whitehorn started life as a Cytogenicist but the Siren Song of the computer was too strong and he is now working as a programmer at a Scottish university, particularly on the application of computers (especially Apricots) to education. 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).

Opinion

Jolly Rogering

We are all pirates now ...

By Eddie Bromhead

How many times have you seen the phrase "software piracy is theft" in the literature, or come across articles explaining why it is now essential that there should be a blanket tax on blank magnetic media?

I have a different view: most software *sales* are theft! Programs that don't live up to expectations are pretty much the rule, rather than the exception. Usually, the more it costs, the less likely it is to work correctly, and there are good reasons for this to be a hard-and-fast rule. Perhaps it is inevitable that even computer users who start with the best of intentions stray from the straight-and-narrow.

The official view

What follows is a set of random thoughts on intentional software piracy. But first — the official line on why you should buy first, try later, *and* pay full price.

Software is sold on the tomato sauce principle. The profit margin isn't on what people eat; it's in what is left on the plate and in the bottle. Software suppliers like people to buy shrink-wrapped packages, find the product useless, and sling it!

What's more, if you pay full price from an authorised dealer (rather than heavily discounted through Penthouse PCs) you get full dealer support and probably hotline telephone support from the originators (sounds of APRICOT FILE readers choking).

More cost, less functionality

Seriously, why should it be that the more you pay, the worse the software performs?

The answer is simple: software is labour-intensive to write, and if the development costs are amortised over a few copies it has to be expensive.

What's more, a small user base — which is likely if the software is specialised and/or expensive — means the product can never be tested fully or be completely refined. Think how many millions of legal copies of WordStar there must be *(about 3.5 million — Ed)*; and even *that* product still has bugs.

Deliberate errors

What about software that runs through release after release, with serial numbers that change like the meter on a London cab?

I'll give an example: Microsoft Fortran. I started with 3.10 (£350). I upgraded to 3.13 for a further tenner (from Apricot direct — my dealer had disappeared — though I did get the lecture on "we don't normally deal with customers direct"). Then I moved to 3.31 (missing out 3.20 and 3.30) for another £100 from Microsoft, who for all the crap in the licence booklet haven't offered me a good deal on the latest version ...

That's over £500 when the VAT is totalled in — and for a product that *still* doesn't work properly. This degree of incompetence must be by design!

The single-user licence

Most commercial software turns up with a legal document licensing you to operate the software in a given fashion but explicitly disclaiming liability should said software louse up your business.

The given fashion normally includes use on a single processor. For a moment, I'll forget that most of my software purchases have been for myself, and give an example from the Polytechnic. Try teaching a group of students on a single machine! Multiple copies of (say) StarWord at £300 a throw come to the cost of a couple of extra machines, and just try preventing those little beastie boys running off copies like the expert counterfeiters they are ...

The Pirate's Charter

Now for some sick-making hypocrisy:

- I am an Author.
- I live on Royalties.
- I therefore hate Copyright violation.
- I write a book on how to use a program.
- That program costs £500.
- My book costs £20.
- With the book, you don't need the manuals — all you have to do is to rip off the software on disks (about £6 worth for Ventura, £5 for PageMaker).
- Aren't I lucky that if Xerox or Aldus puts my book on a disk, I can sue?

An example of this is Antonia Stacey Jolles' *Making the Most of Pagemaker* (published by Sybex: don't ask the price, it's £19.95).

This book duplicates the Aldus manuals, but is inferior in many ways. For one, it doesn't lay flat — the manuals do. Typography in the manuals is better, and there are two vital small documents which the book doesn't give: an installation guide and a quick reference card.

The book is useless to the enquirer who wants to know whether it will work with a particular computer, a printer, whether it needs a specific version of WINDOWS, what font support there is, and so on. *It exists solely to support the software pirate.* £19.95 is not that much dearer than photocopying the manuals if you take your time into account.

The Apricot lament

Then there are the systems which you think promote portability across the IBM-compatibility divide: GEM and WINDOWS (at least for some!).

Just try it! List some useful GEM software, like Ventura Publisher, GEM Draw Plus, GEM DTP ... and what do they all need? Certainly not the GEM 1 you get on Apricots (ok, so I know you can get version 1 of a few GEM packages, and really well-written software like Prospero's Products for GEM don't care which version you use).

So what about WINDOWS? The likes of IN*A*VISION (CAD) and PageMaker require WINDOWS version 1.03 at a minimum, and what do you get on a Xen? Version 1.02, that's what! You could easily lash out £1,500 on software that won't run (or won't run well) on Apricot's version of the systems. (Actually, the WINDOWS-compatibility of those two is such that 1.02 is ok. But that's not guaranteed.)

What then is the answer? Try before you buy? Well, in the majority of cases *that's* software piracy!

When theft isn't

I wouldn't like anyone to think I was advocating wholesale duplication of software and manuals — and especially the resale of same. That goes beyond the bounds of reasonableness, and definitely needs stamping on.

Let's start at the other extreme.

I keep copies of WordStar at home and in my office. I can't use both at once, because I haven't got arms 30 miles long. On the other hand, I couldn't ever swear to the fact that I didn't have a Research Assistant slaving over a hot PhD thesis, or a wife pounding out a Mills & Boone best-seller as I unsuspectingly keyed my latest missive to APRICOT FILE.

Is this a crime? Legally, yes. But morally?

Having laid out £500-plus on Microsoft Fortran (and hundreds more to Prospero, Digital Research, Supersoft *et al*) I 'borrow' a copy of version 23.97 to see if it works at last. I fail to purge it from my hard disk, even though because it works no better than the last two hundred versions bought at enormous cost. For serious work I stay with my (legal) previous version, which has the advantage of using 100MB less of disk. A crime? Legally, yes. Morally?

What to do?

My advice is simple. There ain't no software piracy detector vans out there, so 'borrow' and copy away with abandon. Try everything you can lay your hands on.

And when you find something that works, do everyone a favour.

First, *buy a legal copy*; that way the author gets the benefit. He certainly deserves it, being in that select few who write a useful product. He deserves it even more if the software is cheap! Not only that, but you get the latest version, and the latest documentation. With any luck, you will find extra facilities, and details of how to use them — but don't bank on it: no-one's perfect! You may even get on to the bandwagon of cheap upgrades, or support.

Second, *let the rest of us know* about this Apricot-compatible, useful, software. That also does the author some good, as well as the rest of the Apricot community. If you get support, sing the praises louder still.

Finally, *scrap the pirate copy or pass it on.* Scrapping is best, because it frees up disks which are still an expensive commodity. I used to pay £60 plus VAT for 10 single-sided when I bought my PC, and I get more than a hundred for that now (from Direct Disk Supplies in Hampton) — but you can easily amass junk which ties up a hundred pounds or so, and scrapping rubbish *is* cheaper than buying new disks. If it's on your hard disk, it is more expensive still — you might have paid up to £100 a megabyte for that disk!

This is treating all software as Shareware, whether it is or not.

And finally ...

Let us all know if you find something that's still marketed for Apricots ... ◆

A Turbo Pascal Primer

Part Six:
Declaration and definition

By Mark Whitehorn

This month we'll have a look at the way in which the programmer can declare and define things in Pascal.

Constants

The simplest definition you can make is that of a *Constant*. Undoubtedly the constant I use most frequently is *pi*, the value which haunted my schooldays — it seemed to crop up everywhere, yet nobody knew its true value. As time passed I got used to it and now find it indispensable, though I am still at a loss to understand why a ratio derived from the measurement of circles is used so frequently.

Every time it crops up in a program, I could type '3.14159'; but this is boring and I risk mistyping the numbers. Instead I declare *pi* as a constant, like this:

```
Const
pi = 3.141592654;
```

(Note that the *Const* section comes before the *Var* section.) Now the constant *pi* is always set to 3.141592654, and I can replace lines like ...

```
Radius = 3.141592654 * Diameter;
```

... with:

```
Radius = Pi * Diameter;
```

That is much more readable. In many ways I can use *Pi* as a Real variable, except that it has a fixed value and I cannot change that. So a statement like this is fine:

```
Writeln(Pi:5:2);
```

But this one will produce an error message:

```
Pi := 3.14;
```

That's ok, because I have no desire to allow *pi* to have any other value.

Note that I have never declared *Pi* to be of type Real — when reading the Constant declarations, the compiler finds the decimal point in the number and automatically knows that *Pi* must be a Real. If I had defined *Pi* like this ...

```
Const
pi = 3;
```

... in the program I would have to treat *Pi* as an Integer. I can also declare text:

```
Const
Greeting = 'Hello there. Welcome to this menu.';
```

Greetings becomes a String variable containing a fixed set of characters.

Paradoxically, constants are also very useful because they allow you to change 'constants' in your program.

Suppose some idiot tells you in the program specifications that the tissue solubility coefficient of acetylene is 0.760. Three months later, after program tests reveal a bug, it turns out that he was wrong and the value should have been 0.768. If you have declared the value as a constant, you have but a single value to alter and you can easily locate it in the *Const* section.

If you'd used 0.760 throughout the program, you will have to hunt all through the source code (which by now is split up into libraries and separate files) and change every occurrence.

Types

The next step is to start defining your own types. Turbo Pascal provides five different flavours of types with which we can play:

- Real
- Integer
- Char
- Boolean
- Byte

It is a mistake however to believe that these are the *only* types we can use. Pascal allows us to define our own; so don't think of the predefined types as the only 'proper' ones — they are the bare essentials, just there to get you going.

Never be afraid to define your own types if and when you need them. Next question: when do you need them?

The answer is "whenever you come across a collection of bits of data which are usually used together". The example most often quoted is the day of the week in something like a payroll program.

Suppose a firm needs to calculate bonuses depending on three factors:

- *who* does the work
- the *day* on which it is done
- the *kind* of work.

The programmer (a talented Antarctic explorer called Penguin Penguinsson, who writes software in his spare time) isn't aware that he can use a data type. He would probably declare an integer variable called say *DayOfTheWeek* to keep track of the day of the week; and he might arbitrarily decide that 1 was Monday, 2 was Tuesday, and so on. This would lead to code that looks like this:

```
If DayOfTheWeek = 5 then begin
   CalculateWages;
   DayOfTheWeek := DayOfTheWeek + 1;
```

The problems start when he tries to remember whether Monday was day number 1 — or was it Sunday?

Worse still, Penguinsson goes off exploring again and *you* end up debugging the code; and you find another bit that looks like this:

```
If (DayOfTheWeek = 7) and
   (Employee = 3) and (Work = 2) then
   WageSupplement = 5.50;
```

This is really annoying, because you know if only he had used defined data types it could look like this:

```
If (Day = Sat) and
   (Employee = TeaBoy) and (Work =
   CleaningUp) then
   WageSupplement = 5.50;
```

So how is Penguinsson going to program this in a readable way? First he must decide upon a name for the new types that he is creating. In this example he needs to keep track of the days of the week, the type of employee and the different extra jobs which can be done. He needs a new type for each of these — say *DayType*, *EmployeeType* and *JobType*. (Remember that these are *types*, in the same way as Char, Integer and so on.)

Next he has to tell the compiler that these *are* types. This is done in the type definition section which goes at the top of the program, between the *Const* section (if fitted) and the *Var* section. At the same time he tells the compiler all of the members of each type.

```
Type
DayType = (Mon,Tues,Wed,Thu,Fri,Sat,Sun);
EmployeeType =
(TeaBoy,Mechanic,Foreman,Gaffer);
WorkType = (CleaningUp,LubricatingChassis,
            ServicingCar,Welding);
```

(I tend to put the word 'Type' at the end of each name, but this is by no means obligatory.) So he now has three new types; he can use them in the normal manner and declare variables to be of whichever type he chooses — just as he would use (for example) Type Integer.

```
Var
Day          :DayType;
Employee     :EmployeeType;
Work         :WorkType;
Count        :Integer;
```

Let's examine exactly what he has done. He has declared *Count* as an *Integer* variable; so he can put integers into that variable, but not characters. In the same way, he has declared *Day* to be of type *DayType*. Integers are whole numbers, DayTypes are any of Mon, Tues, Wed, Thu, Fri, Sat, Sun. So these are both legal statements:

```
Count := 45;
Day := Fri;
```

But these are not:

```
Count := 'a';
Day := 4;
```

These user-defined types are useful, but there are problems associated with them. The biggest is that you cannot read or write variables which use them. So this line doesn't work:

```
Writeln(Day);
```

(But see the demonstration program for a way to deal with this.)

The demo

Examples always aid me to see how new things work, so I hope DEMO.PAS helps. Bear a couple of points in mind, though:

- This is a small program, so a relatively large effort goes into setting up and using the data types (procedures like *DayOutPut*). A tiny program like this can be written more neatly without user-defined data types. Don't let that put you off them; in a larger program the effort involved in handling the data types doesn't grow, but the benefits do.

- Just in case anyone criticises the overall approach, I agree this code is not the best solution for the particular problem — something like PROLOG is better suited to this kind of rule-based application. But it is designed to demonstrate complete code which uses defined data types in as little space as possible. As an exercise, try adding the missing procedures. ◆

DEMO.PAS

```
Const
Pi = 3.141592654;
Greeting = 'Welcome to this menu';

Type
DayType      = (Mon,Tue,Wed,Thu,Fri,Sat,Sun);
EmployeeType = (TeaBoy,Mechanic,Foreman,Gaffer);
WorkType     = (CleaningUp,LubricatingChassis,
               ServicingCar,Welding);

{  Don't split that WorkType declaration over two
   lines — we had to do it that way to fit our page
   layout  }

Var
Count,Total     : integer;
Answer          :Char;
Day             :DayType;
Employee        :EmployeeType;
Work            :WorkType;
WageSupplement : Real;

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

Procedure DayOutPut;

{  One of the problems with variables which are of a
   user-defined data type (like Day) is that Pascal
   will not allow you to Read or Write them. To
   overcome this, procedures like these next two are
   useful. They can be called from anywhere within
   the program and effectively write the contents of
   the variable to the screen.  }

Begin;
   Case Day of
      Mon  : Write('Monday');
      Tue  : Write('Tuesday');
      Wed  : Write('Wednesday');
      Thu  : Write('Thursday');
      Fri  : Write('Friday');
      Sat  : Write('Saturday');
      Sun  : Write('Sunday');
   End;
End;    {  Of Procedure DayOutPut;  }

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

Procedure EmployeeOutPut;
Begin;
   Case Employee of
      Teaboy   : Write('Teaboy');
      Mechanic : Write('Mechanic');
      Foreman  : Write('Foreman');
      Gaffer   : Write('Gaffer');
   End;
End;    {  Of Procedure EmployeeOutPut  }

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

Procedure SortAnswer;

{  Again, a problem with I/O involving defined data
   types. We can't Read into the variable Day. So we
   Read into the Char variable called Answer, and
   then this procedure effectively converts this to
   the correct value in Day. If this all strikes you
   as unwieldy, then I can only agree. This is a
   waste of time in small programs, but useful in
   large ones.  }

Begin;
   Case Answer of
    '1' : Day := Mon;
    '2' : Day := Tue;
    '3' : Day := Wed;
    '4' : Day := Thu;
    '5' : Day := Fri;
    '6' : Day := Sat;
    '7' : Day := Sun;
   End;
End;    {  Of Procedure SortAnswer  }

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

Procedure Menu;

{  Allows the user to select a day of the week in
   which some work was performed. A real program
   would then allow the user to pick an employee and
   the job which they performed. Since this is
   simply a demo. I haven't included those menu
   procedures; they would be very similar to this
   one. Instead, I have cheated and assigned values
   to the remaining variables at the end of this
   procedure.  }

Begin;
   ClrScr;
   GotoXY(5,5);
   Writeln(Greeting);
   Writeln('Please choose a number');

   Day := Mon;          {  Starts Day off on Mon  }
   For Count := 1 to 7 do
   Begin;
      Write(Count,'...');
      DayOutPut;
      Writeln;
      Day := Succ(day); {  Sets Day to the next day
                           in the sequence. More on
                           this next time.  }
   End;

{  This next bit only allows the program to proceed
   if the user has entered a number between 1 and 7.
   Good isn't it? Not original of course, all the
   best code is borrowed.  }

   Repeat
      Read(Kbd,Answer);
   Until Answer in ['1'..'7'];

SortAnswer;             {  Calls up the procedure
                           SortAnswer to set Day to
                           the user's choice.  }

{  These two lines are the cheat. A real version of
   the program would allow the user to select
   these.  }

Employee := TeaBoy;
Work := CleaningUp;

End;    {  Of Procedure Menu  }

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

Procedure SortOutPay;

{  This is really the important bit. You will have
   noticed that all of the rest of the program seems
   inelegant. This is where the user defined data
   type comes into its own. This section in now much
   more readable. In a full-size program, this
   section would be much larger and would deal with
   lots of different combinations of the factors
   which control extra pay.  }

Begin;
   WageSupplement := 0;

   If (Day in [Mon..Fri]) and (Employee = Teaboy)
   and (Work = CleaningUp)
      then WageSupplement := 3.63;
   If (Day = Sat) and (Employee = Teaboy) and (Work
   = CleaningUp)
      then WageSupplement := 5.50;
   If (Day = Sun) and (Employee = Teaboy) and (Work
   = CleaningUp)
      then WageSupplement := 9.99;

   Writeln('Extra for wages = ',WageSupplement:5:2);

End;    {  Of Procedure SortOutPay  }

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

Begin;
Menu;
SortOutPay;
End.
```

Archive your files

and save disk space

By Guy Sturges

The basic objective of an archival program is to reduce the amount of space used on a disk by files. The storage space of an MS-DOS floppy or hard disk is split up into what is known as 'sectors'. Each sector is a fixed size as defined by the disk type — sizes of 128, 256 or 512 bytes are the most common for Apricot disks. To gain improvements in accessing data a unit called a 'cluster' was introduced by Microsoft; a cluster contains a fixed number of sectors, again as defined for specific disk types, and cluster size can vary from 1 to 16.

Any file, no matter how big it is, must take up at least one cluster. So you can imagine the amount of space that would be wasted by small files on a hard disk which has 16 sectors per cluster. Take a batch file of 405 bytes: it will require 16 sectors of disk space, each of which is 512 bytes long, and that small file will therefore take up 8,192 bytes. The unused space in this cluster cannot generally be used for anything else.

This is where one of the space-savings of the archival programs comes into play. It makes use of that wasted space by packing all the files it archives into one big file; this alone could have quite a dramatic effect on a series of small files — for instance, 21 files of 378 bytes would occupy 168KB if stored normally, archived as one file they use only 8K.

The savings are much less on a floppy disk, where the sector size may be smaller and the number of sectors per cluster is generally 1 or 2. Taking the same example on a floppy of two 512-byte sectors per cluster, 21 files of 378 bytes would use 21KB while in one file they would take 8K.

Compression

The saving on space is still useful; but as the files get bigger, the value of archiving obviously decreases. So the clever people came up with compression techniques, sometimes called 'squashing', 'squeezing' or 'crunching'; these reduce the amount of data saved in the archived file without losing any information.

Several compression techniques can be used, so the archiving program will first read the source file to decide which method to use. When it has established that, the archiver saves the compressed file in the archival file.

I recently acquired the archiving programs ARC version 5.20 and PKARC version 3.5 both of which can be found on DEALS FOR READERS disk AP021. I wondered which was better, so I set about testing them with a large file (a worst case for space-saving, as seen above) and got the following results:

Archive utility	Original length	Compression method	Size now	Ratio
ARC 5.20	97,979	crunched	29,664	7Ø%
PKARC 3.5	97,979	squashed	28,Ø21	72%

As for saving of disk space, the physical storage requirement reduces from 96KB to 32K in both cases for a disk with a cluster size of 16 sectors each of 512 bytes.

ARC versus PKARC

So much for the method; but other factors need to be taken into account — the most important to my mind being the time taken to archive and de-archive a file.

- PKARC works faster than ARC by a factor of up to 10.
- PKARC generally produces archive files about four per cent smaller than ARC's.
- Usefully, PKARC allows you to add comments (during or after the archiving) to files being archived. There is no way to do this with a straight MS-DOS file.
- PKARC also provides the possibility of creating a single file that when executed expands itself into individual files — thereby permitting more than 315K or 720K of information to be stored on one disk, including the extraction and compression program itself.
- PKARC can handle files produced by ARC, but not necessarily *vice versa*.
- By default, PKARC date-stamps the archive file with the current date and time; this I prefer to ARC's use of the most recent date and time of the files in it.
- ARC does have facilities that PKARC lacks, however. In particular, it can merge two archive files — providing you have the MARC program, which is not on AP021.

Both programs can encrypt your data. This could be of use if you wish to transmit confidential data or send it by post without it being of any use to anyone except the person to whom it was destined. But if you lose the key, there is no way of recovering the data.

In general it seems that PKARC is an upwards-compatible improvement on ARC. The files it creates, which are smaller than ARC's, can't be processed with ARC; many of the files created with PKARC use the same algorithms as ARC and are therefore the same size and compatible. As for ARC's ability to merge two archived files, this can still be done by 'extracting' from the original archive and recompressing into the new one. ◆

The Q&A Clinic:

Questions by subscribers, answers by APRICOT FILE

Between us, the staff of APRICOT FILE and our contributors offer an unusually wide range of expertise and experience. As many subscribers will know, we handle a lot of queries each month by mail or by phone; and every so often we run the Q&A page to present what we feel are the more interesting and/or useful of these. If you want to make use of the service, just drop us a line ...

Q. XCOPY switches

Could you please print a list of the switches for the MS-DOS 3 utility XCOPY.

A. XCOPY, which is provided with MS-DOS 3.2 and 3.3, is a very superior form of the COPY command. It's primarily designed for moving an entire directory structure (subdirectories and all) from one disk to another; it will create directories on the target disk automatically if they don't exist already.

XCOPY can be pretty powerful in selective copying, too; it may have a total of ten parameters. The syntax is XCOPY followed by one or more of these:

```
sourcename
targetname
/A
/M
/D:date
/S
/E
/P
/V
/W
```

- **Sourcename** is the only mandatory parameter — obviously it's the name of the file(s) to be copied. This can include standard wildcards, pathnames, and drive identifiers; it can also be simply a drivename, in which case everything on the disk will be copied.
- **Targetname**, equally obviously, is the destination. Again, it can be any valid combination of drive identifier, pathname, and file name. If you omit the target, XCOPY will copy to the default directory.
- **/A** copies file(s) only if the archive bit is set — this flag is applied automatically by MS-DOS if a file has been changed since it was last copied, so it can be used to screen out files that don't need to be copied this time around. The /A switch doesn't reset the archive flag on the new copy
- **/M** copies file(s) that have the archive bit set, and it does reset the archive indicator
- **/D:date** copies file(s) that have the date given or a later one. The style of *date* should correspond to the datestamp in a DIRectory — so if your datestamps have the format *dd-mm-yy*, use that style with the **/D** switch
- **/S** automatically sets up corresponding subdirectories on the target disk
- **/E** sets up subdirectories on the target even if it finds nothing in the original subdirectory on *source*
- **/P** prompts for confirmation before each file is copied
- **/V** verifies the copy, just as with the equivalent COPY switch
- **/W** prompts you to insert a disk before copying — useful if you're using XCOPY in a batch file to automate a backup of several floppies.

Q. Basic books

Can you recommend a book on MS Basic that is more understandable than the manual provided with the Apricot?

A. If you can get hold of a copy, the manual shipped with the Sirius is good — so for that matter is the MSBASIC compiler's documentation. There are also several books on BASIC for IBM-compatibles, and judicious reading between the lines (or ignoring some of them altogether) means they can be used with the Apricot. Of this lot, we recommend two by David Lien — *Learning IBM Basic* for starters (£17.95) and *IBM Basic Handbook* for reference. (*The* APRICOT FILE DEALS FOR READERS *service can obtain these on request, and indeed just about any computer-related book — Ed.*)

Q. Apricot to AT links

Are to able to advise me about making up a cable for file transfer between Apricots and an IBM AT?

A. The AT and many clones are fitted with a nine-pin DB-9 connector for RS-232 communications, rather than the more common wide-mouthed DB-25 as used on the Apricot. The DB9 carries what are usually RS-232-C pins 2-8, 20, and 22. The table following shows the pin assignments and their equivalents from the 25-pin setup.

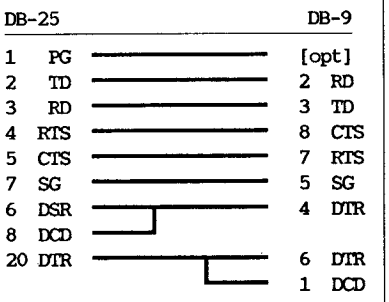
AT-type DB-9 connector

DB-9		DB-25
1	DCD	8
2	RD	3
3	TD	2
4	DTR	20
5	SG	7
6	DSR	6
7	RTS	4
8	CTS	5
9	RI	22

One option is to buy an AT-to-RS-232 converter; this is the route we opted for with the OMEGA AT that we sell as an Apricot upgrade. The converter is a short length of cable with a nine-pin connector on one end and the standard 25-pin socket on the other, so you can use a standard 25-way null modem cable with it. The APRICOT FILE DEALS FOR READERS service's price is £22, order no CC189.

For D-I-Y use, here is a diagram for a null modem cable to connect the old with the new (or *vice versa*). Pin 1 — Protective Ground — is optional and can go to any bolt attached to the frame. This sort of cable is less readily available from suppliers than the 'straight-through' DB-25-to-DB-25 sort, though it should be easier to construct yourself.

Null modem cable: DB-25 to DB-9



It is worth speculating on why IBM adopted such a tiny connector. Perhaps the best reason is the size itself, permitting a greater number of serial ports to fit on the back of the micro. This is mostly an important consideration for those who care to transform an AT-type into a multi-user machine — in which case the more ports the better (to a point).

Aside from this less-is-more approach, it has to be a great saving to use DB-9s instead of DB-25s. And then, of course, all those special cables to connect DB-9s and DB-25 devices (most of the rest of the world) need to be sold to the AT masses. Get out the accountants!

And why did IBM chose that particular selection of pins? Well, the only slight surprise is the loss of Protective Ground (PG, pin 1 from the 'standard') in favour of Ring Indicator (RI, pin 22 from the 'standard'). The trade-off is not too bad, though; PG is not often needed, and RI can be very useful with modems.

Of course the PG pin does have its place. It is mostly used as a matter of safety, in case there's a short in a power supply or some such. Basically, because it is earthed it could harmlessly carry away any potentially dangerous voltages that might arise from some major internal electrical fault. This is an unusual occurrence, however.

But there are a few other unusual considerations, such as when certain pairs of devices are connected together and only one has its frame electrically earthed. In a few rare instances, connecting the PGs from each is a necessity for the rest of the comms to operate. Even less common is a condition called a ground (or earth) loop, where the presence of connected PGs could actually hinder communications.

It is important to emphasise that these are not usual situations, especially with the state of equipment manufacture today; the overwhelming majority of PC serial comms will be fine without a PG. The purpose of mentioning these rareties is to point out that PG has not always been, nor is yet, a useless pin. Besides, even if it is absolutely needed, the cables can always be earthed through a wire to each chassis.

As for the RI line, this can be used to control the autoanswer facility found on many 'smart' modems. The modem will detect the ringing of the telephone line, send a signal on the RI pin to the micro (if RI is connected), and signal the micro/comms software to let the modem answer the call. This is primarily useful when you want a system to receive the call automatically from another micro's modem — such as for file transfers, or when operating a BBS — where modems calls are expected and need to be received in an automated fashion.

If your system has this autoanswer capability, and you are experimenting with it, be sure to disable it when it's not needed. This is safer from the viewpoints of your system and of outside callers. There are situations where it is possible for an outside modem caller to 'break in' to your modem and access any phone numbers stored there (if your modem has that capability), and possibly work some mayhem with the modem commands (especially if the unit uses some common command set, such as Hayes').

The other unpleasant possibility is when a 'live' caller rings to be greeted by the very annoying squeal of your answering modem. The easiest solution to both potential bothers is simply to disable the modem from its autoanswer mode, for instance through the comms software; or you could just turn off the power on external units.

Q. Extended memory

Do you know of any accelerator boards or add-in extended memory that will take the Apricot above one megabyte? I want to use the extra RAM for Symphony and/or RAMdisks.

A. As far as we know, it isn't possible to utilise more than the MS-DOS maximum of 896KB on the Apricot.

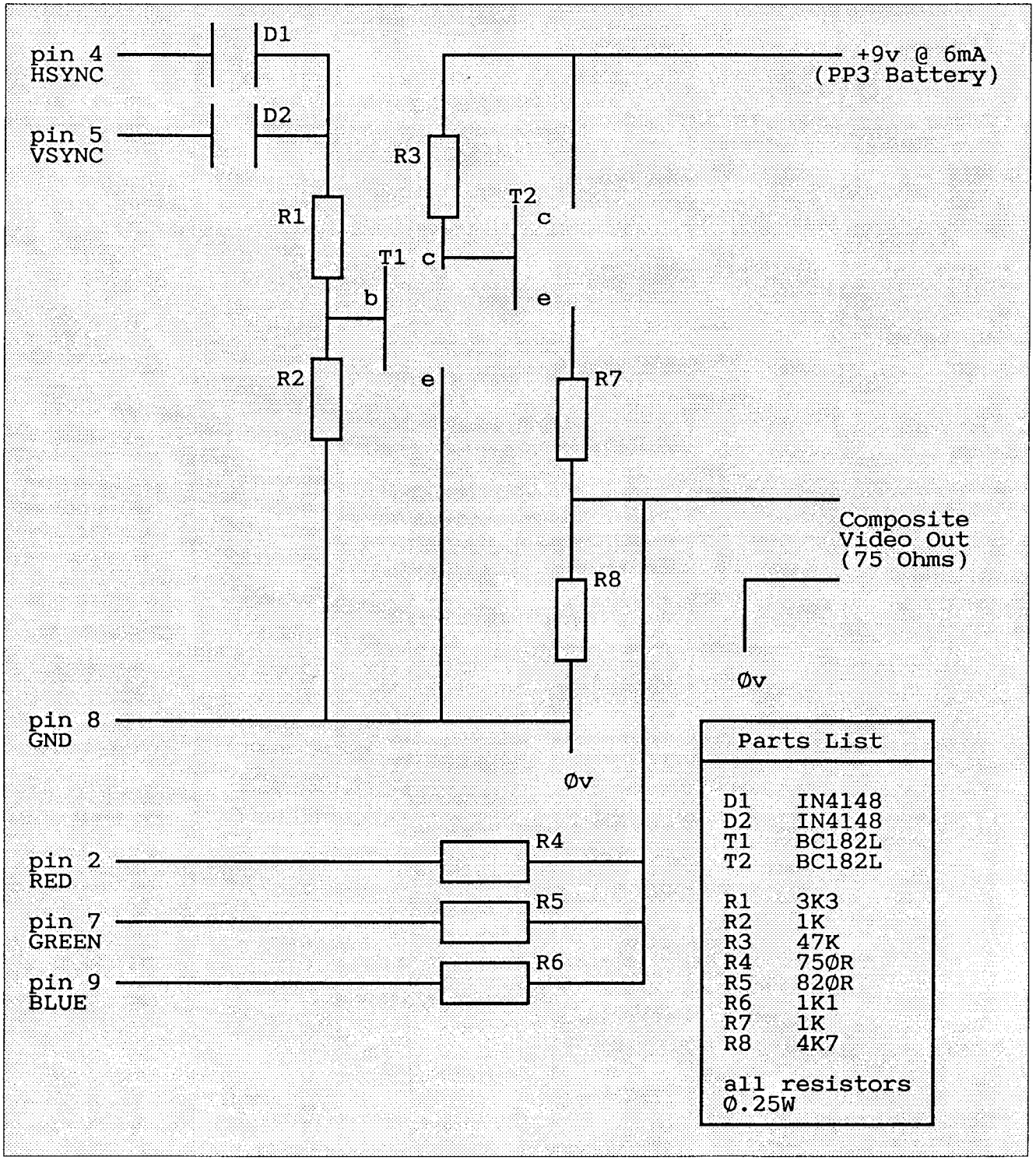
Q. Portable colour

Can you tell me how I can connect my Portable to a Phillips composite video monitor? Do I need a special connection?

A. You need an RGB-to-composite converter; **Fraser Associates** on **0280 816087** sells such an animal, or (courtesy of Andrew Bessant of the TRINITY 2 bulletin board — 0367 81507) you could make one yourself. It costs only a few pence to build and will even give a (monochrome-only) picture on a TV set via a video recorder. If you have any problems, Andy's own company **Canidae Electronics** can supply any parts — including a PCB, a complete kit and/or the ready-made unit. Canidae is at Blueberry House, Clanfield, Oxford OX8 2SR.

Pin	Use
1	+12V [but see below]
2	RED
3	intensity
4	horizontal sync
5	vertical sync
6	frame ground
7	GREEN
8	0v (GND)
9	BLUE

On some Portables, pin 1 isn't connected to +12V; to connect it to +12V a wire link is needed in the place where resistor R7 should go on the display PCB.



Parts List	
D1	IN4148
D2	IN4148
T1	BC182L
T2	BC182L
R1	3K3
R2	1K
R3	47K
R4	75ØR
R5	82ØR
R6	1K1
R7	1K
R8	4K7
all resistors Ø.25W	

All buttoned up

Practical experience with PC-FILE/R

By Ted Webber

Jim Button's initial series of PC-FILE database programs were available as 'shareware': you paid if you wanted to.

Why would anyone but a saint pay if not absolutely obliged to? Well, actually it is a good sales gimmick, which lets you have the program with only rudimentary documentation as an 'evaluation copy'. This way, you can find out whether the program is likely to meet your requirements, at negligible promotional cost to the author or the user.

I obtained my copy of PC-FILE III from a Gateway PD disk; it's also available from Apri-Soft-Exchange and from APRICOT FILE'S DEALS FOR READERS service. The package included an IBM emulator, EMIBM.EXE version 1.3.2. The documentation provided on the disk was quite enough to get my address lists whizzing, and it certainly persuaded me to lay out the mere $49 asked for the full documentation from the USA.

But PC-FILE III was the last shareware/public domain version. The manual for the next upgrade, PC-FILE/R, states "this is *not* user-supported software like PC-FILE III ... You are not allowed to copy and share this program".

IBM emulation

I duly sent off $75 (the price has increased since 1984), stating explicitly that I had successfully run PC-FILE/III on a non-IBM compatible with *3½in* floppies, and mainly wanted the complete documentation. Within a couple of weeks, PC-FILE+ landed in my post box: a nice paperback book, listing lots of lovely enhancements, and two *5¼in* disks!

Buttonware also offered a further enhancement (a debug?) if I sent another $15. This I did, again stressing the superior disks on the Apricot. This time they got it right ... and I found that Jim Button had dropped the previous policy of catering for non-IBM owners. PC-FILE+ will not load on an Apricot: it claims to be tremendously faster than previous versions, but no longer has a PCSETUP.EXE with which to configure your system.

Never despair! I returned everything Buttonware had so far despatched, on the promise of something between PC-FILE III and PC-FILE+ called PC-FILE/R plus an undefined credit. This again arrived by return of post, *and* in 3.5in format, would you believe. What frabjous joy when, after running PCSETUP and EMIBM, PC-FILE/R loaded and ran with the same data files I had set up under PC-FILE III.

Before reading any further, you probably want to get some idea of what PC-FILE/R can do. I will outline this very broadly, enough so you can see whether it is worth paying for the PC-FILE III evaluation disk.

PC-FILE/R in summary

Fields per record	max 42 (like columns in spreadsheet)
Records per database	max 32,767 (like rows)
Relational	read-only: input from other file
Editor	mini word processor
Calculations	limited to fields of current record, plus column totals
Help	pop-up screens
Search	fast binary on sorted field; four types of sequential search
Sort	on any field or combination of fields. Roman numerals supported!
Data transfer	import/export from/to almost any format used by other programs
Path	supports subdirectories
Macros	12 programmable 'smart keys'
Minimum hardware	128KB, MS-DOS 2+, one 360KB floppy drive
Options	IBM, ROM BIOS, or ANSI.SYS
Apricot requirement	BIOS 3+ (for implementation of ALT key)

Setting up the keyboard

It goes without saying that Buttonware does not supply an Apricot keyboard to emulate IBM; the main modules of the program can be operated without keyboard emulation, however. For instance, you can type in 'ADD' instead of pressing F1 to add a record to the database.

But several useful features are not accessible without a (software-) modified keyboard. These include the help screens, the smart keys, and the word processing functions, all of which are intended to be accessed using an ALT key combination. More important, the IBM emulator requires different coding of the cursor keys.

The PCFILE.KB keyboard supplied by Gateway/Apri-Soft with the evaluation copy of PC-FILE is not very helpful. ALTIBM.KB on DEALS FOR READERS disk AP027 sets the cursor keys correctly; but it puts the ALT key on HELP, where F1 is needed, and leaves the ALT mode keyboard empty — there's still no effect of ALT-H when a help screen is needed. So I had to prepare a keyboard specially, and for this I consider EDITKB on AP027 is just about essential.

I favour ALT configured on the STOP key, because STOP is redundant and even downright dangerous. But the action of ALT from the STOP key seems to be temperamental, at least on the Portable — sometimes it works, sometimes it does not. So I had to put it on the ENTER key, just as EDITKB does for itself.

I also found that ALT toggles a lock on the emulating keyboard: you have to press ALT, then the key or keys in ALT mode, then press ALT again

to return to normal keyboard mode. This is inconvenient: there is no LED to indicate whether ALT is on or off.

A solution may be to put the ALT functions on the CONTROL keys, which are largely unused by PC-FILE. For instance, ALT-H is coded as ^@# (the NUL symbol ^@ is obtained in EDITKB by ENTER-000). If this is put on CONTROL-H, you obviously lose ^H. But ^H is duplicated on the backspace key. In fact, the only item of the CONTROL keys which PC-FILE needs is the tilde (CONTROL-9), and this could be moved to CONTROL-F8.

So I have sent APRICOT FILE two versions of the keyboard file — PCFILERA.KB emulates the program's keyboard allocations as closely as possible, using ALT: PCFILERC.KB uses CONTROL instead of ALT. To complete this bit of botching, I used DEBUG to change ALT text references to CONTROL in PCFILE.EXE and PCFILE.HLP. *(Ted's keyboards are now on our AP015 disk; see the box at the end of the article — Ed.)*

There is one other problem with the ALT keyboard. According to EDITKB, each ALT key takes up four bytes of space in the keyboard file. So it is not possible to load *all* the ALT keys (whether in ALT or CONTROL mode), and it is necessary to select which functions are essential and which of marginal utility. I opted to omit T, S and G out of the ALT-shifted keys nominated for PC-FILE in LETter mode — ALT-T changes all text to 'tall' (upper case); ALT-S changes to 'small' (lower case); and ALT-G gives access to graphic characters. All of these are still obtainable via ALT-shifts using the numeric pad (the relevant numbers are partially listed in the PC-FILE documentation for the smart keys).

Extended ASCII with the IBM emulation

Code	^@ +	Effect
259	CONTROL-C	NUL
271	CONTROL-O	Shift TAB
272	CONTROL-P	ALT-Q
273	CONTROL-Q	ALT-W
274	CONTROL-R	ALT-E
275	CONTROL-S	ALT-R
276	CONTROL-T	ALT-T
277	CONTROL-U	ALT-Y
278	CONTROL-V	ALT-U
279	CONTROL-W	ALT-I
280	CONTROL-X	ALT-O
281	CONTROL-Y	ALT-P
286	[30]	ALT-A
287	[31]	ALT-S
288	Space	ALT-D
289	!	ALT-F
290	"	ALT-G
291	#	ALT-H
292	$	ALT-J
293	[37]	ALT-K
294	&	ALT-L
300	'	ALT-Z
301	—	ALT-X
302	.	ALT-C
303	/	ALT-V
304	0	ALT-B
305	1	ALT-N
306	2	ALT-M
315	;	Function key F1
316	<	Function key F2
317	=	Function key F3
318	>	Function key F4
319	?	Function key F5
320	@	Function key F6
321	A	Function key F7
322	B	Function key F8
323	C	Function key F9
324	D	Function key F10
327	G	Home
328	H	Cursor Up
329	I	PgUp
331	K	Cursor Left
333	M	Cursor Right
335	O	End
336	P	Cursor Down
337	Q	PgDn
338	R	Ins
339	S	Del
340	T	Shift F1

Code	^@ +	Effect
341	U	Shift F2
342	V	Shift F3
343	W	Shift F4
344	X	Shift F5
345	Y	Shift F6
346	Z	Shift F7
347	[	Shift F8
348	\	Shift F9
349	]	Shift F10
350	^	CONTROL F1
351	_	CONTROL F2
352	`	CONTROL F3
353	a	CONTROL F4
354	b	CONTROL F5
355	c	CONTROL F6
356	d	CONTROL F7
357	e	CONTROL F8
358	f	CONTROL F9
359	g	CONTROL F10
360	h	ALT F1
361	i	ALT F2
362	j	ALT F3
363	k	ALT F4
364	l	ALT F5
365	m	ALT F6
366	n	ALT F7
367	o	ALT F8
368	p	ALT F9
369	q	ALT F10
370	r	CONTROL PrtSc
371	s	Previous Word
372	t	Next Word
373	u	CONTROL END
374	v	CONTROL PgDn
375	w	CONTROL Home
376	x	ALT 1
377	y	ALT 2
378	z	ALT 3
379	{	ALT 4
380	|	ALT 5
381	}	ALT 6
382	~	ALT 7
383		ALT 8
384	ç	ALT 9
385	ü	ALT 0
386	é	ALT 1
387	a	ALT =
388	[132]	CONTROL PgUp

The panel gives a full list of the extended ASCII codes used with IBM emulation. The lefthand column is the same as the 'keystroke code for smart keys' listed on page 74 of the PC-FILE/R manual, and is the ASCII code of the middle column plus 256.

The ALT-shifted number-pad keys do not recognise the extended codes directly; so to set up say smart key 1 in EDITKB, for activation by ALT-1, toggle to the ALT screen; move the cursor to the box for the 1/! key; hold ENTER and press 000 on the numeric pad (^@ should appear); and (still holding ENTER) press 120 on the number pad. You should now see ^@x in the key box.

PCFILEA.KB

My installation of PC-FILE is on an Apricot Portable, but is equally appropriate on any other Apricot with a suitable IBM emulator. (If you normally use Ansible's FLASH.COM on a Portable, delete it before going to EMIBM unless you want the screen filled with broad arrows.)

The Gateway EMIBM also works on an Xi. Its KB files prepared do not utilise the MicroScreen, however, and if you have a machine with a MicroScreen you may want to insert something there so that the calculator is active. See APRICOT FILE Release 3.2 for more on this, and check out DEALS FOR READERS disk AP027.

PCFILEA.KB has the special key assignments shown in the panel. They may be seen more graphically in the EDITKB display, which can be recorded with MXDUMP (if you have a non-F series machine).

IBM	Function keys	Apricot
1		HELP
2		UNDO
3		REPEAT
4		CALC
6		PRINT
7		INTR
8		MENU
9		FINISH
5 (FIN)		Shift-MENU
	(default on PC-FILE main menu)	
10 (^A)	Shift-FINISH or (unshifted) CLEAR	
ALT 1		ALT-HELP
ALT 2		ALT-UNDO
ALT 3		ALT-REPEAT
ALT 4		ALT-CALC
ALT 5		ALT-PRINT
ALT 9		ALT-FINISH
ALT on/off		ENTER toggles
ALT number pad	functional for any ASCII code	
PgDn		SCROLL
END		CONTROL-HOME
PgUp		CONTROL-SCROLL

ALT keys implemented	
Smart keys	1 to =
Numeric Pad	
R, U, D, H, J, L, C, V, M	

To understand how these keys are implemented, refer to the table of extended ASCII codes above. All other keys have the same function, whether IBM or Apricot — though do not all carry the same code (the cursor keys, for instance).

PCFILEA.KB has been RINSTALLed with PC-FILE in LODPCFIL.COM, for speedy loading and restoration of keyboards. Hence my batch file to load PC-FILE/R contains these essential lines:

```
ECHO OFF
EMIBM
LODPCFIL
EMIBM NO
```

Profile files

PCSETUP.EXE produces 'profile files', those with the extension .PRO. The manual explains how to obtain a profile file for each set of database files. For instance, your address list can probably be displayed adequately in 80 columns, plus 'snapshot' mailing labels; a stocklist, on the other hand, may require 250 columns of condensed printing on 15in paper. These variations can be automated by the PROfile, or by equivalent instructions on the command line.

I have found one or two undocumented features of PROfiles. The first is that although PCSETUP does not ask whether you want European or US-style dates, you can edit your PROfile (with EDLIN, say) and insert an extra line reading /EUROPE. Similarly, /KEYIN and /END could also be inserted to control the beginning and end of processing.

In fact, PCSETUP can be bypassed completely and PC-FILE can be run more rapidly by using a text editor to write the PROfile directly; refer to the Configuration Commands summary on page 100 of the manual for details.

You can see the possibilities from this typical PROfile (don't copy the comments!):

```
/FG,7,/BG,0        Foreground and background for
                   monochrome
/ROMBIOS           Apricot-compatible
/FASTADD           Use buffers to speed data
                   entry
/FASTANSWER        For single keystroke commands
/FORMFEED          Printer accepts formfeeds
/LINES, 60         Print 60 lines per page
/PAGELEN, 66       66 lines between perforations
/NORMAL,27,80      Print Pica
/CONDENSED,15      Print condensed
/PAGEWIDTH, 150    80-column printer, condensed
/DRIVE,A           Look for data on A
/PATH,\DATA        Look for data in A:\DATA
/EUROPE            Print dates as DD/MM/YYYY
```

The most important question asked by PCSETUP concerns the type of computer — Apricot users should answer 'ROMBIOS' (option 2).

Note that if you do not want to use a subdirectory for data, you will be saved some irritating questions if the relevant line reads

```
/PATH,\
```

File configuration

Here I am going to continue to write about the Portable, on the grounds that if something works on a low-grade machine it will usually work even better on an Xi or a Xen. Indeed, I have run PC-FILE on an Xi.

Appendix D of the PC-FILE manual lists all the files Jim Button puts on his disk and tells you which ones are not essential. For instance, PCFILE.PIF is superfluous unless you have the IBM-specific TopView front-end. Weeding out useless files will give faster access to your data. I create a 350KB RAMdisk and put in it the following files:

AUTOEXEC.BAT	with EMIBM and LODPCFIL in it, as above
COMMAND.COM	
LODPCFIL.COM	including PCFILERA.KB
EMIBM.EXE	
PCDEF.EXE	
PCEXPOR.EXE	
PCFILE.EXE	the main program
PCIMPOR.EXE	
PCLTR.EXE	the word processor files: see below
PCLTR0.EXE	
PCLTR1.EXE	
PCOVL.EXE	
PCPRINT.EXE	
PCSORT.EXE	
PCUTIL.EXE	
PCFILE.HLP	
PCFILE.PRO	the default PROfile: others are kept with data

These leave 33,280 bytes free on the RAMdisk, enough space to bring in PCSETUP.EXE or some other utility if required. If you have a machine with more than one drive (and plenty of memory) it is still worthwhile putting the program files into RAMdisk so that the overlays load quickly.

The word processor

Entering 'LET' or ALT-F5 (or rather, ALT-F6 on my PCFILE*.KB: F5 is key 97, and cannot easily be coded on the 96-key layout assumed by EDITKB) loads PCLTR.

The screen comes up just as shown in the manual, and you can enter text. But none of the function keys designated for letter mode work! F1, for instance, just produces a semicolon instead of displaying a HELP screen. Worse, the program depends on a function key to save the work and exit: so you're totally stuck. The problem must be in the IBM emulation. Anyway, this is a subsidiary feature of the program, and I no longer load the LTR programs.

Database reports

With the exception of LIST (output), all functions of the program worked with very little fuss — and most worked with single keystrokes.

But printing of reports has a wide variety of options, even after an output format has been defined and saved. So here is one place where smart keys do need to be set up. The output format file (with the suffix .RPT) does not record the selection of records to print, so this may also need to go into a smart-key definition.

One small quirk: if totals or subtotals of fields are printed, it is not easy to stop averages *also* being printed. But hidden away in the instructions for LIStings you find that this can be achieved by putting [ST] after relevant fields in the .RPT file generated by 'Field to list?'. [ST] means 'print subtotals and totals only'; the default is [STA].

PC-FILE/R applications

My requirements for databases are mainly of the elementary 'find an address' or 'sort employees by location' type. PC-FILE is ideal for this kind of application, and there is no scary compilation as required by dBASE or DATAFLEX. For example, I had a list of some 900 public domain software files, compiled using SuperWriter with fields FILENAME, XTN, DESCRIPTION, DISK NO. and CLASS. SuperWriter's 'find' routines were sufficient to locate a file; but I wanted sorted listings, so I 'imported' the SuperWriter data file to PC-FILE/R.

But is my application best suited to database or spreadsheet? That is a question which must often arise.

The two types of program have many similar features, but with a difference of emphasis. Spreadsheets can have 'lookup tables', used mainly for discontinuous calculations like PAYE. They are usually not so good as a database when searching for a record — you really need to know where to look in the first place!

PC-FILE can do simple calculations; SuperCalc can do much more complex ones. PC-FILE/R can have 32,767 records and 42 fields; SuperCalc 1, which provides a similar level of sophistication, can have only 254 rows but up to 64 columns. (PC-FILE+ has a maximum of 70 fields).

A database is much less tolerant of layout changes than a spreadsheet: you must plan the fields and their widths carefully for PC-FILE, whereas SuperCalc allows you to add or delete columns, or change their widths, at will. To do this kind of thing in PC-FILE requires the contortions of 'cloning' the database, which basically means copying to another file name and creating a new HDR file with PCDEF. It pays to get the field definitions right the first time.

TIMSHEET.DTA

Having been highly satisfied with PC-FILE's capability for listing, searching and sorting, I tried it with a previously-modelled spreadsheet application for the analysis of timesheets — see my Workshop in APRICOT FILE Release 3.7.

I immediately came up against the limitation in number of fields or columns. Using Lotus 1-2-3, I put hourly rates for each member of staff in one row, and the hours spent on each project in multiple rows below. Thus one column was sufficient for one staff member, and the limitation on number of staff conveniently logged was only the width of the printout.

But in PC-FILE I found I had to replicate the hourly rate in every record, so that each member of staff needed at least two fields. With four fields allocated to Project Name, Client, Period Total, and Cumulative Total, this meant I could fit in only 19 staff in total (4+19x2=42). If I want to fetch the rate from an Employee database, I need a *third* field in each record for employee name.

Spreadsheets refer to their data by cell coordinates: D21 is the cell at the intersection of column D and row 21. PC-FILE uses the field name for the 'column', and the first field name in each record as the 'key' to the record or 'row'.

At first sight, this looks much more long-winded. But you don't have to give the entire name, only enough of it to be unique in a sequential search. So life is made much easier if every field starts with a different character — or, failing that, if the first two characters are unique. Fields named 'TARGET 1988' and 'TARGET 1989', though legitimate, are no-nos: use '88 TARGET' and '89 TARGET', and save a lot of typing when specifying 'field to print' and the like.

For my timesheet problem, I labelled my fields with 42 different first characters. That meant only one character was needed to identify each field in the calculation. For example, sequential fields were 'A Smith', 'B Hours', 'C Jones', 'D Hours' ... and so on. In 'A Smith' went Smith's rate, using the 'automatic data' edit mask; and in 'B Hours' went the number of hours Smith spent on each project (record) for the current period.

Also in the edit mask for the Period Total field went (A*B+C*D+ ...), taking up the minimum space in the mask. And that is where I hit the next limitation: there was space in the edit mask for only about ten of those 'A*B+' sequences. So it was necessary to have two bites at the cherry, using a subtotal field — and losing another staff field.

Next, it appears it is necessary to divide my staff into groups not greater than 18. That means separate databases for the Pipeline Design Section, the Treatment Works Section, and so forth. It's feasible. Yet another database could summarise the section databases — one section per field, plus project name, client, and total.

There remains the problem of accumulating totals from previous periods, which is probably surmountable.

The invoice 'report'

The end-product would be a set of invoices. It would be easy to generate a report format for printing which selects only those 'hour' fields containing non-zeros, with associated staff and rates; and to select only the projects for specified clients.

Compare this with a spreadsheet, where one normally has to specify a fixed 'range' of cells to print. Note that PC-FILE/R can read Lotus .WKS files too, so another route might be to use PC-FILE/R to generate invoices from TIMSHEET.WKS.

It must be obvious that I have not actually got that far yet. But this lengthy description serves to illustrate the preparation phase necessary, using plain old paper and ink, before going anywhere near a computer — unless you want to try the 'Thought Processor' described in the PC-FILE/R manual!

Last words

Such an application clearly pushes PC-FILE/R close to its limits, and perhaps beyond them. Instead of struggling to make it fit, it would be far wiser to get more powerful software (maybe better hardware too).

But this is no contra-recommendation: within its limits, PC-FILE/R performs extremely well. Furthermore, it is cheap, and easy to get to grips with. ◆

AP015 Updated

We have updated the current shareware PC-FILE III disk in the DEALS FOR READERS lists to incorporate a heap of goodies that Ted Webber sent with this article — including the improved IBM-emulating keyboard files and a sample database. Anyone who has bought AP015 is welcome to a free upgrade — just send us the original disk and a stamped self-addressed envelope.

Note that the additions have taken AP015 beyond the capacity of a single-sided disk; if you don't have double-sided drives, send a second blank disk when you return your AP015 original.

We strongly recommend that you support the shareware principle and register with Buttonware if you have PC-FILE III. For that, and for more information on the non-shareware PC-FILE/R, the address is:

Buttonware Inc
PO Box 5786
Bellevue, WA 98006
USA

Buttonware accepts Mastercard (*aka* Access, Carte Bleue) or Visa (same as Barclaycard, TSB Trustcard and others).

File o' Facts

Cries and whispers

Notes from the working week (1)

By Eddie Bromhead

While raiding a friend's F2 for any useful utilities, I came across two files which I hadn't seen before: FECG.EXE and IBMINT17.EXE.

Swiftly applying DEBUG to the former showed it to be a precursor to 640200EG.EXE as supplied widely (but not understood) with BIOS 3.1 systems. IBMINT17 was obviously a way to intercept the IBM PC BIOS interrupt 17 hex, the printer driver. Why should you want to do that?

IBMINT17

To start with, why *shouldn't* you want to do that? Well, no matter how slow your computer, it runs miles faster than any printer that a PC owner can afford. Indeed, Ray Duncan in his *Advanced MS-DOS* writes: "Use of the ROM BIOS printer services is rarely necessary or justifiable when writing programs to run under MS-DOS". Not only that, using an IBM PC ROM BIOS service makes your software incompatible with Apricots. Only in extreme cases, such as the output of bit image data, is every millisecond of saved time valuable.

Hence, the answer is simple — it must be something to do with printer graphics dumps; and it is. Any graphics dump written originally for the IBM PC probably *does* use INT 17H, and that includes GSX and GEM. Either the Black Hole GSX printer drivers have been patched by Apricot, or they are so early that Digital Research (or rather, Graphic Software Systems, for they bear GSS copyright notices!) used MS-DOS services rather than BIOS services for output. The later GSX printer drivers use INT 17H calls, and so too do the unmodified GEM printer drivers. To get GEM running in a hurry, Apricot seem to have released this interception utility.

So if you haven't been able to get hardcopy from GEM, possibly you need to put this little program in your GEM start-up batch file.

Alternatively, it makes the later GSX hardcopy drivers (those you get in the GSX package from Digital Research) work on an Apricot. I had found that they wouldn't work with the IBM emulator — or should I say, with any of the IBM emulators; but they do work with IBMINT17 loaded. For the cognoscenti, the later drivers produce a slightly slicker overall impression — considerably improved for the Epson drivers which I've tried. The C Itoh printed a slightly longer page, and replied more correctly to the enquiry functions. Some IBM GEM printer drivers work on the Apricot, too: that's worth a try if you're stuck.

Maybe there's some future in intercepting other interrupts one at a time in separate programs, rather than using a catch-all like the IBM emulator. One day, perhaps, even Ventura Publisher will run on my Xen ...

Actually, that's not frivolous. A Xen is the perfect Ventura engine, with that extra RAM making large documents possible and a high-resolution paper-white screen perfect for previewing document layout. Ventura loads and runs; but the lack of one or two functions in the GEM screen driver stops it from displaying text properly, and makes it impossible to edit. With a few small alterations to the driver, I have no doubt Ventura will run.

No output from GSX?

I have a program which runs under GSX and/or GEM to draw very complex pictures. On IBM-compatible machines, the program has to run under GEM 2.2 (which it does very well). But on the Apricot I mostly run it under GSX.

A colleague reported some problems on his Tandon, so I moved in to investigate on the Xen. That program wouldn't produce hard copy no matter what I did. It seems that GSX writes a temporary file named ROLLO$$$.TMP, which is a sort of metafile of commands. The plot is generated by rescanning this to see how it interacts with the current printer line. Anyway, when ROLLO$$$.TMP is more than about 120K long, GSX keels over and dies. I think it must exhaust the capacity of certain fixed-length tables in the driver's data space, but that's a guess.

The problem with the Tandon was that whereas GSX handles the clear function by emptying ROLLO$$$.TMP, GEM generates a paper feed instruction. I had put in a number of clear operations during development in order to fix bugs that were actually something else, and hadn't taken them out. Solving his problem wasn't difficult, but I did spend three (long) evenings trying to discover what had gone wrong on the Xen.

By contrast, the screen driver handles things perfectly. Each time GSX operates, pixels are set but the command is forgotten. Plotters are a bit like that too: they are described as having 'inking capability'. Hence with GSX or Apricot GEM hardcopy it is worth being as frugal as possible when programming complex pictures — for instance, do a single polyline rather than many vectors.

V30 chips

What always dissuaded me from fitting a V30 was the thought of the dreadful noise and the Error 31 on power-up! I finally plucked up courage, because I'd heard (from Martin Filmer, who patched up a sick PC for me) about the German Apricot User Group boot ROMs which get over the problem. *(The* APRICOT FILE DEALS FOR READERS *service is*

negotiating to supply a complete V30 *package including the German boot ROMs for around £30; details in our next issue, hopefully — Ed.)*

The V30 came from REDLED in Walton on Thames for £6 (telephone 0932 243600, and don't be discouraged if you get TASKFLOW: they're one and the same); a matching 8087 from the same source was £60 (both plus VAT). To think that the first 8087 I bought cost £295 plus VAT!

Sure 'nough, pre-boot there's a purring buzz from the machine and Error 31 — which means keyboard not connected. Still, Norton's SI reported a performance relative to the IBM PC of 2.1 or 2.2 (depending on which BIOS I used — and BIOS 1.4 gives you 2.9 with a V30!). Compare that with the unenhanced Apricot's 1.1 to 1.2.

But with the new ROMs fitted there was no buzz, no Error 31, no problem. The only difference I could see was that instead of the 'apricot' logo in the top right, you get 'AUG – APRICOT USER GROUP'. The 'AUG' is a sort of logo — and it isn't even in German.

Using this machine with FAST and with WordStar in a RAMdisk, performance is subjectively equivalent to using my Xen HD. Very well worth the price, which totalled £109.65 inc VAT (and including FAST). More than half of this is the 8087, which does no good at all if you're only using the Super packages and/or one of Microsoft's BASICS.

In the past I've found the double-sided Apricot PC sluggish in responding to disk accesses compared to the single-sided equivalent, even though the extra disk space was invaluable. Single-sided floppy machines are ok for word processing, and even for some programming and numerical applications; the disk space is too small for GEM or WINDOWS, though. The V30 provides just that extra bit of oomph in disk operation to make the machine nicer to use — I originally thought it was slow drives, not the limits of processor speed which held me up.

Scrambled disks

I use MSBASIC only as a last resort. But the other day I was demonstrating it to a Basic buff and changed disks before typing SYSTEM. The directory on the new disk was scrambled.

Not wishing to experiment further, I merely offer this as warning to others.

A similar thing happened while running Norton Utilities to recover a deleted file. Having loaded the appropriate program I switched the Norton floppy for my floppy with the deleted file. Sadly, I got the directory of the Norton Disk written over my second floppy. Maybe it's the IBM emulator, or maybe the Norton Utilities don't check for this situation. Beware! ◆

The Workshop

How to do it:

Replace the MicroScreen

By Roger Gann

The MicroScreen keyboard is a self-adhesive membrane strip that is stuck on the outside of the keyboard case. It is attached to the keyboard circuit board by a seven-way ribbon connector; this inevitably means the ribbon has to be folded, and failure of the keys is invariably caused by cracking across the seven metallised tracks. Replacement is not difficult, though.

1. Unplug the keyboard from the system unit. Unscrew the eight screws in the base of the keyboard (one is hidden in the battery compartment); they are of differing lengths, so note where each one came from.
2. Remove the keyboard cover and then the battery. Lift up the lower panel of the keyboard and feed the battery terminal lead through the small hole in the battery compartment. The lower panel is now free.
3. Holding the upper panel to the keyboard chassis, unplug the MicroScreen ribbon connector from the keyboard PCB. Remove the upper panel.
4. Use a knife or some other thin implement to unstick the old MicroScreen keyboard — carefully, now! Pull the now detached ribbon connector through the slot in the upper panel — you should be able to see the cracks in the ribbon.
5. Clean up the gunge in the area beneath the old MicroScreen. Take the opportunity to clear out all the belly-button fluff, fag-ash and breadcrumbs from the rest of the keyboard.
6. Remove the backing paper from the new MicroScreen. Push the ribbon connector through the slot and carefully line up the new MicroScreen with the keyboard; press it in place and smooth it out. The adhesive is fairly strong, so get it right the first time — there are no second chances!
7. Clean the transparent MicroScreen panel and the LCD screen — it's irritating to find fluff under the MicroScreen afterwards!
8. Offer up the keyboard upper panel and plug in the ribbon connector — make sure you catch all the pins.
9. Turn the keyboard upside down, taking care not to stretch the ribbon cable. Offer up the lower panel and feed the battery terminal back into the battery compartment. Replace the battery and the screws.

That's it. Plug the keyboard back into the system unit — and away you go. ◆

Adapted Basic screen dump

Making more of Victor Sirius File

By Eddie Bromhead

In Release 1.11 of the late-lamented VICTOR SIRIUS FILE, Steve Thompson had an article entitled *Make More of Basic*. This involved two parts: a terminate-and-stay-resident handler for interrupt 5AH handler and modifications to MSBASIC.COM; in essence, the mods to MSBASIC made it summon INT 5AH whenever ALT-C is pressed, and the INT 5AH service routine produced a text screen dump on an ACTwriter 12 (*aka* C Itoh 1550).

This was useful: you could obtain a perfect screen dump from a program which displayed text in reverse video, or underlined etc, so that a manual could be prepared showing exactly what the screens looked like.

Steve Thompson's little utility fills a niche when modified for the Apricot. Here's the info to make it work on an Apricot PC/Xi, and with the smaller but otherwise similar C Itoh 8510 (or ACTwriter 10).

MSBASIC.COM

As far as I can see, version 5.28 is functionally identical to MSBASIC under MS-DOS 2.11 on the Sirius. You do not have an ALT key, unless you are running BIOS 3.1, but CTRL-C serves the same purpose on the Apricot.

SDUMP.ASM

To account for the different location of the default character set on the Apricot (line 4 in PSCREEN), the line ...

```
MOV AX,0F000H
```

... must be changed to:

```
MOV AX,00800H
```

If you have a C Itoh 8510 (ACTwriter 10) which has the 10in carriage, modify the E in the string labelled SETUP: to Q. This sets the dot pitch closer, and thus upsets the aspect ratio, and hence the appearance; but it does mean that a printout can be obtained.

SDUMP turns off the cursor before printing and turns it back on completion. This prevents an unpleasant blob appearing on your screen dump. But a program of mine displayed the title as a block of inverse video text followed by a 'press any key to continue' message; the cursor was part of the display, and SDUMP's switching it off caused a 'hole' in my printout. You have to decide whether SDUMP or your main program controls the cursor.

Convert SDUMP.EXE to a COM file with DEBUG rather than EXE2BIN, thus:

```
A>  DEBUG SDUMP.EXE
    - N SDUMP.COM
    - W
      WRITING nnnn BYTES
    - Q
A>
```

SDUMP from compiled languages

Quite simply, write an assembly language subroutine which issues INT 5AH. You don't need to save the registers and restore on completion; SDUMP does that. You have to catch the 'hot key' yourself, though.

Problems

SDUMP doesn't dump graphics. On a PC/Xi you have to look into the CRTC registers to see how many cells across the screen there are: if there are 50, you are in graphics. On the Xen, there is a separate port which you can examine — oh, and the character pointers are a different address in Xen native mode (E0000H not F0000H).

If the screen has scrolled, SDUMP doesn't work properly — its display is staggered on the printout. On the Sirius, it seems things are fine if you reset the screen with Escape E and only use direct cursor addressing thereafter. On the Apricot, use ESC z to reset the screen defaults first.

It is possible to interrogate the CRTC chip to sort out how the screen has been scrolled, or even whether or not we are in text or graphics mode etc. This is even easier on the Xen. The font pointers for lines 1 to 24 are independent of those for the status line (25), however, so the job gets very difficult. Using ESC E only unscrolls lines 1-24! So use something else to dump graphics screens, and reset the screen attributes with Escape z before displaying the screen which you want to dump.

SDUMP with another printer

This means revising the printer Escape code strings at the end of SDUMP. Epsons display their bit image data with the least significant bits at the top: this is inverted with respect to the C Itoh printers. This means rotating some of the SHIFT and ROTATE commands from left to right and *vice versa*. ◆

Eddie's mildly amended reworking of the original VICTOR SIRIUS FILE listing of SDUMP.ASM follows. The assembler code and the .COM file will appear on a forthcoming DEALS FOR READERS disk.

SDUMP.COM

```
; Written by Steve Thompson
; Copied from VICTOR SIRIUS FILE Release 1.11
; Trivial amendments by Eddie Bromhead
; (mainly in the comments!)
;
; Sets up interrupt 5AH service routine to dump text
; screen to ACTwriter 12, modified to suit ACTwriter 10
;
;
CODE    SEGMENT
        ASSUME CS:CODE,DS:CODE
        ORG Ø1ØØH
;
START:  MOV AX, 255AH           ;Load routine
        MOV DX, OFFSET SDUMP
        PUSH CS
        POP DS
        INT 21H                 ;Set INT 5AH vector
        MOV DX, OFFSET ENDMARK
        INT 27H                 ;Terminate and
;                                stay resident
;
SDUMP:  PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        PUSH BP
        PUSH SI
        PUSH DI
        PUSH DS
        PUSH ES
        PUSH CS
        POP DS
        MOV DI, OFFSET REPORT   ;Turn off screen cursor
        CALL PSTR               ;Save cursor position,
        MOV AH, Ø8H             ;report character under
                                ;cursor
        INT 21H                 ;Print character under
        MOV DI, OFFSET REVERSE  ;cursor in
                                ;reverse field
        MOV [DI+2],AL
        CALL PSTR               ;Send setup string
        MOV DI, OFFSET SETUP    ;to printer
        CALL LPSTR1
        MOV DH, 8ØH
        CALL BORDER             ;Print border top line
        CALL PSCREEN            ;Dump screen to printer
        MOV DH, Ø1H
        CALL BORDER             ;Print bottom line of
                                ;border
        MOV DI, OFFSET REVERSE  ;Print char under
        ADD DI, Ø2H             ;cursor in normal
                                ;video field
        CALL PSTR
        MOV DI, OFFSET RESTORE  ;Turn on cursor and
        CALL PSTR               ;restore position
        MOV DI, OFFSET RESET    ;Reset printer
        CALL LPSTR1             ;to normal mode
        POP ES                  ;Restore registers
        POP DS
        POP DI
        POP SI
        POP BP
        POP DX
        POP CX
        POP BX
        POP AX
        IRET                    ;End of main routine
;
PSTR    PROC NEAR               ;Print string to screen
        MOV AH, Ø2H
NEXT:   MOV DL,CS:[DI]
        OR DL,DL                ;Test for end marker
        JZ ENDSTR
        INC DI
        INT 21H
        JMP NEXT
ENDSTR: RET
PSTR    ENDP
;
LPSTR1  PROC NEAR               ;Print string at [DI]
        PUSH CX
        MOV CX,CS:[DI]          ;Set CX to length
        CALL LPSTR2             ;of string
        POP CX
        RET
LPSTR1  ENDP
;
LPSTR2  PROC NEAR               ;Print CX characters
        INC DI
        INC DI                  ;DI points to
        MOV AH,Ø5H              ;first character
NEXTC:  MOV DL,CS:[DI]
        INT 21H                 ;Print character
        INC DI
        LOOP NEXTC              ;Goto next character
        RET
LPSTR2  ENDP
;
BORDER  PROC NEAR               ;Print top or bottom
        MOV DI,OFFSET LINE      ;border
        MOV CX,Ø6H
        CALL LPSTR2             ;Print prefix string
        MOV CX, Ø322H
        MOV DL, DH
        MOV AH, Ø5H
REPEAT: INT 21H                 ;Print 808 dots
        LOOP REPEAT
        MOV DI, OFFSET LINE
        ADD DI, 8Ø8
        MOV CX, Ø2H
        CALL LPSTR2             ;Print suffix string
        RET
BORDER  ENDP
;
PSCREEN PROC NEAR               ;Screendump routine
        PUSH DS
        PUSH ES
        MOV AX, ØFØØØH          ;ØEØØØH for Xen
        MOV DS, AX              ;DS points to
        XOR AX, AX              ;screen RAM segment
        MOV ES, AX              ;ES points to character
                                ;table segment
        MOV SI, ØØ8ØØH          ;ØFF6ØH for Sirius
        MOV CX, 32H             ;Set CX for 50
                                ;half-lines
NEXTL:  PUSH CX
        MOV DI, OFFSET LINE
        ADD DI, Ø9H             ;DI points to bit image
                                ;print string
        MOV BP, ØØ1ØH
        ROR CX, 1
        JC ODDLINE
        ADD SI, ØØAØH           ;Increment screen RAM
        XOR BP, BP              ;address on even
                                ;half-lines
ODDLINE: XOR BX, BX
        MOV CX, ØØ5ØH           ;Set CX to number of
                                ;characters per line
NEXTA1: PUSH CX
        PUSH BX
        PUSH SI
        LEA SI, [BX+SI]         ;Set next address in
        MOV AX, [SI]            ;screen RAM
        MOV DH, AH
        AND AX, Ø1FFH
        MOV CL, Ø5H
        SHL AX, CL
        MOV SI, AX      ;SI contains char table offset
        ADD SI, BP      ;plus offset for top/bottom
        MOV BX, Ø1H     ;half-line
        MOV CX, ØAH     ;Set CX to character width
NEXTA2: PUSH CX
```

```
        PUSH SI
        MOV CX, Ø8H
NEXTA3: MOV AX,ES:[SI]  ;Load character table word
        INC SI
        INC SI
        TEST DH, 2ØH    ;Check for underline
        JZ WMASK
        TEST AX, 8ØØØH  ;Underline position?
        JNZ ON
WMASK:  AND AX, BX      ;Mask character table word
        JZ OFF
ON:     STC             ;Set dot on
OFF:    RCR DL, 1       ;Assemble bit image in DL
        LOOP NEXTA3     ;Next word in character table
        TEST DH, 8ØH    ;Check for reverse field
        JZ NORMAL
        XOR DL, ØFFH    ;Invert bit image
                        ;if reverse video
NORMAL: MOV CS:[DI],DL  ;Move bit image to print string
        INC DI
        SHL BX, 1       ;Shift mask
        POP SI
        POP CX
        LOOP NEXTA2     ;Next in character width
        POP SI
        POP BX
        POP CX
        INC BX
        INC BX
        LOOP NEXTA1     ;Move to next char along line
        MOV DI, OFFSET LINE
        CALL LPSTR1     ;Print bit image of half-line
        POP CX
        DEC CX
        JZ FINISH
        JMP NEXTL       ;Next half-line
FINISH: POP ES          ;Screen dump complete
        POP DS
        RET
PSCREEN ENDP
;
REPORT:   DB 1BH,'X5',1BH,'J',1BH,'$',ØH
;
REVERSE:  DB 1BH,'P ',1BH,'Q',ØDH,ØAH,1BH,'H',ØH
;
RESTORE:  DB 1BH,'Y5',1BH,'K',ØH
;
SETUP:    DB 25H,Ø,1BH,'>',1BH,'T16',1BH,'Q',1BH,
            'LØØØ',ØDH
          DB 23 DUP(ØAH)
;
;         (this 'Q' is 'E' for a wide-carriage
;         ACTwriter 12)
;
RESET:    DB 26H,ØH,24 DUP(ØAH)
          DB 1BH,'<',1BH,'T24',1BH,'N',1BH,'LØØØ',ØDH
;
LINE:     DB 2AH,Ø3H,1BH,'SØ8Ø2',ØFFH
          DB 8ØØ DUP(Ø)
          DB ØFFH,ØDH,ØAH
;
ENDMARK:  DB Ø
;
CODE    ENDS
        END START
```

About the newsletter

Our aim is to provide a complete editorial and information service to users of Apricot computers.

Except in the normal course of day-to-day contact, sub sales and the like, we are not affiliated with or supported by Apricot plc or any of its dealers. Nor for that matter are we sponsored by or directly associated with any user group or any other such organisation, though our goals are similar to the declared aims of many such bodies.

The Publisher does not necessarily agree with the views expressed by contributors, and assumes no responsibility for errors of reproduction or interpretation in the subject matter of this newsletter or from any results arising therefrom.

APRICOT FILE is a publication of TP Group Limited, registered in England no 1929195.

Postal address

TP GROUP
PO Box 509
London N1 1YL, England

Personal calls and deliveries

Unit 4, 47-51 Wharfdale Road
London N1 9SE, England
(please phone first!)

Phone, E-mail, Telex

Telephone	**01-833 3501**
MicroLink/Telecom Gold	**72:MAG068**
	[from overseas 10072:MAG068]
The Source	**BDE566**
Telex	**265871 MONREF G**
	quote reference 72:MAG068

The People

Editor	**Dennis Jarrett**
Subscriptions	**Terrance Duggan**
DFR/Technical	**Roger Gann**

The Guarantee

The annual subscription to APRICOT FILE buys 12 monthly issues as follows:

Britain	£50.00
Rest of Europe	£60.00
Elsewhere (air mail)	£65.00

If having subscribed you come to feel that APRICOT FILE does not meet your requirements or that it does not match our claims, we will gladly cancel your commitment to the magazine and refund the unused portion of your subscription.

■■■ ISSN 0268-6066 ■■■

Free Small Ads

Apricots for Sale

■■■ APRICOT XI 10MB, 896K, 12in monitor. Owner reluctantly forced into the Big Blue world, so good home needed for faithful computer. Lightly used; original manuals, original boxes. Original SuperCalc, SuperWriter, SuperPlanner; plus Open Access, WorkMate, QuickCode, PASD, UserPrompt Guides, many other utilities and public-domain software — over 5MB in all. Offers closest to £800. Contact Colin on 021-308 2521.

Portables for Sale

■■■ APRICOT PORTABLE — 512K, MS-DOS 3.2, BIOS 3.1, NEC V30 chip, Second-issue GEM (debugged 'B' version), mouse, MiniPrinter, Dragon speech, Turbo Pascal 3 with Tutor, Transend comms plus serial cable, carrycase, 30+ public-domain and shareware disks: £425. Ring Mark Wren, 06273 2520 or e-mail to Gold 72:MAG20590.

Xens for Sale

■■■ XEN HD20 with 14in paper-white monitor, mouse and software, on-board Apricot Modem and Communiqué. Substantial software (with documentation) also available by negotiation. Three months left on transferable maintenance contract. Best offer over £1,750 for quick sale. Call 0222 759562.

■■■ THREE XEN HD20s: installing VX network late summer — will have three HD20s with paper-white monitors for sale Sept-Oct. Light office use only. Singly at £1,699 each; or two for £1,649 each, three for £1,599 each. John Andrews, Geevax Ltd, 0684 850850.

Extras for Sale

■■■ F SERIES DISK DRIVES (two) and MICE (two) for sale. Offers to Colin on 021-308 2521.

■■■ CLEAROUT — Multicard Xi EXPANSION BOX (takes one Xi slot, room for five cards); Apricot internal MODEM; external printer BUFFER; Apricot Xi COLOUR CARD and driver, as new. Also BACK ISSUES of Apricot File (back to 1.1), Comms File (first year), 16 Bit Computing (incomplete). Offers? Ring Mark Wren, 06273 2520 or e-mail to Gold 72:MAG20590.

TP Group Services

■■■ DISK RECOVERY: problems with deleted files, scrambled hard disks, and the like? We can (probably) help — no fix, no fee! Call Roger Gann at TP Group on 01-833 3501.

■■■ FILE TRANSFER: we can convert files between Apricot single- and double-sided, IBM PC format, PC/AT format, and Sirius — cheap! Call Roger Gann at TP Group on 01-833 3501.

■■■ DESKTOP PUBLISHING: we can produce your newsletters, price lists, press releases etc in near-typeset-quality on our laser printer using our DTP software NEWSWRITER from your disks in virtually any format produced by word processor and/or computer. Fast turnaround, guaranteed quality, low prices. You're reading an example of our work right now ... Call Roger Gann at TP Group on 01-833 3501.

Other services offered

■■■ HELP for PC/Xi/Xen users — setting up systems, MS-DOS, directory organisation, comms, modems. ALSO utility and business software written (in C, assembler or Basic). Call Tony Bridges, 01-948 1603 any time.

■■■ REPAIRS guaranteed. Martin Filmer Computing Services. 01-761 0425.

■■■ A0 PLOTTING — time on a Calcomp A0 plotter available for rent: Sirius or Apricot disks plotted on your paper or ours. Terms negotiable, but cheap. Phone James McEvoy on 01-859 4014 or 01-852 7981; or write to 16 Dunvegan Road, London SE19 1SA.

■■■ SUPERWRITER BUGS CURED — twelve files identifying and curing errors in SuperWriter version 1.03 program and manual, all for a copying fee of only £5.75 (includes VAT). Call Tudor Wynn-Williams on 097 081 375.

■■■ HARDWARE AND SOFTWARE at bargain prices. Will beat any written quotation. Try us before you buy. 0382 66312 (day), 021-454 0898 (evening).

■■■ APRICOT F1 MOD — end keying uncertainty. If you can solder two wires, you can apply this simple mod. £5 for full details: W Houston, 11 Brian Close, Chelmsford, Essex CM2 9DZ.

Wants

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

Contacts

■■■ F SERIES user wants to get in touch with others in the States. Contact Bob Mandel, Lewis & Clark College, Box 96, Portland, OR 97219, USA.

■■■ GOSPORT BULLETIN BOARD has an active and knowledgable Apricot section (used to run on an Xi!) with mail, PD software, lots of tips, etc. Set up for N-8-1, call 0705 552 4805 — speeds between 300 and 2400 bps.

■■■ SOUTHERN APRICOT GROUP meets near Salisbury six times a year. Information: Mike Dirom, Highover, Glasses Lane, Sutton Mandeville, Salisbury, Wilts SP3 5NB (SAE please) or call 0722 70219.

■■■ SWITZERLAND: Apricot user would like to contact others, preferably in the Zurich area, to exchange experiences and help each other with problems. Contact John Solomir, Stockenstrasse 67, 8802 Kilchberg.

■■■ PORTABLE user would like to contact another in South London to exchange information. Contact Geoff Reade, 9 Taymount Grange, London SE23 3UH.

Use the service!

The Classified Ads service is free to all readers of APRICOT FILE. Just let us have a sheet of paper with the words (not too many); your ad will appear at least once in the newsletter and for around three months in the quarterly DEALS FOR READERS broadsheet.

More maintenance

Mark Wren writes from the edge of Dartmoor to recommend **Micro Systems Maintenance** on **0793 616888**; MSM provides nationwide next-day cover but Mark says the engineer has often arrived the same day, always supplies a temporary replacement for equipment that can't be fixed on the spot, and can justify the claim to look after all Apricots with a whole series of fixes to an F1, a Portable, and an Xi20 (not to mention three printers).

Epson increases prices

Following the EEC's decision to impose a **33.4 per cent levy** on dot matrix printers imported from Japan, **Epson** has announced price increases across the board. The average increase is 10 per cent, though the actual amounts include just £4 on the RRPs for several models — from the low-end LX800 nine-pin printer (now £279) to the GQ3500 laser (now £1,799). Epson says these modest rises are directly attributable to its policy of manufacturing in Europe (some printers are made entirely in Telford and Paris, all have parts manufactured there) — but the company still declares that "this levy will hurt Epson. It will hurt our profitability. It will hurt our investment in the UK".

Epson issued a detailed statement attacking the levy. This was echoed by a press release from **Fujitsu**, also subject to the surcharge — "Fujitsu believes that this duty does not accurately reflect the difference between prices in the EEC and Japan ... The method used to calculate the dumping margin was unfair and inaccurate ... Far from causing injury to European industry, Fujitsu has improved the state of the market in Europe ... which has been of positive benefit to European industry and employment". Fujitsu says its Malaga plant will now be meeting all of its European demand, with the European content of its printers due to rise from about 40 to over 60 per cent.

Software sales

The slight mid-1980s hiccup in the growth of the US computer market appears to be over. First-quarter retail sales of **microcomputer software** were up 50 per cent to **$938 million** over the year-ago figures; the US **Software Publishers Association** comments that 1988 is likely to be "spectacular" for its members.

Fax keeps booming

The facsimile business's trade body, the BFICC (British Facsimile Industry Consultative Committee), claims that **UK fax installations** have now topped the **200,000** mark — which means there are now twice as many fax terminals in Britain as Telex terminals. BFICC is forecasting a total of 150,000 new installations for 1988; British Telecom has a more bullish estimate of 180,000 sales for the year.

Apricot update

Latest moves from **Apricot** include a somewhat tardy promotion for the PC-286 upgrade (via selected dealers only, with the from-£999 pricelist we gave at the start of the year) and an OEM agreement for SCO's XENIX System V for the Xen-i 286 and 386 — SCO is the world leader in XENIX implementations and probably represents a good bed for Apricot to get into.

Laser alternative

Prices for laser printers are slowly drifting downwards, but a realistic alternative with better price-performance seems to be in the offing now that **Casio** has started to find serious OEM customers for its **liquid-crystal shutter** engine. LCS works very like laser printing, the only difference being that a row of liquid-crystal light sources is used to write the image instead of a moving beam of laser light. In theory this should mean better reliability, (fewer moving parts), smaller printers (a more compact mechanism), and easier (hence cheaper) manufacturing because LCS utilises much simpler opto-electronics.

The first LCS printers are starting to appear, with the **Data Technology Crystalprint** in the van. **Micro Peripherals** has just signed up to sell the two Crystalprint models — a simple Diablo-compatible daisywheel replacement at **£1,149** and a very attractive LaserJet emulator at **£1,499**. The latter should be the stronger seller, of course, though the real price benefits won't appear until production volumes rise; still, compatibility with the HP LaserJet Series II usually costs over £2,000. Plug-in fonts and toner cartridges appear to be priced at the HP level, though, which is slightly disappointing.

It will certainly be interesting to see how LCS technology shapes up in the price-performance stakes for DTP-quality applications. A recent review in *Byte* magazine suggests that print quality from another Crytalprint model was slightly inferior to the LaserJet, though the untrained observer might notice that only on grey-scale printing. In the meantime, **Micro** Peripherals is on **0706 217799**.

Yet more MicroLink

Back at Telecom Gold, the latest from the MICROLINK electronic mail service — which runs as a closed user group on Gold — is **Mortgage Desk**. This is a mortgage comparison facility which offers personal advice on mortgages and house conveyancing.

We think MICROLINK is getting better and better; if you'd like to check it out for yourself, try the free interactive Dial-a-Demo — call **01-583 1275** if you have a 1200/75 bps modem (or **01-583 3000** if you have 300 bps), at the PAD> prompt enter 'CALL 72' and Return, sign on with 'ID MAG111 DATABASE'. That will bring up the MICROLINK demo menu, from which you can browse through descriptions of different aspects of the system.

As a service to readers, we have fixed up a free-subscription deal for APRICOT FILE subscribers interested in joining MICROLINK — it saves you the £10 registration fee. Drop us a note if you'd like details and the sign-up form.

Where are they now?

Chris Buckham was one of Apricot's more visible marketing people around the time of the Apricot's launch and the abortive US venture. He's now resurfaced with **Graphic Point**, a drop-in DTP bureau in Birmingham that features some £100,000 worth of equipment. "Liaisons" with dealers are on the cards. ◆