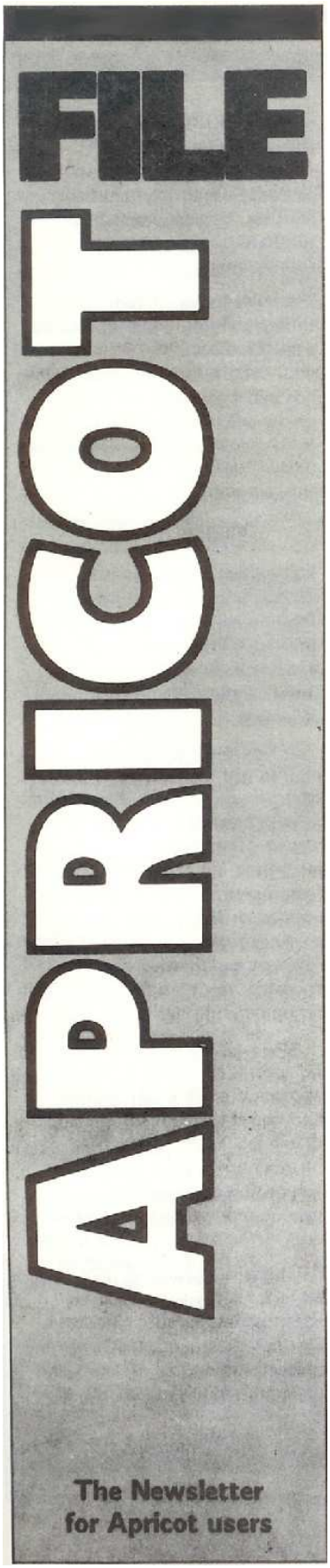
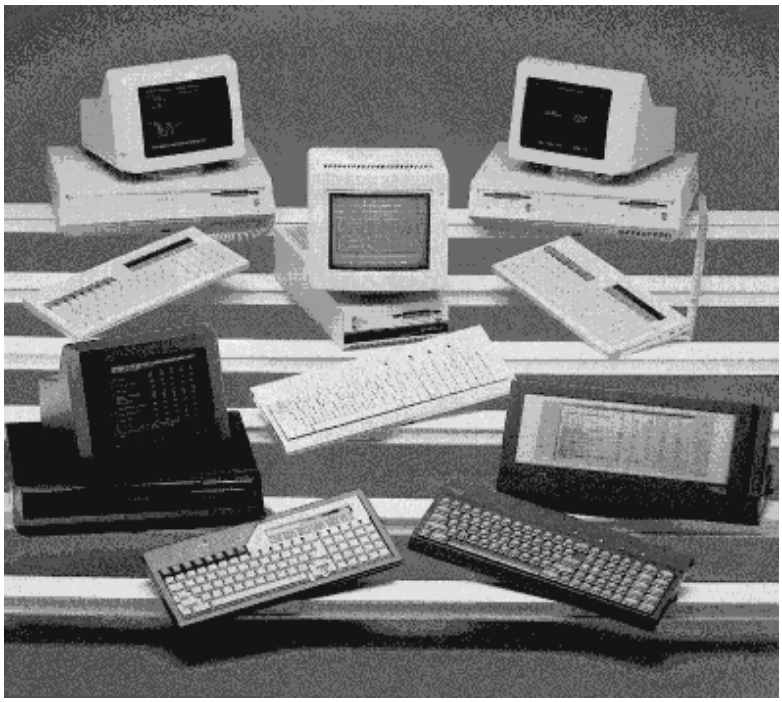


Apricot File 2_9



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 2.9

FILE APRICOT

The Newsletter for Apricot users

June 1987

Pulsar and Apricot Accountant

A new lease of life? PACT acquires the accounts packages, guarantees support

In the days when ACT and subsequently Apricot was keen to promote itself as a one-stop-shopping company that sold *everything* you might need, the Pulsar range of accounts packages was a keynote product. Originally launched in 1982 on the Sirius, Pulsar was implemented on the Apricot (with rather fewer bugs) and subsequently changed its name to Apricot Accountant: there are reputed to be "in excess of 10,000 active users", most of whom have the non-Xen 8086-based Apricots.

On the other hand, Apricot Accountant clearly doesn't fit the current image of Apricot Computers plc — it would rather the VARs and systems houses actually serviced the needs of end-users, which is why its current software offerings are limited to reselling other people's stuff. Running a software development and support operation is not the way Apricot plc will make its money in the future.

Clearly, though, there's a market in support and upgrade for existing Apricot Accountant users. And Apricot Accountant is reportedly a pretty good suite of accounts packages, particularly in its most recent multi-user forms: so there should also be a market in new sales — especially for networks (Apricot Accountant will run on the IBM Token Ring and Novell as well as Apricot's own networks) and especially in non-Apricot formats. That means IBM-compatible, and indeed Apricot did develop and halfheartedly attempt to sell such an implementation.

Now all rights to Apricot Accountant have been acquired by a new company called **PACT Associates**. This is effectively a management buy-out; PACT comprises a collection of ex-Apricot research and support people under **Paul Wintermantle**, a former Apricot Technical Director (and 'PACT' apparently comes from 'Past ACT'. He reckons "Apricot Accountant is still one of the most versatile and effective accounting systems available" (but then he does have a vested interest).

The Good News is that PACT will cheerfully cater for existing customers — many of whom have been getting a bit twitchy of late — with "a full upgrade path for existing Pulsar and Apricot Accountant users" and "a firm commitment to provide continued support". Wintermantle said this "most definitely" means support for packages on the orphaned 8086-based machines, though there won't be any further development.

Where's the border between 'development' and 'upgrade'? We sprang a couple of trick questions, like will existing users be upgraded to take account of the revised VAT arrangements on offer from October to smaller businesses, and he manfully said "we'll have to look into that one" (we heard him scribbling a note, so he might).

▶▶▶ CONTINUED ON PAGE 2

In this issue

News of Apricot 2
The Complete Apricot Escape Codes — part one 3
Books to buy (or not) 8
Doodles by David: Tips and Tricks 11
The Workshop: Splitting up big ASCII files 12
The Workshop: Inside MS-DOS — the 'environment' 14
The Workshop: Apricot's IBM emulators 18
Contributors in this issue 21
Diary Dates 21
Bandage and Plaster: Boilerplating with SuperWriter 22
Review: On-line attendant — Ansible's ACOLYTE 28
Free Small Ads 31

Apricot Accountant

And although it's obviously eager to sign up dealers — PACT believes it can give them "higher levels of sales support" — Wintermantle said "we'll talk to anybody ... We much prefer people to use their own dealers" (because that keeps the dealers sweet, and no decent-sized software company can survive without keen dealers) "but we recognise that many of the dealers who supplied Accountant and Pulsar are no longer around".

Initially that means PACT welcomes one-off queries; but the new company also intends to introduce a direct end-user hotline service. This comes as part of a support package which will cost you **£200** a year for the first three modules, plus **£50** pa per module thereafter. (Those are prices for single-user installations; for multi-user packages the rates are £300 and £75 respectively.)

As well as unlimited telephone help — it's quite rare among software firms that there's no restriction on calls for hotline service — the subscribers will get a half-yearly bulletin and free upgrades. To qualify you'll need to quote your Pulsar/Apricot Accountant registration number.

PACT Associates is at **Lygon Court, Hereward Rise, Dudley Road, Halesowen, West Midlands B62 8AN** — phone **021-503 0022.**

Back at the ranch ...

According to market researchers **IDC**, **Apricot** and **IBM** were the main losers of **market share in Western Europe** during 1986 when **Amstrad** became the largest single supplier with **1.75 million units shipped.** IBM's shipment volume was down 11 per cent, Apricot's by **55** per cent.

That's obviously a reflection of Apricot's departure from the low-end mass-shipment market. And in the first quarter of this year, **Romtec** reckon Apricot was responsible for **5.7 per cent** of the micros sold in Britain and **8.8 per cent** of the market's value. In terms of units sold, that gives it third place (a long way behind **Amstrad** with 39 per cent and **IBM** at 20.4 per cent); for value, Apricot's shipments place it behind IBM (33.6 per cent), Amstrad (13.4 per cent) and Compaq (8.9 per cent).

The strong performance of Apricot and Compaq suggest that the market as a whole is moving away from straight PC-compatibles towards 286- and 386-based machines. If you bought Apricot shares on our recommendation six to eight months ago, when they were below 60p, we feel you shouldn't sell yet — in our view they'll dip during the summer from the current price (around 115p as we go to press) but should reach 150p by the end of the year.

That's largely because we think the the Xen-386 family should do quite well this year. Along which lines, we hear that it's getting some decent-looking software to improve its market prospects. Following the February 1987 deal with **Root Computers** for an Apricot-brand version of that company's Unix/Xenix front-end utilities, Apricot has just signed up with another of the early majors in the **Xenix software** market: **Sphinx** reckons the contract for a range of Xenix packages could be worth £200,000 a year.

Products included are those for which Sphinx is acting as a distributor — the **Informix** 4GL applications developer, the **FoxBASE** database system, **SCO**'s spreadsheet, the **Lyrix** word processor, and Redwood's best-selling **Uniplex** integrated 'office automation' system.

That certainly fleshes out Apricot's 386-based machines with some real end-user products. On the other hand, supplies of Xenix itself are hard to come buy; Apricot was originally quoted April as a shipment date, but **Microsoft** apparently decided to amend chunks of the code as a result of discussions with AT&T that pointed up further opportunities for merging the specs of Xenix and Unix.

And Apricot has moved pretty quickly to graft its own kit on to the **HiNet local networking system** it acquired at the end of March as part of the **DMS** operation it bought from Extel. Hitherto DMS has based its hardware offerings on rebadged Zenith machines; now a raft of hardware and software announcements from Apricot include program upgrades and **model 4000** file servers based on the 80386 Xen engines to support HiNet concurrently with MS-NET, while at the same time supporting gateways into public comms services like PSS, Telex and Telecom Gold. There are also new workstations based on the 286 Xens.

That should give existing HiNet users some long-awaited upgrade paths; the sales line is the 4000 series "combines the power of a new 386-based machine with our proven communications software", and that sounds convincing enough.

DMS has now been subsumed into Apricot's Networking and Communications Division, based at the ex-DMS offices in Wokingham. The group claims a combined total of 8,000 installed networks, which must make it one of the serious players in that particular game. ◆

APRICOT FILE contact points

POSTAL ADDRESS

PO Box 509
London N1 1YL
England

PHONE, E-MAIL, TELEX

Telephone	**01-833 3501**
Telecom Gold	**81:JET767**
Mercury Link	**19011170**
The Source	**BDE566**
Telex	**265871 MONREF G** quote reference 81:JET767

THE PEOPLE

Editor	**Dennis Jarrett**
Production	**Fiona Carey**
Writer	**Garret Keogh**
Subscriptions	**Terrance Duggan**
Technical	**Roger Gann**

ISSN 0268-6066

Escape plans

All about ESCape codes: Part One

By David Langford

'Escape sequence' is another of those phrases which cause a nervous drying-up of computer users' comprehension ... which is a pity, since this is the key to all Apricot screen control beyond the simplest line-by-line text display.

When you send text characters to the screen, they're displayed and that's that. For suave effects you must send special control codes, and the most powerful of these is 'Escape'.

Once you've grasped its rules, you can make the Apricot display sing, dance and cook Brown Windsor Soup. And for that, you'll need this three-part series providing as complete an overview as you'll find.

Preliminaries

The boring historical background is that the Apricot screen, like that of its ancestor the Sirius, has been set up to resemble something called a VT52 terminal ... via software control. As with the famous editable keyboards and fonts, Apricot display handling isn't 'hard-wired'. Programmers don't need to know anything about how the video memory is arranged: the BIOS screen driver will handle all the fiddly details if sent the correct command via an escape sequence. (Incidentally, there are many more commands available to Apricot users than were defined in the old VT52 standard.)

'Sending' something to the screen simply means displaying it, as with Basic's PRINT command or Pascal's Write, or such MS-DOS commands as ECHO and TYPE. Sending special control characters is a legal fiction: you pretend you're displaying them, but in fact the BIOS nobbles them *en route* to the screen, and take appropriate action rather than simply showing the relevant font character.

This is why the font characters with ASCII codes from 0 to 31 [hex codes 0 to 1F] — the funny faces, playing-card symbols, musical notes, arrows, and the like — are something of a snare and delusion. Normally, all are intercepted and treated as control characters, though only a few actually 'do things'; see the panel.

(Note the treatment of ASCII character code numbers: throughout this series I'm giving the decimal value followed by the hexadecimal value in square brackets.)

Selected control characters

Name	ASCII	Function
beep or BEL	7	sound internal beeper
backspace or BS	8	move cursor one space left
tab	9	move cursor to next tab stop
linefeed or LF	10 [0A]	move cursor one line down
return or CR	13 [0D]	move cursor to lefthand column
escape or ESC	27 [1B]	signal escape sequence

Notes:

- A backspace at the beginning of a screen line will 'reverse wrap' to the end of the previous line, except at the beginning of the screen.
- Tab stops fall at eight-space intervals, in columns 9, 17 , 25, etc; the tab code itself will not wrap to the next line at a line-ending, despite MS-DOS's misleading response to it.
- A linefeed on the last line of the screen will cause the whole display to scroll up. Remember that a Return alone doesn't move the cursor to a new line ... but see the ESC x8 and ESC x9 controls under 'Environment' in Part Two of this series.
- More esoterically, I find that character codes 11 [hex 0B] and 12 [hex 0C] — vertical tab and formfeed — behave just like linefeeds on the Apricot.

One possible relic of the Sirius is to be found on the Xi before the coming of BIOS 3.1. Characters 14 [hex 0E] and 15 [0F], otherwise known as so and SI, feebly imitated their Sirius function of switching in and out a second character set; so caused subsequent text output to appear as blanks until SI restored sanity.

Let's clarify the phrase 'control character'. Character 7, BEL, is also known as Control-G or ^G; BS may be referred to as Control-H or ^H; and so on. The ASCII codes correspond to letters of the alphabet (and even beyond: the ASCII character after 'Z' is an opening square bracket, so we get ^[for Escape); and special characters are traditionally generated by holding down the CONTROL key plus a letter key, so it makes a sort of sense.

But holding down CONTROL and pressing 'G' in MS-DOS or a program does not send ^G to the screen and cause a beep. It merely sends ^G to the MS-DOS command interpreter or your program, which will have its own interpretation — many word processors including SuperWriter and WordStar take ^G to mean 'delete current character', for example. Only when a program sends ^G to the *screen* does the beep occur. In MS-DOS you can achieve this with ECHO ^G, where the ^G is entered with CONTROL plus 'G' as above.

Getting ESC sequences to the screen

To invoke the special screen driver functions, we send Escape to the screen. This is straightforward except at the MS-DOS command level, where the ESC key itself introduces some special editing functions which can be quite handy. They're buried deep in the MS-DOS manual: the most useful is ESC U (it must be a capital U), which repeats the previous command line.

Actually to produce an ESC character, you have to key ESC R. Type ECHO followed by a space, then hit ESC and 'R' to put in the ESC character (it appears as ^[), and finally add the rest of your escape sequence.

As an example, let's take the simple escape sequence which when 'displayed' clears the screen — ESC E. The ESC is received by the Apricot screen driver, and interpreted as meaning 'Don't print the next character but execute the screen driver routine corresponding to its ASCII code'. The E is the code for 'clear screen'.

From MS-DOS you could do it with the CLS command, but ESC E has an extra advantage over MS-DOS 2 — it 'homes' the cursor to the top left position on the screen. (This is done by some versions of COMMAND.COM encountered with Apricot's MS-DOS 3.) In a batch file you could include the line ECHO ^[E to do this (where the ^[can be inserted by ESC R as above, or by your word processor's 'insert control character' function).

(Note that ESC sequences involving the characters which are the MS-DOS redirection operators — the less-than arrow <, the greater-than arrow >, or the upright bar — cannot be sent to the screen using ECHO. But they can be put into a small text file and the file sent with TYPE.)

In MSBASIC it's worth defining a string to provide the ESC character — ESC$=CHR$(27), for example. Then PRINT ESC$+"E" will clear the screen. Since this is such a common function, you might as well simplify it further by defining a special string, something like CLS$=ESC$+"E". That would allow you to PRINT CLS$ whenever you like.

Alas, GWBASIC doesn't permit you to use ESC sequences: its graphics screen merely displays the font equivalents of control characters, such as a little left-pointing arrow for ESC.

Turbo Pascal allows much the same approach as MSBASIC. Put 'Escape=#27;' or 'Escape=^[;' among the Constants, and in the main program have 'Write(Escape,'E');'. Or define 'ClearScreen=^['E';'. Then you can Write that as required. This is a rotten example since Turbo has its own 'ClrScr' function, but never mind.

In Assembler the temptation is to define sequences with such lines as

```
CLRSCR DB 27,'E$'
```

and use the MS-DOS 'print string' function call. Beware, though, the fact that the dollar sign marks the end of a string! Not only is ESC $ an escape sequence in its own right, but we'll be meeting sequences which can include a dollar sign in their data parameters. It could be safer to send them character by character or to write your own simple string display routine. Here's a tiny Assembler screen-clear procedure:

```
CLR_SCREEN PROC NEAR
ESC EQU 1BH        ; define ESC
CHAR_OUT EQU 29H   ; MS-DOS `display character'
                     interrupt
                   ; (use Apricot's F1H if this
                     needn't
                   ; run on Xi BIOS 2.4 or
                     earlier)
MOV AL,ESC
INT CHAR_OUT       ; send ESC from AL
MOV AL,'E'
INT CHAR_OUT       ; send E from AL
RET                ; return from procedure
CLR_SCREEN ENDP
```

Under way

After all those preliminaries, it's time to tackle the actual vast table of ESC sequences and their functions. The best approach seems to be to group them under logical headings rather than leave you to ferret through a listing in arbitrary order of ASCII number *(in the final part of this series we'll provide one as an end-of-term summary and* aide memoire *in any case — Ed)*. I'll put in cross-references where they seem needed. Always consult the 'Notes' before rashly inserting anything into a program ... there are many variations and traps to watch for!

Except where specifically mentioned, references to the Xi include the PC and Xen, and references to the F series include the Portable.

Several ESC sequences expect additional characters of data after the initial ESC and its single-character function code. I've represented data parameters by appropriate letters where possible: n or m for plain numbers, x and y for row and column co-ordinates, c for colour settings.

There are opportunities for confusion here. Normally any following characters are interpreted as numbers, but must be sent as ASCII character codes. For example, ESC Qn scrolls the screen left by (n-32) columns. To scroll the screen left by one column, we send ESC Q followed by 33 ... not thirtythree but the ASCII character whose code is 33 [21], being the exclamation mark. ESC Q! is the needed sequence.

But there are a few cases where a code needs to be sent in ASCII form, whereby '1' refers not to ASCII character number 1 but to character 49 [31], being the ASCII code for the digit 1. See for example the 'program key' and 'set screen mode' sequences. In these cases I've distinguished 'ASCII-form' parameters with the letter a.

Remember that the first character of an ESC sequence (the second if you count ESC itself) will always be shown exactly as it should be sent. ESC n means ESC followed by 'n' (ASCII character 110 [6E]), but ESC /n means ESC followed by a slash (47 [2F]) followed by any number n represented by the appropriate ASCII code.

Controlling the cursor position

See also the table of Control Characters on the previous page; and note that there are no ESC sequences which exactly duplicate Backspace, Tab, Linefeed or Return.

ESCape	Dec	Hex	Function
ESC ;	59	[3B]	move cursor to status line
ESC A	65	[41]	cursor up one line
ESC B	66	[42]	cursor down one line
ESC C	67	[43]	cursor right one space
ESC D	68	[44]	cursor left one space
ESC H	72	[48]	home cursor (to top left of screen)
ESC I	73	[49]	cursor up one line; scroll at top line (reverse line feed)
ESC Yyx	89	[59]	a three-character sequence — move cursor to position designated by y (row+32) and x (column+32)
ESC h	104	[68]	reverse tab, to previous tab stop
ESC j	106	[6A]	save current cursor position
ESC k	107	[6B]	restore previously saved position

Notes:

- The point of the status line is that it isn't affected by other screen operations: it doesn't scroll out of sight, ordinary screen-clear commands don't affect it unless the cursor has previously been moved to the status line, and so on. It's the place to put semi-permanent information.
- The ESC A to ESC D codes are normally generated by the Apricot cursor keys.
- ESC H is normally generated by the HOME key.
- ESC A and ESC B will not scroll the screen as do their counterparts ESC I and LF; ESC C and ESC D will not ' wrap' to the next or last lines.
- ESC C offers one oddity on certain F series BIOS versions — it won't go all the way to the right-hand edge of a screen window (including the default screen) but instead stops one space short. Indeed, if the cursor is otherwise positioned at the right-hand edge and ESC C issued, it moves you one space *left*!
- The normal status line is usually referred to as line 25 of the screen, but for the sake of logical consistency it seems wise to think like programmers and count from zero. Thus we number the screen rows from 0 to 24, the columns from 0 to 79. This makes the format for the ESC Y command easier to understand: we don't send the *actual* row and column number but add 32 to each, presumably to avoid the possible confusion of having to send control characters (0 to 31) as data. So ESC Y followed by two spaces (character 32 [20]) would move the cursor to row 0, column 0 — the 'Home' position. ESC Y will not move the cursor to the status line unless this line has previously been 'enabled' — see the 'Environment' section in next month's installment.
- As noted above, ESC Y commands which move the cursor to row and/or column 4 require the sending of character 36 [24], which is the dollar sign. Beware of the MS-DOS 'display string' function in Assembler.
- ESC h should have no effect when issued at the beginning of a line, but does weird and wonderful things on certain F series graphics screens (like the F10 with BIOS R1.6): the cursor can actually move to a position *between* lines.
- 'Home' is where ESC k will put the cursor if no previous position has been saved with ESC j. Each new ESC j saving of cursor position wipes out the previous one; only *one* can be stored.
- A useful multiple sequence:

  ESC j ESC ; MESSAGE ESC k

  This will put the 'MESSAGE' text on the status line without messing up the main screen cursor position. (But see the caveat under 'Windows' following.)

Erasure and deletion

The difference here is that erasing text equates to overwriting it with spaces, while deletion implies 'closing up' the surrounding text.

There is no 'erase current character' escape sequence: sending a space to the screen performs this service.

ESCape	Dec	Hex	Function
ESC +	43	[2B]	erase all high-intensity characters (not F series)
ESC -	45	[2D]	erase all low-intensity characters (not F series)
ESC E	69	[45]	clear screen and home cursor
ESC J	74	[4A]	erase from cursor position to end of screen
ESC K	75	[4B]	erase from cursor position to end of line
ESC M	76	[4D]	delete current line and scroll up any below
ESC N	77	[4E]	delete current character and close up from right
ESC b	98	[62]	erase from start of screen to cursor position
ESC l	108	[6C]	erase current line, preserve cursor position
ESC o	111	[6F]	erase from start of line to cursor position
ESC z	122	[79]	reset screen to 'power-on' state and home cursor

Notes:

- ESC + has a bug in Xi BIOS 3.1 as supplied for MS-DOS 2.11, and clears low- rather than high-intensity characters. It and ESC - are generally to be avoided, since their use requires that your program be completely rewritten for the F series.
- ESC E has an important variation on the Xen: on other models, ESC E issued while the cursor is on the status line will simply clear the status line, but on the Xen it clears the *main* screen leaving the status line untouched. This sequence is generated by the CLEAR key on an unmodified Apricot keyboard.
- ESC M is supposed to reset the cursor to the left margin, but in BIOS 3.1 and most of the oddly-numbered F series BIOS variants it doesn't: it's advisable to follow this sequence with a Return for complete cross-compatibility.
- As with ESC E, ESC M fails when you try to use it to clear the Xen status line (a main-screen line vanishes instead). ESC l is the safest 'generic' way to clear the status line once you're there.
- ESC J, ESC K, ESC b and ESC o include the character at the cursor position in their destructive work, and do not move the cursor.

- All these sequences work within the bounds of the current screen window only (see below), with the exceptions of ESC +, ESC - and of course ESC z.
- ESC z not only blanks the screen but homes the cursor, turns off all special 'attributes' (see below), and generally clears everything you've done except for any saved 'environment' (in next month's episode). Colour settings provide further exceptions: the overall display mode is unaffected, and likewise the 'palette'.

Inserting text characters and lines

These functions are generally less useful than deletion and erasure, although ESC L is occasionally handy. The Apricot's normal 'overwrite' mode is much easier to keep track of, especially since careless insertion of characters into a text line may mean the loss of characters at the right.

ESCape	Dec	Hex	Function
ESC @	64	[40]	set insert mode: text is henceforth shuffled right to make a space for each new character sent to the screen
ESC L	76	[4C]	insert blank line (scrolling rest of screen down)
ESC O	79	[4F]	turn off ESC @ insert mode
ESC P	80	[50]	insert a blank space at cursor position (by moving text right)

Notes:

- The ESC @ insert mode slows the display badly — even when you're writing text into a 'blank' line, the BIOS wastes time conscientiously moving spaces to the right and off the edge of the screen.
- ESC P does not insert an underlined, inverse-video or struck-out space, even when these attributes are set (see below).

All about text attributes

'Attributes' refer to the way screen characters are presented — bright/dim, underlined, reverse video, and the like. Thus when the underline attribute is set, all characters appear underlined until further notice. One pair of escape sequences I wish were included in the BIOS would be to save and restore the current attributes ...

ESCape	Dec	Hex	Function
ESC (	40	[28]	set high intensity display
ESC)	41	[29]	set low intensity display
ESC 0	48	[30]	set underline mode
ESC 1	49	[31]	turn off underline mode
ESC 9	57	[39]	set strikeout mode
ESC :	58	[3A]	turn off strikeout mode
ESC _	95	[5F]	set current attributes over entire Xi screen
ESC p	112	[70]	set reverse video mode
ESC q	113	[71]	turn off reverse video mode

Notes:

- All these fail when 'full-colour' display mode has been set on an F series monitor, but can be made available by switching to two-colour 'Xi monochrome compatibility' (see next month's 'Colour' section).
- For the F series, 'high' and 'low' intensity mean not 'bright' and 'dim' but 'thick' and 'thin' characters, with the former usually being rather illegible.
- Strikeout mode, with a horizontal line running through the text characters, is a concession to some US legal oddity whereby documents sometimes need to show supposedly struck-out text.
- Underline and strikeout on the Xi are dependent on the character font: the positions of these lines are set separately for each character via undisplayed font-cell bits, leading to messiness if (as in certain Apricot Black Hole fonts) these bits are misaligned or omitted.
- ESC _ worked on the Xi only, was never very useful, and has been dropped as of BIOS 3.1. (Flipping the whole screen into reverse video with a single command was fun, but this effect was cocked up as soon as the screen was scrolled.)
- The ESC z screen reset cancels all current attributes and leaves the display 'dim'; ESC E and partial screen-clear functions leave attributes unchanged.

Windows, scrolling, and wrapping

ESCape	Dec	Hex	Function
ESC ,yYxX	44	[2C]	set screen window size (top row y-32, bottom row Y-32, left column x-32, right column X-32) and home the cursor within the window
ESC .	46	[2E]	restore screen to default 80 columns and 24 rows (plus status line) and home cursor
ESC Qn	81	[51]	scroll current screen window left by (n-32) columns
ESC Rn	82	[52]	as ESC Qn, but scroll right (n-32) columns
ESC Sn	83	[53]	as ESC Qn, but scroll up (n-32) rows
ESC Tn	84	[54]	as ESC Qn, but scroll down (n-32) rows
ESC v	118	[76]	wrap text at end of each line (default mode)
ESC w	119	[77]	don't wrap at the end of a line

Notes:

- When a window is set, the defined area of the screen is treated as being the whole screen for all purposes of display, scrolling, clearing, etc. An ESC z reset or (F series) ESC 7 colour mode change will however restore the default full-screen window.
- Only one window is allowed, though multiple non-overlapping windows can be handled by adroit juggling of ESC , commands.
- Neither the setting nor the resetting of windows erases any screen text, provided ESC z isn't used for resetting.

- I've found ESC , one of the more dangerous screen controls in which to place your trust. In some BIOS versions, silly values for the parameters can hang up the computer ... though the system is tolerant of over-large values for the right and bottom window edges, and it will quietly set column 79 and /or row 23 when thus provoked (counting from zero, remember).
- Xi BIOS 2.4 (presumably along with earlier versions) gets the right-hand edge of the window wrong, making it one column narrower than specified.
- Some F series BIOS versions, like the F10's version R1.6, will not allow the setting of windows that are only 1 or 2 units either high or wide: the command merely homes the cursor in the existing screen.
- Those parts of the screen outside the current window are generally inaccessible and inviolate: ESC E will clear only the interior of the window, ESC H will home the cursor to top left of the window, and so on. (See 'Erasure and Deletion' above for some exceptions.)
- The BIOS tries to provide a status line (accessible as usual with ESC ;) for any window. Such *ad hoc* status lines can sometimes appear in odd places, however. Mark Fisher points out a bug here: if the window doesn't begin at the left edge of the screen, the command ESC ;MESSAGE will put the word 'MESSAGE' at the left edge of the screen rather than at the left of the status line (which is aligned with the window). This is curable by sending a Return to the screen immediately after the ESC ; command.
- When using ESC Y to move the cursor within a window, you should still use 'absolute' row and column numbers, counting from the true top-left corner of the screen — not of the window. Attempts to position the cursor *outside* a window tend to leave it at the window's edge.
- For ESC Q, R, S and T some Apricot documentation incorrectly states that the screen is scrolled by (*n*-31) rather than (*n*-32) units. Nothing at all happens when the character for *n* is a space (ASCII 32 [20]). These four functions scroll text harmlessly off one edge of the screen and introduce new blank rows or columns at the opposite edge. (See 'Environment' for an additional control over the appearance of vertical scrolling on the F series.)
- 'Wrapping' with ESC v and ESC w means that when a character has been printed at the end of a line, normally column 79, the cursor moves to the start of the next line (with screen scrolling if need be). With this feature turned off, the cursor 'sticks' when it reaches the end of the line — subsequent characters overwrite each other until a Return and linefeed are issued, or wrapping is turned on again, or whatever.

Graphics, fonts and funny characters

ESCape	Dec	Hex	Function
ESC *a	42	[2A]	select font 0 (ESC *0), 1 (ESC *1) or 2 (ESC *2)
ESC 8n	56	[38]	ignore any control function of character n and print the appropriate font character
ESC F	70	[46]	turn on VT52 graphics mode
ESC G	71	[47]	turn off VT52 graphics mode
ESC iyx	105	[69]	display logo at stated position
ESC }	125	[7D]	set Xi graphics mode (50x25-cell screen)

Notes:

- ESC *a (not available on the F series) allows instant swapping between two other fonts besides the default font 0 loaded at 0:800H. You need a bit of programming skill to take advantage of this, since Apricot has provided no explicit means of loading font 1 at 0:2800H — let alone font 2.
- ESC 8n can be used to show any character, but its only real point is to allow display of the normally inaccessible control characters in the ASCII code range 1 to 31.
- ESC F, switching on 'VT52 graphics', also allows oddball characters to be displayed; but it mucks up the entire character set: ordinary text letters become unavailable, a high price to pay for the accessibility of characters 1 to 31 — which in this mode appear when you try to display characters 95 to 125. (And 'VT52 graphics' is a misnomer anyhow — it's not graphics at all.)
- Note that there's no displayable character in the default font corresponding to ASCII codes 0 (null) and 255 (FF — formfeed).
- ESC iyx (not available on the F series) displayed the attractive 'Apricot' logo, or any other that you might have installed in the dear dead days of LOGOEDIT and SETUP, with its top left corner at a position calculated by interpreting the y and x parameters exactly as in ESC Y cursor positioning. The function being nifty, popular, and good for impressing punters, it was naturally dropped from BIOS 3.1.
- I looked further into the possibilities of ESC } in an APRICOT FILE column subtitled *Graphics without GSX* (*Release 1.6 — we have back issues available at £4.00 each. Ed*). Used alone, it makes a right dog's dinner of the Xi screen — MS-DOS doesn't sense that lines are now 50 rather than 80 columns wide, and puts its prompts in all sorts of weird places. A mildly interesting point is that the wider display cells allow you to see the normally invisible 'underline' and 'strikeout' position marker bits at the right of each character. ESC Z will restore sanity.

Part Two next month — the remaining ESCape codes. Includes keyboard and printer specials; altering the cursor's appearance; driving the MicroScreen; and miscellaneous leftovers.

User Experience

The Worst Hundred Quid I Ever Spent

Books I shouldn't have bought (and a few that were worth it)

By Eddie Bromhead

Following a quick read of David Langford's enthusiastic review of Peter Rodwell's *Advanced User's Guide to the Apricot* in APRICOT FILE a while ago (Release 1.12), I rushed round to my local bookshop in Kingston and ordered a copy. A month later, my deposit was returned: "The publisher has cancelled it". A sarcastic letter to APRICOT FILE didn't help. After blaming DL for indulgence in science fiction (though I can recommend his *Third Millenium*), humble pie had to be eaten: it was my bookstore that cocked it up!

I get phases of book buying. This little fiasco set me off on a couple of sprees. I liked Rodwell's book, eventually bought for £15 from DEALS FOR READERS, and it whetted my appetite for more. This note is the result. Oddly enough, the price for virtually every one of the volumes was a little under £20 — my own publisher told me that hardbacks cost £40 per inch of thickness; and you get the same in softback, or half as much again, depending on how glossy it is.

GEM vs Windows revisited

Had I got it wrong with my article in APRICOT FILE Release 1.10 comparing GEM and Windows? I finally got hold of a working copy of GEM Desktop (it only took my ex-dealer a little over a year), and it installed fine on my new Xen HD — even though the GEM installation batch files *still* don't work on twin-floppy machines! And GEM is certainly a lot smaller and faster than Windows.

My need for a WIMPS system lies in two areas: preparing masters for overhead-projector transparencies, and a possible use for multi-font text of moderately high quality including formulae and so on. Basically, the former is needed whenever I wear my lecturer's hat; the latter needs the scientific word processing package that I'm too mean to buy.

My book purchases included Kathy Lang's *Getting it Done with GEM* (Heinemann), and Nancy Andrews' *Windows* (Microsoft Press). Beautifully produced, the Windows book reflects the generally smoother style of Windows compared to GEM. No information whatsoever for the programmer, alas, just the bits left out of the user guides provided by Apricot (as well as everything they included).

It's a 'Gee whiz, isn't Windows wonderful' book, let down really by the lack of applications that actually run in the Windows environment. After the Desktop applications, all that she could find to write about in depth were the IN*A*VISION computer-aided draughting package, and a little over a paragraph on each of five other applications (including PAGEMAKER and Palantir's FILER database). Just so much 'vaporware' (sic), I guess ...

I learnt two things from this book. First, I need no longer waste time trying to set a page size — it's fixed by the printer driver. And second, I got a few hints on installing printers. Not a lot of value for twenty quid; but the book always invites envious comment from colleagues whose BBCs don't run Windows, so I leave it conspicuously on display.

Lang's sign

Kathy Lang's alternative shows less graphic design, and a more hurried writing period. She has used the 'snapshot' facility in GEM to fill her book with captured full and partial screens. Often the choice of illustrations reflects enthusiasm, rather than judgement: she obviously had fun (and access to a laser printer) drawing this book!

Kathy Lang has used the post-Apple-lawsuit version of GEM as her model: this differs in detail from the earlier version (as supplied to me — and, for that matter, still purveyed under the name 'GEM Support' with Xens).

One thing I couldn't find was a means of making GEM show black on green like Windows, and some of the facilities she describes appear to have been removed (or never provided) with the Apricot version. Now where have we seen that before?

Again, two interesting things were learnt: for one, GEM has two printout modes — one using bit-image graphics (for graphics alone, or mixed text and graphics), the other a simple print spooler. This enabled me to get hard copy from GEM Paint. Secondly, there was a good account of the SNAPSHOT facility. Again, not exactly good value for £20!

Oddly enough, even though Digital Research seems to have many more applications for GEM than Microsoft has for Windows, Kathy Lang can't find any non-DR products to describe or even list.

Old friends in new clothes

Even odder was my purchase of the Glentop *GSX Manual*. This is basically a reprint of the Digital Research manual as supplied by Apricot at — you guessed it — £20; and Glentop's price ... well, do

you need to ask? It is complete with the same typographic errors as the DR manual, although obviously reprinted (I needed a second copy, and couldn't face the slog at the photocopier, never mind the copyright laws).

A friend of mine observed: "That's just like my *GEM Programmer's Toolkit Manual*". And so it is. And that leads further and further into GEM ...

While trying out some more of the facilities in GEM, I double-clicked (note the WIMP-speak) on to one of my own programs. Damn, thinks I, that won't run because GSX isn't loaded. Surprise, surprise, it did.

The picture came out fine, but it flashed off the screen when it came to the bit where it is supposed to wait for cursor input. Actually the program gets the cursor position twice, and then draws that part of the screen between the two crosshairs to an enlarged scale. If the two pairs of coordinates are the same, it assumes 'end of pictures' and goes into text mode.

It wasn't a problem with the way I called GSX, because it happened with other programs — some of which used the HWa disk utilities to call GSX. And changing the interrupt from GSX's 224 to GEMVDI's 240 made no noticeable difference. Finally, I tried it on a Amstrad PC with GEM. Again, no cursor input.

Hotfoot I went to borrow the *GEM Programmers Toolkit Manual*. (A charitable friend had one; his sister's name and mine were once linked publicly in connection with a long-lost 1920s Channel Tunnelling machine — but that's *completely* another story!)

The GEM manual is far thicker than the GSX equivalent; and if anything it's even more impenetrable. You soon discover however that GEM appears to have two components. The first is a GSX replacement called GEMVDI; if you can program for GSX, you ought to be able to program for GEMVDI.

Using GEM

In fact there appear to be only minor differences between GEM and GSX. There's another CONTRL array element for a 'handle'. GEM keeps the screen driver resident at all times, and just swaps hard-copy drivers in and out. It also supports multiple screen overlays ('virtual workstations') as a method of managing the dialogue boxes and suchlike gimmickry. For the user there's a handful of new facilities, best of all one can use coordinates specified in pixels or plotter steps as well as the NDC coordinates so beloved of GSX.

Indeed, if you had a late-model Apricot Xi you probably got a file called GSX.DOC in the issue software. This details a number of the extra facilities that exist in GEM.

By the way, running a program using GEMVDI doesn't need GEM Desktop. GEMVDI can be loaded standalone in the same way as GSX; or with the following syntax, it can load itself, run your GSX application, and unload on completion:

```
A> GEMVDI /whatever
```

Back to the bookstore: "Anything on GEM? For programmers?" I emerged clutching Balma and Fitler's *Programmer's Guide to GEM* (Sybex). They forgot to add 'and C' to the title.

It seemed to be in DR manualspeak — not surprising, they are both ex-Digital Research — and the book demands not only fluency in C, but also access to the C programmer's toolkit codes. It concentrates on the AES Application Environment Services, the second part of GEM and the one which is bound into your applications. The AES handles all the drop-down menus, screen sizing and windows ... but not the underlying graphics drivers.

No help there for the lost cursor, and another £20 down.

(Incidentally, where does one look for a PC with Apricot VT52 escape codes? No further than Amstrad! DOSPLUS is none other than Concurrent CP/M-86 cut down, dressed up, and fired up to run all your Apricot software. Now isn't that fine?)

Assembler programming (or not)

At the same time as I was ordering Rodwell's book, I asked for the Rollins book on Assembler — also recommended by David Langford. This also didn't turn up either. So in lieu I picked up a handful of others: Coffron's *Programming the 8086/8088* (Sybex), Bik Chung Yeung's *8086/8088 Assembly Language Programming* (published by Wiley — is he a Wiley Oriental?) and Erskine's *Assembly Language for the 8086 and 8088* (Pitman).

James Coffron must be a computer: he certainly doesn't write English like a human. Erskine's handbook is fine, if you already speak Assembler and need a quick reference. Yeung not only has a Sirius, but can write code that is so Sirius-specific it won't run on an Apricot ... Maybe I should have waited for Rollins.

While in the bookshop I picked up two other tomes with some reference to Assembler. I can heartily endorse Richard Startz' *8087 Applications and Programming for the IBM PC, XT and AT* (Brady); he explains more about interfacing to a high-level language than any text I've seen.

He uses MS-BASIC, so he's not perfect; but his book is both simple and comprehensive. One begins to understand Assembler after a chapter of this one. Not only that, but he provides double-precision functions (missing from Basic) so that the language is useable for number-crunching —

though only with an 8087, of course. (As an aside, the US Library of Congress index reference cites it as 'Eight thousand and eighty seven applications and programming etc'. I wonder if anyone was disappointed there weren't that many?)

Buy this one if you want to write simple Assembler subprograms to interface to your favourite high-level language (or if you just want to understand them) especially if that language doesn't let you get at MS-DOS functions directly.

Ray Duncan's *Advanced MS-DOS* (Microsoft Press) is another book that was written to be *read*. I'd already seen a review of it that suggested it's complicated: it isn't, if you read it through — it just seems so if you dive in at the middle.

He's really writing for PC-DOS, but he clearly identifies the IBM-specific pieces of code. So this is the one to read if you want to know what MS-DOS actually *does*. The fruits of my reading of this book will eventually appear in APRICOT FILE, I hope ...

ACTWritings

Finally, another friend lent me two volumes of the *ACT Dealer Handbook*. Amazing! Ever want to know the escape codes for a particular printer? Or the codes to drive an ACT Writer 80 or 81 plotter? Or how to patch WordStar's installation program so it doesn't say 'insufficient memory' if you've got more than 512KB? It's all there.

And you can find out about IBM emulation, and character sets, and Pulsar accounts, and SuperCalc or SuperWriter XQTs ... and a thousand and one things that must interest someone somewhere. Maybe its the dealers who are wallies, and not those Brummie lads.

By the by, the patch for WordStar INSTALL isn't necessary — there are two other fixes. One is to run it under Concurrent CP/M (it's just WordStar itself that doesn't). Or you can load up a RAMdisk that leaves less than 512K free. Both of those allow you to let sleeping dogs lie ...

For anyone who is interested in IBM emulation, there have been a number of tries. The first ran under MANAGER (remember MANAGER?). The latest is a standalone program called EMIBM.EXE; it seems to work tolerably, if you also load alternative keyboards and so on. One warning, though: the emulator gives you a 'bare' IBM with no ANSI.SYS, so use escape sequences at your peril. *(We have scheduled a Workshop on Apricot's IBM emulators for the next issue, written by an ex-Apricot insider — Ed.)*

Booklist

ADVANCED USER'S GUIDE TO THE APRICOT
by Peter Rodwell
published Heinemann 1986
price £14.95 [or £15 inc P&P from DEALS FOR READERS]
ISBN 0-434-91744-3

GETTING IT DONE WITH GEM
by Kathy Lang
published Heinemann 1986
price £16.00
ISBN 0-434-91067-8

WINDOWS
by Nancy Andrews
published Microsoft Press 1986
price £14.95
ISBN 0-914-84570-5

GSX MANUAL
by Digital Research
published Glentop 1986 [dated 1984 by DR]
price £19.95
ISBN 1-851-81056-0

GEM PROGRAMMER'S TOOLKIT MANUAL
published Digital Research 1985
price £400, but discounted [from DR dealers — in theory]

PROGRAMMER'S GUIDE TO GEM
by P Balma and W Fitler
published Sybex 1986
price £19.95
ISBN 0-895-88297-3

PROGRAMMING THE 8086/8088
by James Coffron
published Sybex 1983
price £16.95
ISBN 0-895-88120-9

8086/8088 ASSEMBLY LANGUAGE PROGRAMMING
by Bik Chung Yeung
published Wiley 1984
price £11.95
ISBN 0-471-90463-5

ASSEMBLY LANGUAGE FOR THE 8086 AND 8088
by R Erskine
published Pitman 1985
price £3.95
ISBN 0-273-02245-8

8087 APPLICATIONS AND PROGRAMMING
by Richard Startz
published Brady 1985
price £19.95
ISBN 0-893-03485-1

ADVANCED MS-DOS
by Ray Duncan
published Microsoft Press 1986
price £19.95
ISBN 0-914-84577-2

ACT DEALER HANDBOOK
two volumes
published ACT/Apricot plc 1983, with subsequent loose-leaf amendments
[not normally available, but try a friendly dealer]

Footnote

Not so SmartPath — try it on MS-DOS 3! ◆

Doodles by David

Tips and Tricks

By David St.John Wallis

Now I've got this theory about power supply units and humidity: the two do not go together. Running as I do to two openable windows and, now, a humidity meter, it is discovered to my great cost that the delinquent purchase of a power supply unit from Apricot Computer Maintenance — tel 021-552 1555 — was a mad escapade, a snare and a delusion. When the humidity was replaced by routine English weather, the flickering screen, the queer read/write errors, the odd snafus, etc — just went away.

Those in the market for a nice Apricot psu need look no further. My old one is ok after all.

Rumour hath it that Apricot designed a capacitor into the psu which de-solders itself after 24 months. Replacements cost £45 plus VAT and postage.

The Truth

Rumour also hath it that Apricot people lie through their teeth when they say they haven't made Apricots since last year. The truth is that they made a heck of a lot for Quest only a few months ago.

It also appears to be true that Dixons have just taken delivery of another 300 or so Xi-FDs and Xi-HDs, if you can remember what these were *(PC and Xi respectively — Ed).*

Disk buffer

MicroCache has come out with a write-to-disk buffer. It's weird to use if like me you bung in a disk, do a quick COPY *.* B:, and then find the computer does the job very quickly so that you can get on with something else. Because, when you get the 'nn file(s) copied' message and then take the disk out, you may get a message some time later which says 'disk in drive B not ready'.

The reason is that the program has copied into *memory*, not on to disk, and it's waiting for you to plant out the liquidambers *before* writing to disk!

Ho hum. But you get used to it. And if you've paid Microcosm Research £195 (phone 01-226 9092), you get to like MicroCache after a bit. Or not.

Some stats

If you *use* your spreadsheet, you can get some quite useful statistics out of it to help you decide if you're operating over or under budget, working efficiently, buying the right pilchards, etc.

I am buying a new house at the moment (well, quite new: we think Thomas à Becket's in the cellar). And I thought it would be nice to know to what extent we went over or under budget on all the bits and pieces outside the routine mortgage capers. So I made a spreadsheet which listed the various things we spent money on and then flagged them for analyses. These are the results to date:

	Budget	Spend to date	Over/Under
Services	50%	46%	− 4%
Garden	20%	16%	− 4%
Tools	10%	10%	0
Travelling	5%	1%	− 4%
Hardware	5%	21%	+ 16%
Valuation	5%	4%	− 1%
O&S	5%	2%	− 3%

Moral: 'Tools' includes all the those Phillips screwdrivers you thought you'd never need again which have now rusted away in the garage; not to mention chain saws, cellar pumps, wrong lawnmowers, bluetit nut-hangers and strange leather gismos bought on a wild spree in Amsterdam.

SoapBox fans

The sumptuously-laurelled Editor says that people either love or hate my stuff. And that's probably because he tones down my tone; *what you see is not the genuine article.* What you see is the Ed's version of my drivel. Or vice versa. Bet this didn't happen to Walter Cronkite or David Langford *(it did and does — Ed).*

He goes on that my column, which *he* subtitled 'Tips and Tricks', should contain a few. So (and I *dread* this — because there may be a reader who knows it already) I remember being thrilled to bits when someone told me that you can have a font — that's the type style which appears on your screen (groan) — which was not of Apricot's devising. You can have Script, Roman, Medium, OCR or my favourite 'NS' — presumably for 'non-serif'. There are *lots* more too.

You can even *use* these fonts, in batch files for example: the mere sight of the font will remind you, after a bit, what batch file you're running. For example — my daily back-up batch file uses font ITALIAN. My CLOSE.BAT, which gets rid of things like *.$??, *.PRN and *.BAK files uses font FUTURE (boy, that's a weird one). And you can switch back and forth between fonts in the middle of a batch file if you're that ready for Broadmoor.

How to do it? First, get a set of fonts; APRICOT FILE has some in the DEALS FOR READERS lists, files which all end in '.CHR'. When you've got them on your disk which might also contain FONT.EXE, type 'FONT FONTNAME' (leave out the .CHR suffix) at the DOS prompt — and the pleasure, esoteric as it may be, will be all yours.

PS: Kuma's OTHELLO — which is a whole lot better than anything else I've seen — produces a strange non-reproduceable and angular type face on screen. To get it, call Kuma on 0735 74335. ◆

The Workshop

Splitting the Infinite

How to chop up your big ASCII files

By David Lewis

I *like* SuperWriter; but I'm frustrated by its inability to edit a text file with more than 33,000 or so characters (to be exact, 33,296 with Version 1.03 and 33,232 with Version D1.09).

And ever since I became interested in communications, I've often needed to edit files larger than that. After three quarters of an hour on-line to Microlink, for example, I'll have created a log file of everything that happened on screen which is well in excess of SuperWriter's workspace limit. I usually like to edit these downloaded files, mainly to delete all the unwanted material and to save or print the relatively small amount of useful text.

But SuperWriter is the only word-processor I've got — apart from Quill (one of Psion's Xchange suite of programs), which *can* theoretically handle large files but which loads slowly and deletes big chunks of text even more sluggishly.

Of course, there's always EDLIN. I don't share the common antipathy to EDLIN, but I have found it unsuitable for editing big files. Deleting several hundred lines — a simple task in SuperWriter — is a lengthy chore in EDLIN because of the time it takes to scan numerous successive screens (not to mention the oddly unreadable way that long lines are displayed). And it is almost as difficult using EDLIN to split a big file into sub-files small enough to be loaded into SuperWriter; EDLIN too is limited in what it can hold in memory, and you have to use the WRITE and APPEND commands — either way, it is easy to make mistakes.

So I wrote SLICE.BAS to solve the problem. I run it to cut files down to SuperWriter size, but you may find other uses for it.

Suppose you've saved a lengthy session on Telecom Gold or Microlink under the name LOG.DOC. The file is 76,006 bytes in size, 42,710 more than SuperWriter can handle. On the basis that SLICE.BAS splits files into roughly equal parts, you will need to divide LOG.DOC into three sub-files each of about 25,300 characters. SLICE.BAS will do this by creating three files — LOG.1, LOG.2 and LOG.3 — leaving your original LOG.DOC intact.

After editing each of the new files, there are of course many ways of printing them as a composite using SuperWriter or MS-DOS. I usually COPY them into a new file and PRINT that:

```
COPY LOG.1+LOG.2+LOG.3 LOG.0
PRINT LOG.0
```

SLICE.BAS was written under GWBASIC, but with one minor amendment (shown in the listing) it will also run under MSBASIC. ◆

Listing for SLICE.BAS

```
100  REM  *********
110  REM  SLICE.BAS
120  REM  *********
130  REM  A program in GW-BASIC or MSBASIC
140  REM  by David Lewis, London (1987):
150  REM  Slices file into two or more roughly equal parts
160  REM  Leaving original file intact
170  REM
180  REM  ************************************
190  REM  SLICE WHICH FILE INTO HOW MANY PARTS?
200  REM  ************************************
210  ON ERROR GOTO 950
220  DEFINT A-Z
230  CLS:       REM Use PRINT CHR$(27)+"E" if using MSBASIC
240  INPUT "Enter file name with extension: ", INFILE$
250  IF LEN(INFILE$) < 1 OR LEN(INFILE$) > 12 THEN PRINT CHR$(7): GOTO 240
260  IF INSTR(INFILE$," ") THEN PRINT CHR$(7): GOTO 240
270  LET DOT = INSTR(INFILE$,".")
280  IF DOT AND LEN(INFILE$)-DOT > 3 THEN PRINT CHR$(7): GOTO 240
290  IF DOT THEN LET DOTX$ = MID$(INFILE$,DOT) ELSE LET DOTX$ = ""
300  IF DOTX$ = ".COM" THEN PRINT CHR$(7): GOTO 240
310  IF DOTX$ = ".EXE" THEN PRINT CHR$(7): GOTO 240
```

```
320  PRINT: INPUT "Slice into how many parts?  ", PARTS
330  IF PARTS < 2 OR PARTS > 999 THEN PRINT CHR$(7): GOTO 320
340  IF DOT = 0 THEN FILENAME$ = INFILE$
350  IF DOT THEN FILENAME$ = LEFT$(INFILE$,DOT-1)
360  REM
370  REM  ***********************************
380  REM  WORKING OUT HOW MANY LINES PER PART
390  REM  ***********************************
400  PRINT: PRINT "Counting number of lines in "; INFILE$; ":";
410  LET INFILE = 1
420  LET OUTFILE = 2
430  OPEN "I", #INFILE, INFILE$
440  INLINES# = 0
450  WHILE NOT EOF(INFILE)
460    LINE INPUT #INFILE, INLINE$
470    INLINES# = INLINES# + 1
480    WEND
490  CLOSE #INFILE
500  PRINT INLINES#: PRINT
510  PARTLINES# = INLINES#/PARTS
520  REM
530  REM  ************************************
540  REM  SLICING FILE INTO ROUGHLY EQUAL PARTS
550  REM  ************************************
560  OPEN "I", #INFILE, INFILE$
570  READLINES# = 0
580  FOR PART = 1 TO PARTS
590    LET OUTLINES# = 0
600    TARGET# = PART*PARTLINES#
610    PRINT "Writing to file "; FILENAME$; "."; MID$(STR$(PART),2); ":  ";
620    OPEN "O", OUTFILE, FILENAME$ + "." + MID$(STR$(PART),2)
630    WHILE NOT EOF(INFILE) AND READLINES# < CINT(TARGET#)
640      LINE INPUT #INFILE, INLINE$
650      READLINES# = READLINES# + 1
660      PRINT #OUTFILE, INLINE$
670      LET OUTLINES# = OUTLINES# + 1
680      WEND
690    PRINT USING "########"; OUTLINES#; : PRINT " lines.";
700    PRINT TAB(50); "Total so far:  ";
710    PRINT USING "########"; READLINES#; : PRINT " lines."
720    CLOSE #OUTFILE
730    NEXT PART
740  PRINT: PRINT INFILE$; " has been sliced into"; PARTS; "sub-files"
750  PRINT CHR$(7)
760  CLOSE #INFILE
770  REM
780  REM  ******************
790  REM  SUBSEQUENT CHOICES
800  REM  ******************
810  PRINT "Press:
820  PRINT "  B  to go into BASIC command level"
830  PRINT "  S  to save this program SLICE.BAS": REM This line is not essential
840  PRINT "  X  to exit from BASIC into MS-DOS"
850  PRINT
860  LET PRESSED$ = INPUT$(1)
870  PRINT PRESSED$: PRINT
880  IF PRESSED$ = "S" OR PRESSED$ = "s" THEN SAVE "SLICE", A: REM This line is not essential
890  IF PRESSED$ = "X" OR PRESSED$ = "x" THEN SYSTEM
900  END
910  REM
920  REM  ********************************
930  REM  ERROR-TRAPPING IF FILE NOT FOUND
940  REM  ********************************
950  NOFILE = ERR = 53 AND ERL = 430
960  IF NOFILE THEN PRINT: PRINT: PRINT CHR$(7); "File not found ...": RESUME 240
970  ON ERROR GOTO 0
```

Inside MS-DOS

Department of the Environment: SET, COMSPEC and more

By Roger Gann

Got a minute? Type 'SET' at the A> prompt. You should get something like this:

```
COMSPEC=A:\COMMAND.COM
PATH=D:\SYSTEM;D:\PROGRAMS;D:\CARDBOX
PROMPT=$e($n$g$e)$ej$e;$ep Status :
$e@$e($p$e) - $t$h$h$h$h$h$h - $d - $v
$eK$ek$eq$e0
```

Well, this is what I see when I try SET on my own PC. A couple of things it displays you ought to recognise immediately: PATH simply restates the directory path search I have specified. PROMPT together with the associated mumbo-jumbo produces the rather convoluted 25th line prompt that I employ.

Nothing extraordinary here, you might say. But what about COMSPEC? This is MS-DOS vernacular for 'Command Specifier'; it merely confirms that the command line interpreter in use is good ole COMMAND.COM, which can be found on A:\ (meaning the root directory of drive A).

This is not, however, the end of the matter.

Secrets of the environment

What we have been examining above is none other than the contents of my 'environment' on my Apricot PC. And what is the environment? It's a tiny area of memory, a mere 256 bytes, in which you can place nuggets of systemwide information that can be accessed by any program or batch file. It's an MS-DOS pigeonhole, if you like.

You may already have come across the mysterious term 'environment'. If you get a bit too ambitious and overdo the length of your prompts and paths, you will be curtly told that you're 'out of environment space'. You have soaked up all 256 bytes there. So if you want to make good use of the environment space, you'll have to be quite sparing in your use of this precious resource.

The trouble is, not many people know much about the environment — and fewer know how to put it to any use. To my knowledge there is only one application that embraces the environment concept, the programming language C with its GETENV and SETENV functions. You certainly won't find any mention of it in the MS-DOS manuals supplied with your Apricot — and Microsoft refers to it as "an undocumented feature". It is only by plundering various IBM DOS reference manuals that the facts surrounding the mysterious environment can be elicited: the paucity of use probably has a lot to do with the half-hearted implementation of the environment in MS-DOS.

MS-DOS versions 2.nn and 3.nn are completely different operating systems from version 1. The latter was nought but an imitation of CP/M-86; MS-DOS 2 was heavily modelled on Unix, though, and so offered many new Unix-style facilities such as subdirectories, dynamic I/O redirection, piping, filters ... and the environment.

Unix naturally boasts all the necessary features which enable users to put the environment to proper use. For instance, as a true multi-user system, Unix permits details of file-locking to be passed to the environment: other users must then consult that before a file can be updated. MS-DOS programs are denied access to the environment proper, and can only access a copy of it.

Expanding the environment

And the space available is small, as little as 224 bytes. The version of COMMAND.COM supplied with most Apricots sadly does not support the /E switch which permits you to expand the environmental space on other implementations; but that *is* available in the COMMAND.COMs supplied with MS-DOS 3.1 and 3.2 (and it's also provided in the Victor Technologies COMMAND.COM on DEALS FOR READERS disk AP020). To expand your environmental space you type COMMAND /E:*nnn*, where *nnn* is the desired size in bytes of environment.

It is however possible to patch your standard MS-DOS 2.11 COMMAND.COM to give it an enlarged environment space. With DEBUG in one hand and a spare copy of COMMAND.COM in the other, enter the following commands:

```
DEBUG COMMAND.COM [Return]
-S 100 4600 BB 0A 00 B4 48 CD 21[Return]
```

The S command searches for the call to allocate space for the environment area; DEBUG should come back with the location of the desired string in the format XXXX:0DF2. Disregard the first four digits (the 'segment address') as this will vary from machine to machine. You should then enter D 0DF2 to display the contents of locations starting at that address, and the first line of the display should include this ...

```
XXXX:0DF2   BB 0A 00 B4 48 CD-21 8E 1E 25 0A A3 25 0A
```

The next step is to replace the first 0A with the new area size factor (in paragraphs). Deft use of the U-for-Unassemble command will reveal the following:

```
BB 0A 00   MOV BX,0AH
B4 48      MOV AH,48
CD 21      INT 21
```

The 0AH in the first line is the 'area size factor'; it's calculated as (A+4)*16, with A being the area factor. The default value of 10 decimal — 0A hex — yields an environment area of 224 bytes; increasing the value to 20 decimal (14 hex) will give an environment area of 384 bytes, 30 decimal (1E hex) will provide 544 bytes, and so on.

So to increase the environment space to 544 bytes you would then enter ...

```
-E 0DF3 1E
```

That's it. You can now Write the revised COMMAND.COM to disk and Quit from DEBUG.

A useful utility for those keen on exploring the environment is ENVINUSE.COM, which displays the size of the environment in handy hexadecimal. It has just been added to DEALS FOR READERS disk AP018.

Policing the environment

The correct name for the environment's memory area is the Master Environment Block. Each item (or, more correctly, each 'string') contained therein is stored in the format VARIABLE = PHRASE, where VARIABLE is the name of one of the environment variables (would you believe) and PHRASE is the information stored with that variable.

Every time you boot up your Apricot, MS-DOS will create certain default environment strings automatically; that applies to variables such as PROMPT, PATH and COMSPEC. The SET command enables you to define your own environment strings.

You are also free to amend the environment to your own tastes — you can create, delete and modify these variables at will. The easiest way to make changes is through SET

The syntax for the command is ...

```
SET [VARIABLE=[PHRASE]]
```

(As usual with MS-DOS syntax explanations, the items in square brackets are optional.)

SET by itself displays the present contents of the environment, as we discovered above.

If you use SET followed by the equals sign but omit the PHRASE parameter, the variable in question will be removed from the environment. Thus SET PATH= will clear the path, while SET PATH will display the path.

A new variable can be created like this:

```
SET A=BBB
```

This will set a variable A to the phrase BBB. Note the absence of any spaces around the equals sign — spaces can be part of the environment too!

The normal command line limit of 127 characters applies to environment strings — though of course 'SET' and its following space take up four characters, so you really have only 123 to play with. PATH and PROMPT, oddly enough, do not require the SET prefix and thus earn a four-character bonus for you.

PROMPTs

Keen readers will already be familiar with the PROMPT command after reading Simon Potter's *Workshop* piece in APRICOT FILE Release 2.7. Customising the prompt is great fun; it's achieved through the use of what are properly called 'metastrings'. An example of a 'metastring' is pg; in conjunction with PROMPT, that will give you a prompt like this ...

```
A:\>
```

All metastrings begin with a dollar sign followed by the metastring character or symbol, as listed in the panel. (Note that all letters are in lower case.)

Metastrings for PROMPT

Metastring	Description
t	the current time [set when MS-DOS was loaded or by the last TIME command]
d	the current date [set when MS-DOS was loaded or by the last DATE command]
p	the name of the current directory of the default drive
q	the equals-sign character
v	the MS-DOS version number
n	the default drive
g	the greater-than arrow character
l	the less-than arrow character
b	the pipe-symbol broken-bar character
s	a leading space only
e	the Escape character (followed by an escape code)
h	an erasing backspace
_	[underline] carriage return/line feed, which returns to the beginning of the beginning of the next line of the screen

You can mix free text and metastrings at will; if you look at my prompt you'll notice the word 'Status' hiding amongst the escape codes.

```
PROMPT=$e($n$g$e)$ej$e;$ep Status :
$e@$e($p$e) - $t$h$h$h$h$h$h - $d - $v
$eK$ek$eq$eO
```

It is possible to have a prompt which consists of just text and nothing else, though altogether more absorbing possibilities are opened up when you dip into the Escape code goody bag — my prompt makes heavy use of the wide range of Escape codes the Apricot is blessed with. To take it in order:

- $e(switches on bright display
- $n the default drive
- $g greater-than arrow
- $e) switches brightness off
- $ej stores current cursor position
- $e; moves cursor to 25th line
- $ep Status : turns on reverse video and displays the word 'Status'
- $e@ enters insert mode
- $e(switches on bright display
- $p gives the current directory
- $e) — switches brightness off and displays a dash
- $t the system time
- hhhhhh six backspaces to overwrite the tenths-of-seconds display
- — $d — the system date, with dashes before and after
- $v the MS-DOS version number
- $eK erases to end of line
- $ek restores current cursor position
- $eq turns off reverse video
- $eO exit from insert mode

Simon Potter's own preferred prompt from the Release 2.7 article illustrates the (quite unusual) use of ANSI escape codes. The Apricot BIOS is

uniquely able to embrace both the VT52 escape codes (inherited from the Sirius) and the IBM PC-style ANSI codes. In fact, the VT52 codes outnumber the ANSI many times over, thus broadening the scope for fun and games; the ANSI codes involve more characters, too, which consume environment space. Simon uses the ANSI version just because he ports the prompt on to any machine he uses, and ANSI is a lot more portable than VT52 ...

As you can see from the above, you can get a bit carried away with the prompt options however you implement them; mine also consumes rather a lot of the environment memory pool.

The PATH command

Many will also be acquainted with this useful feature of MS-DOS. The PATH variable consists of a list of drives and directories that MS-DOS searches to find programs; it thus enables you to run a program or batch file that is not present in your current directory. So you can place related data files in separate subdirectories, yet keep just one copy of your applications program on the disk.

It is restricted to finding only those files with COM, EXE and BAT extensions. So applications with overlay files do not lend themselves to being PATHed, unless you keep a copy of the overlay in the appropriate subdirectory — a rather messy solution. Alternatively, you can use a public domain path enhancement such as DPATH30 (on DEALS FOR READERS disk AP018) which can take care of this shortcoming.

We power-users depend heavily on PATH; it would otherwise be impossible to organise or use a large hard disk efficiently (hands up all of you with a hard disk and *no* subdirectories? Shame on you). In the root directory of a really well organised hard disk system, you should have nothing else but COMMAND.COM, CONFIG.SYS and AUTOEXEC.BAT (plus CONFIG.BAT if you are running the Victor version of COMMAND.COM). And there should be a whole *heap* of subdirectories.

The syntax for PATH is ...

```
[SET] PATH=[D:]DIRECTORY [;[D:]DIRECTORY] ...
```

In short, you can have a preceding SET command (but it's not necessary); directory paths can be prefaced by a drive identifier; and if you want to set up more than one path, separate them by semicolons.

Whenever you run a batch file or enter a command, MS-DOS begins by comparing your entry against its list of internal commands — DIR, COPY and so on. If that search is unsuccessful, MS-DOS then assumes the entry refers to an external command and searches the current directory for COM files, then EXE files, then BAT files. And if *that* proves fruitless, the operating system consults the PATH variable in the environment; it then searches the specified subdirectories in the COM-EXE-BAT order.

On boot-up the path is initialised to a null or empty environment string. If you type 'PATH' at this point you'll see ...

```
A> PATH
   No Path
```

Brief diversion

Sadly, PATH does not offer any solution to the 'Insert Disk with COMMAND.COM' problem — because it can't. COMMAND.COM is loaded into memory in two parts, one low in memory and the other right at the top. This is to permit the temporary erasing of the high bit should the system require the memory it occupies (such as when it loads a program in memory from the top downwards).

The high bit contains the parts of COMMAND that aren't needed while running an application — like the internal commands. More importantly, it also contains that part which calls EXEC in MS-DOS which loads and executes programs.

Once the high bit has been zapped the only way it can be put back is by being reloaded from disk, if it can find it, that is.

You can however force COMMAND.COM to be permanently resident, thusways:

```
A> COMMAND/P
```

All this does is load the high bit of COMMAND.COM low in memory.

Or you can force COMMAND to load from another drive, like this:

```
A> COMMAND B:
```

There's a third method using COMSPEC; more of that later.

But PATH is supposed to help MS-DOS locate COM files; and COMMAND very definitely *is* a COM file. So why can't PATH locate COMMAND.COM?

You can load a second copy of COMMAND.COM by typing 'COMMAND' at the system prompt; and PATH *will* find it. Curiously, when you EXIT from it, COMMAND tries to reload itself from the boot disk — and *not* from the drive and/or directory whence it was loaded. Further, PATH won't oblige you when you exit from an application; the high bit has been erased and COMMAND is nowhere to be found.

If it were an ordinary program. MS-DOS would attempt to load it via the operating system's EXEC function call; and if it can't locate it, MS-DOS will try the path. With COMMAND it's different: MS-DOS will issue an EXEC COMMAND.COM when it realises that part of COMMAND is absent, and the PATH just gets left out in the rush ...

Multiple choice paths

Once a path is set, it's set for the whole system. But there will be times when the path is inappropriate; and changing it manually can be a chore. The batch files described in this section

permit you to change paths at will.

This batch file sets up a variety of paths as variables in the environment. It needs to be run just once.

```
REM SETPATH.BAT
ECHO OFF
CLS
SET PATH1=A:\
SET PATH2=B:\data
SET PATH3=A:\utils
```

You can then alter the paths by running this one:

```
REM CHANGE.BAT
ECHO OFF
CLS
IF /%1==/ GOTO EXIT
IF %1==1 PATH=%PATH1%
IF %1==2 PATH=%PATH2%
IF %1==3 PATH=%PATH3%
:EXIT
CLS
ECHO PATH NOW CHANGED TO %PATH%
```

The syntax is CHANGE *n*, where *n* is the number of the path you want to install.

The percent signs on either side of PATH1, PATH2 and PATH3 identify them to MS-DOS as an environment label in a batch file. When DOS spots the percent signs, it tries to find that variable in the environment. If it's there, the operating system will substitute the phrase it points to for the variable itself and then re-reads that line of the batch file.

Back to the plot

COMSPEC is an environment variable that defines the drive, path, filename and extension of the command line interpreter. To change COMSPEC with SET you would enter ...

```
SET COMSPEC=[D:] [PATH] FILENAME.EXT
```

As described above it is possible to avoid the 'Insert disk with COMMAND.COM' nuisance by telling MS-DOS to use another copy of COMMAND. This SET tells DOS where to find that copy. For instance, on a single-disk F1 or Portable you can set up a small RAMdisk as drive B, copy COMMAND.COM into it, and SET COMSPEC B: so that MS-DOS can find COMMAND.COM again.

Although this does not appear to be necessary on other micros, with Apricots you do have to load the second command processor before COMSPEC will work like this — see David Langford's illuminating exposé of this in APRICOT FILE *passim*. So your AUTOEXEC.BAT file might include these lines at the end:

```
COPY COMMAND.COM C:
C:
SET COMSPEC=C:\COMMAND.COM
COMMAND/P
```

Curiously, COMSPEC seems to duplicate the purpose of the MS-DOS SHELL command. SHELL sits in the CONFIG.SYS file and tells MS-DOS which command processor to use. So you could include the following line in your CONFIG.SYS:

```
SHELL = DEBUG.COM
```

DEBUG would be then loaded right after MS-DOS to become the operating environment for your computer. For reasons best known to Apricot plc, SHELL itself is rejected by MS-DOS as an unrecognised command — pity.

Callable BATs

Another trick uses a link convention that allows one batch file to call another. This is illustrated by CALLER.BAT and CALLED.BAT:

```
REM  CALLER.BAT
IF NOT %BATLABEL% == . GOTO %BATLABEL%
REM  BATLABEL is assumed to be set to '.' as a
     null value
REM  Now do some work, like call and run an
     application
SET BATCALR=%0
REM  This gives the called batch file a name to
     get back to ...
SET BATLABEL=RETURN
REM  ... and that indicates where to come back to
REM  Now set further parameters if necessary to
     save %1 to %9
CALLED
:RETURN
REM  Finish your work here
```

```
REM  CALLED.BAT
REM  Do whatever CALLED was called to do
REM  Then return to caller
%BATCALR%
REM  Separate keywords will be necessary if
     further calls will be made
```

Passwords

Here's another handy trick with the environment — a poor person's password security system. You can stop most uninvited browsers by including something like SET PASSWORD=GANN before you let someone on to the system; then attach these lines to the batch file that loads your sensitive data ...

```
ECHO OFF
CLS
IF %PASSWORD%==GANN GOTO OK
GOTO EXIT
:OK
CLS
ECHO YOU'RE IN!
GOTO END
:EXIT
CLS
ECHO ACCESS DENIED!
ECHO GO AWAY!
:END
```

Unless the user had previously used SET PASSWORD= to kill the PASSWORD variable, this batch file will unceremoniously eject unwanted guests. Don't forget to include a SET PASSWORD= in the batch file *after* the line that loads the application — otherwise the phrase 'GANN' will be left in the environment, and the batch file above will work every time. ◆

IBM on the Apricot?

The inside story of the IBM emulator

By Nick Wilson

At the time when Apricot was designing its first home-built computers, the company was making a lot of money from the Sirius. And very early on, the decision was taken to make the Apricots Sirius-compatible in many ways; this had many advantages, both from the technical viewpoint (the superior screen resolution, for instance) and for marketing — the Apricots could run hundreds of ready-to-go Sirius applications.

On the other hand, it also meant that the Apricots would not even attempt to run IBM programs, because of the differences in the hardware and the software of the two machines.

But it was clear even then from the Statesside vibes that IBM was taking over the market; and it rapidly became apparent that the new Apricot machines would have to have at least *some* degree of IBM-compatibility.

The 'degree' decided upon emerged in the shape of a utility somewhat grandiosely described as an IBM emulator.

Incompatibilities

In order to describe what the IBM emulator is, and what it can and can't do, we must first examine the major differences between the Apricot and the IBM PC.

■■■ BIOS calls

When a program wants to perform an operation such as writing a character to the screen, it can do it in two ways.

The first is to call MS-DOS, which in turn will call the BIOS (Basic Input Output System); but this is slow, as MS-DOS has to check all sorts of things before it can call the BIOS — whether any redirection is currently active, for example.

A faster method is to call the BIOS directly, and that's what most system designers prefer to do. The problem is that the BIOS is provided by the manufacturer rather than by Microsoft; Apricot and IBM machines have very different BIOSes, and their calling conventions are *completely* different.

■■■ Keyboard layout

It does not take an expert to notice immediately that the Apricot keyboard doesn't look like IBM's — different keytops, fewer functions keys (or more, if you include the MicroScreen buttons), a different layout.

At a lower level, however, the differences are much greater. For example, the cursor keys on the Apricot produce an Escape character followed by a letter which denotes the actual direction of cursor movement. The IBM PC, by contrast, sends a NULL character and then a letter which indicates which direction key was pressed; and needless to say, the direction-indicating letter codes are entirely different.

■■■ Screen memory

We've noted two ways of writing to the screen. There is another method, however — it's called Direct Screen Accessing.

If a program calls MS-DOS to print a character, the BIOS eventually writes a word into the area of memory reserved for the screen. But Direct Screen Accessing programs ignore MS-DOS and the BIOS as well, and write directly into the screen display memory.

This is by far the fastest way to write to the screen, and that's why most software packages use it.

The Apricot solution

Those points cover only the major areas of incompatibility. Others include the way communications facilities are provided, disk handling (if it isn't done through MS-DOS), printing, sound generation, and how a program finds out what hardware configuration it's running in — all of which are totally incompatible between the IBM PC and the Apricot.

Apricot released two emulators to help bridge this gap. The earlier utility is the easier to understand; it is supplied in one of two forms, IBM.COM or IBM.OVL. The programs are practically identical, but the OVL version is an overlay file in a format that means it can be loaded by MANAGER.

When it's executed, the first thing the emulator does is set up the IBM calls in memory. A program written to use IBM calls will 'talk' to the BIOS via interrupt 10 hex. The Apricot uses interrupt FC hex (among others). So the emulator sets part of itself to re-route and translate the calls that come in on interrupt 10.

Suppose for example that an application wanted to print a character on the screen. It would do this on an IBM PC by putting the ASCII value of the character in the AL register and function code 14 in the AH register, then issue interrupt 10 Hex. Apricot's IBM emulator will trap this interrupt call, move the character into the AL register, and issue an interrupt 29 — which is Apricot's back-door way to print a character on the screen.

The emulator works like this for all the significant differences mentioned above. The only problems come where the IBM has a function call which Apricot cannot match. In early days one such example was the 'determine system configuration' call; the IBM PC uses this to tell a program how many disk drives it has, how much memory, how many comms and printer ports, and so on. The Apricot had no such call, so defaults were put in as the replies to these questions — and that has caused more than one problem to date!

The keyboard handling is quite simple. The Emulator contains its own internal keyboard table mapped as closely as possible to IBM's keyboard layout, and this is temporarily substituted for the standard Apricot table. On the other hand, there are many key-related calls in the IBM that the Apricot doesn't have — such as a facility to tell if the Shift key is held down. The Emulator makes no attempt to match these.

The screen

Direct Screen Accessing is the most complex part of the Apricot's IBM emulator. Unfortunately, it is unlikely that most users can use it!

The problem is that the IBM has its screen memory area at address B8000 hex, while the Apricot uses the area starting at F0000 hex. On a standard 256KB Apricot, there is literally no memory at all at B8000; so any characters written into memory there will disappear down a black hole never to be seen again.

Things are a bit better if you have a 640K Apricot, for on that configuration there *will* be memory at that address. The IBM emulator scans this area of memory 20 times a second and reads each byte, converts it into Apricot screen format and writes it into the Apricot's screen area at F0000. This whole process slows down the machine by a factor of ten!

Emulator Mk II

The later version of the IBM emulator makes much more effective use of the Apricot, and it also supports most of the calls that were not supported in its predecessor.

For example, the newer versions of the Apricot BIOS (in fact, from 3.1 onwards) support calls to indicate how much memory is available and whether or not the Shift key has been held down.

On this version — called EMIBM.EXE — the useless practice of scanning the memory at B8000 has stopped. And the keyboard table has been taken out from inside the Emulator and put into a separate file on the disk — just as well, because the keyboard table is hopelessly inaccurate and had to be changed a number of times.

Last words

So the Emulator is limited. Less popular software will usually run to some extent. But most IBM packages will *not* run on the Apricot — the well-known best-sellers like Framework all use Direct Screen Accessing, and that immediately counts them out.

On the subject of major packages, you may recall some excitement at the time of the Xen's launch about a utility called Softclone. This was touted as being the new solution for IBM emulation — in particular, Lotus 1-2-3 has definitely been seen running on the Xen under Softclone.

In fact, Softclone is not an IBM emulator as such; instead it patches the desired application with calls to its own driver software. And only a few IBM packages are convertible in this way. So despite all the brouhaha, Softclone is still not yet available for the Xen — or for any other Apricots.

We were told at the time that Apricot was rushed into testing Softclone (a third-party development, from Control C Software) by its US affiliates who suffered most from the lack of IBM-compatibility. They rather jumped the gun; our Apricot source said the boys at Birmingham looked at Softclone and decided that (a) it provided less than 100 per cent compatibility; and (b) it had more than a few Major Bugs. On the other hand, it's also possible that Apricot used Softclone as an excuse for not being IBM-compatible while it was waiting for the Xen-i to be developed.

IBM BIOS calls and EMIBM

We've compiled a list of all the IBM BIOS calls which work with the newer EMIBM emulator, presented here in IBM BIOS call order. In case anyone wants to write a proper IBM emulator, we've put in the corresponding Apricot 'Control Device' calls — on which subject, expect a meaty piece on the Apricot Control Device in an issue or two's time. ◆

IBM BIOS		Description	Apricot Control Device	
Interrupt	Function number		Interrupt	Function number
Diskette Controller:				
13H	0	reset disk controller	39H	00H
13H	1	return disk status	39H	05H
13H	2	read sectors	39H	0BH
13H	3	write sectors	39H	0BH
13H	4	verify sectors	39H	0BH
13H	5	format area	39H	04H
13H	15	return DASD/BPB information	39H	0AH
Keyboard Controller:				
16H	0	read the next character	32H	0BH
16H	1	lookahead to check queue	32H	0FH
16H	2	return the shift status	32H	06H
Parallel Printer I/O:				
17H	0	print desired character	35H	07H
17H	1	initialise printer port	35H	00H
Serial RS-232 I/O:				
14H	0	initialise comms port	34H	04H 05H 06H 07H 08H 09H
14H	1	send a character	34H	01H
14H	2	receive a character	34H	02H
14H	3	return status of comms port	34H	04H 05H 06H 07H 08H 09H
Screen and Video I/O:				
10H	0	set screen mode	31H	01H
10H	1	set cursor type	31H	not available
10H	2	set cursor position	31H	not available
10H	3	read cursor position	31H	not available
10H	4	read light pen position	31H	not available
10H	5	set cursor type	31H	not available
10H	6	set active display page	31H	not available
10H	7	scroll display page down	31H	not available
10H	8	read character	31H	not available
10H	9	write character/attribute	31H	not available
10H	10	write character only	31H	04H
10H	11	set colour palette	31H	not available
10H	12	write dot	31H	not available
10H	13	read dot	31H	not available
10H	14	write Teletype mode	31H	04H
10H	15	current video state	31H	not available
10H	19	write string	31H	keep calling 04H

The Apricot does not have much screen support from within the BIOS; the IBM Emulator manipulates the screen memory, however, and as such can emulate all the functions listed with the exception of the Set Cursor Type primitive.

Memory size determination:			
12H	none	read address 0000:0402H	not available
Equipment determination:			
11H	none	initialise all devices, see which are present	not available

It is possible to obtain information from the Apricot BIOS about what's installed by attempting to initialise the devices and noting the BIOS return code. If the return code is anything other than FAIL (0FFFFH), the device is present.

Cassette Tape I/O:			
15H			not supported

Contributors in this issue

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).

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.

Terence Green is an unemployed Apricot owner (currently looking for a job). Formerly an unlicensed truck driver in Lesotho, he started on IBM 370s and worked his way down to micros; these days he minds the kids and scrapes a living washing dishes (and writing occasionally for APRICOT FILE).

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.He is inordinately proud of being a guest of honour at the 45th World SF Convention at Brighton in August 1987).

David Lewis is a solicitor for a London borough that happens to be a keen user of new technology; he started computing himself six years ago with a ZX-81, but says his *annus mirabilis* was 1986 "when I bought an Apricot Xi and a Miracle Technology WS4000 modem".

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.

Bob Smith has worked his way from big computers that did little to big computers that did a lot to little computers which did little once but now do a lot. This took some 18 years to accomplish, usually with big firms in big towns, but he now tries to do a lot for a little with an Apricot and a minicomputer for a company in a small town in the West Country.

David St. John-Wallis is a sort of psychologist who runs an export marketing consultancy and writes *enormously* thick marketing books about the developing world. He now runs a sort of computer software company which doesn't do very well. He says that all contributions will be gracefully acknowledged.

Nick Wilson wrote his first article when he was 13 and first appeared in print when he was 17. He helped write the first Apricot BIOS, and as such now wears a convincing disguise. Since then he has worked for a Swiss software company writing PC programs; he now writes the software that sits inside petrol pumps.

Diary dates

◆◆◆ June 1987 ◆◆◆

16-18 Networks 87 *London.* Network manufacturers and suppliers on show for the sixth year running, primarily looking for a corporate DP and telecomms audience. Quite an interesting event last year, even though it was geared to large users. *Organiser: Online, tel 01-868 4466*

16-18 Comdex/Europe *Nice.* The European version of the big US Comdex shows for the computer trade is now in its fifth running. Last year it was dire, but the South of France in June was pleasant ... *Organiser: Interface Group, tel (USA) 617 449-6600*

16-18 Scottish Technology Week *Glasgow.* Three parallel shows for design engineering, manufacturing technology, and electronics production. *Organiser: Cahners, tel 01-891 5051*

23-25 KBS 87 *London.* A second running for the 'Knowledge Based and Expert Systems Show' — still the only event devoted to the currently esoteric but definitely emerging world of applied AI ideas. *Organiser: Online, tel 01-868 4466*

29 June — 2 July Electronic Publishing *London.* Conference and show on this slightly spurious but still sexy subject — nearly 6,000 people turned up to last year's event. This one promises everything from desktop publishing to CD-ROM. Day One has an introductory seminar for a mere £225; the mini-exhibition runs from 30 June and is free; the conference programme over the last three days will cost you £660 ... *Organiser: Online, tel 01-868 4466*

30 June — 2 July PC User Show *London.* Yet another of EMAP's shows based on a magazine title. This one is starting to look like one of the three key British exhibitions for anyone interested in small computers (the others being the PCW Show and Which Computer). Despite the IBM-compatible bias, there's always much to interest the Apricot user; fight your way into Olympia with 22,000 other people to admire 200-plus stands. *Organiser: EMAP, tel 01-868 4466*

◆◆◆ August 1987 ◆◆◆

18-19 Scottish Accountants' and Solicitors' Exhibition *Glasgow.* A new one from the organiser of successful (separate) events for these professions South of the Border. Should include sensible systems and services, though other organisers have made the mistake of lumping together the two rather different professions. *Organiser: Truemist, tel 01-749 9535*

19-21 Comdex/Australia *Sydney.* Second running of the Aussie version of the show geared to computer-industry resellers and marketeers. *Organiser: Interface Group, tel (USA) 617 449-6600*

◆◆◆ September 1987 ◆◆◆

23-27 Personal Computer World Show *London.* Not as essential as it has been in the past, but the lure of 65,000-plus visitors means there will be a lot of exhibitors — should be around 400, though many of them will still be selling to the home-computer users around which PCW was originally based. *Organiser: Montbuild, tel 01-486 1951*

Advanced SuperWriter

Part Two: Automatic Boilerplating

By Bob Smith

Well, folks, what did you think of the SoapBox analysis in Release 2.8? For me, it was a dream come true to rate top, middle *and* bottom all in the same list! Half a mo', though, we *are* talking about statistics, aren't we? Follows a closer examination — ah! And here's another ah!

First ah! is about that December report. Actually, there *was* no *B&P* that month: goodbye middle score. *(That was inadvertent over-terseness on our part; the article in question was actually Neil Mathers'* Bandage & Plaster Revisited *piece on generalised backup routines — Ed.)*

And the second ah!, dearest Editor, concerns the lowly rating of 2.5 for the February *B&P*. As pointed out in the accompanying discussion, there were only two responses in the February issue. Well, one of them was mine (and I've got a copy to prove it). As I didn't score my own article — should have made the return anonymous — it must mean that some wonderful person out there gave it *five*! Right? *(Right — Ed.)* Prove it, whoever you are, and you're on for a Devon cream tea or a stiff double!

Back to earth

Meanwhile, to get down to the less serious stuff, this *Report* is the second on squeezing the utmost out of SuperWriter. It describes the most complex utility that I have created so far with this software.

It stems from a need here at Bandage & Plaster to produce certain regular documents quickly, yet flexibly. First, a little terminology.

I love the word 'boilerplating'. Hopefully, I'm long past the age when speaking jargon seemed more important than speaking sense; but every now and again up pops a word or phrase that just cannot be resisted. I don't mean I go around continually dropping remarks into conversations like "just been doing some heavy boilerplating today" or "of course, if you don't use boilerplating, what do you expect?". No, it's simply a word that seems to fit what it's used for.

Having said that, I'd better spell out what I mean by the term. In the broadest sense, it's the art of non-creative writing — the manipulation of words in such a way that a document can be created by building it up in chunks from a number of smaller, pre-written pieces of text, then polished up with whatever original text and formatting is needed to produce the final result.

Actually, it's usually easier to write the original text first with markers to indicate where the standard stuff goes; then you can bring it in, check through for a final edit or format, and print. This article describes a method to automate that procedure from completion of the original text through to output.

We are talking here of very boring and repetitive documents that have to be produced regularly — legal documents, sale descriptions, contracts, catalogues, and the like. In fact, if it's anything that's routine and follows a recognisable pattern, always look for a way to get the machine to do the drudgery.

Economic writing

With that motive in mind, it may be more fair to slot boilerplating into the broader category of techniques that can be called 'economic writing'. Let's have a look at some of the tricks every computer user can employ to get the most out for the least put in — particularly, what goodies SuperWriter itself provides to assist the devotee of economic writing.

It is said that a computer programmer plying his/her trade in the business systems world needs only ever to learn three styles of program-writing — the update program, the report program and the data entry program. Having successfully produced one each of these, every program that follows is in some way a copy of what went before. Such an appraisal of programming is a bit mean, but it's the reason why every programmer worth his/her salt builds up a library of proven code from full programs down to tiny routines that can be used over and over again. Programmers who make a living directly from the programs that they write especially use such methods, for it makes them both efficient and cost-effective.

Form letters

In other words, good programmers follow the principle of economic writing. So can ordinary users of word processing. Keeping the programming example in mind, consider what types and formats of text you produce regularly, be it letters, memos, reports, notes, or whatever. In the case of letters, let me illustrate with my own routine.

I keep an electronic copy of every letter I write. If I write to someone for a second time, I invariably hunt out the first letter, copy it to a new name (usually something really original like 'AF2', say, for the second letter to APRICOT FILE). I then delete everything from this copy except formatting controls (if they exist), the address, date, salutation and sign-off or ending. That's if the new text is unrelated to the last letter; if there's anything in the original that may be of use, I leave that in too.

If it's a new addressee, I either simply use someone else's letter as the base and replace the non-matching bits with what's relevant; or I employ a file called PAGE1. This is a form letter: it has the form of one of my letters but there is no actual text other than words that mark the places for the address, date, salutation, etc.

This is important for me, because the letterheads that I use have a preprinted company address and logo that cover a good two and a half inches of the top righthand side of the sheet. As I always put the date on the right, it has to be positioned so that it will always print below the logo. Further, the letter should look balanced when printed: and so the address and salutation need to be carefully positioned as well.

So it's important for economic writing to have a form letter (or an existing letter based on it). It allows you to forget all that fiddly positioning and get on with the writing. You could put all your print formatting into the form letter as well — margin settings (left, right, top, bottom), escape codes for preferred printer settings, and so on.

My own preference is to use the simple trick of starting up SuperWriter from a batch file that copies a letter-writing definition file (called LETTER.DEF) to SW.DEF. When SuperWriter is loaded, it will then set its defaults according to the instructions in the new SW.DEF. Printer settings are either preset by the print function menu (PFM) described in this column in APRICOT FILE Release 2.4 or directly through SuperWriter by the equivalent PFMLIB method described in Release 2.8.

If you take a little time and thought to set up similar outline files for other regular formats of text that you do, perhaps with appropriate batch files similar to the one mentioned, you've started to do with text what professional programmers do with code: you're on the trail of economic writing!

Doing it with SuperWriter

The three basic economic writing facilities of SuperWriter are ...

- 'find-and-replace' — ESC R
- 'include file' — ESC I
- 'include section' — a library utilities option

The first two techniques should be common to all decent word processing software, but the latter is special to SuperWriter (more on that later).

By the way, what I prefer to call 'find-and-replace' is simply called the 'replace' command in SuperWriter parlance. In terms of economic writing, it is most effective when used to expand abbreviations to their full extent. For this, you need to compile a list of frequently-used long words or short phrases (maximum 25 characters including spaces) and work out unique abbreviations for them. Then you have to train yourself in using them — which may seem a strange piece of advice, but if you try abbreviating on the fly, while you're actually writing, you'll probably find it disrupts your train of thought too much. Once again, thorough preparation is the chief starting point of this and every other form of economic writing.

The 'include file' facility is better if you prefer to create the full text during composition; and it's essential if your standard pieces of text are greater than 25 characters. This facility is easy to use, but suffers from two disadvantages.

First, though you could create *one* file to hold all your standard text, SuperWriter will not let you capture less than a screen's worth at a time — thus forcing you into unwanted editing just to get rid of the extraneous text. Further, every time you use the facility, you will have to skip one at a time through all the screens in front of the one you want.

Second, you may decide to avoid these inconveniences by creating a separate file for each standard piece of text. But you then find yourself using up a great deal more disk space and cluttering up the directories with a lot of small files.

Keyed section files

In the manual, the 'include section' facility is actually described as boilerplating; but on its own, it doesn't quite make the grade. It requires the addition of automation to make it really useful. And how do we get automation? Through XQTs, of course!

Keyed section files were described in *Advanced SuperWriter — Part One* in Release 2.8. But to summarise, their main characteristic is organisation into chunks of text — 'sections' — each preceded by a 'key'. The keys are a word or phrase preceded by a formfeed character (appears as a dim exclamation mark) and followed by a Return. If

the first word of each key is made unique (thus permitting exact matches), the remainder can be used for a descriptive comment. Everything after the Return up to the next formfeed is the 'section' of standard text.

That way, all the text for boilerplating can be kept together in a single file; and yet it can be retrieved directly by specifying the key. The sections can be as small as a single word (unlikely in practice), or as large as you wish to make them up to the limit imposed by SuperWriter's text memory capacity (about 32K).

A practical example

Legal documents, particularly wills, are superb candidates for boilerplating; I know of at least one legal supplies company that provides just such a service (on disk) as part of its offerings.

My example concerns a lease document. Look at Figure 1 to see the finished result and at Figure 2 to see what was actually typed in; the only operator action required between the two was starting up an XQT and a couple of editing keystrokes before the final document was saved and printed (as part of the XQT).

How was it done? First, the standard text has to be prepared in advance in a keyed section file as described above. The BPLIB file shown in Figure 3 was used for this example. Best to organise it in some sort of logical sequence, such as alphabetical key order, and to keep keys short but meaningful — remember comments are allowed (as with RIS, COND1, and elsewhere in Figure 3).

Next you write the document, placing section keys wherever appropriate, and identifying them by placing formfeeds directly in front and after the key name within your text.

Finish the document with all original text and key names, then invoke the boilerplating XQT called BP. You *could* boilerplate in stages, but it's usually faster to do it all in one go. Even so, the XQT will give you a chance to edit the document after pulling in the standard text and before saving and printing.

Finally, go to the top of the document before running BP.XQT. That could be done in BP.XQT; but since the action would be repeated for every section pulled in, it would also add a further (and avoidable) delay.

The XQTs

Six XQT files are involved altogether; five of them you write, BPKEY.XQT is generated by the others. Before running BP for the very first time, however,you should create a dummy BPKEY.XQT simply to provide a directory entry. Again, this preparatory action saves running time in the main process.

The six XQT files

BP	finds the keys and directs the process up to completion
BPGEN	generates BPKEY
BP1	first part of BPKEY
BP2	final part of BPKEY
BPKEY	BP1 + key name + BP2
BPOUT	run by BP when all section text retrieved.

The XQTs themselves are all listed and described in detail at the end of this article. But as they involve a number of tricks (sorry, techniques) not previously illustrated in APRICOT FILE, it's worthwhile having a look at exactly how they accomplish the boilerplating.

BP — an intelligent XQT?

The sequence of execution is as follows:

BP ▶ BPGEN ▶ BPKEY ▶ BP ... ▶ BP ▶ BPOUT

BP finds the first key name within your document by looking for a formfeed. If it finds one, BPGEN is executed. If it doesn't, it 'knows' that all the section text has been retrieved and pauses to let you do any final editing before continuing with an ampersand and running BPOUT — which saves the document, prints it and returns to the Main Menu (from where you can start editing another document or quit).

BP demonstrates the use of a decision-making or conditional XQT. This is definitely a trick, not a technique, as SuperWriter macros don't really have a proper condition-testing mechanism. Considering SuperWriter is really one of the earlier (and now seemingly quite primitive) word processors, it's somewhat unfair to castigate it for this omission; but such a property is very useful and highly desirable in any batch language.

Never mind. If you can wangle a 'Find' into your XQT, you can simulate conditional operation by using the response needed for found/not found to direct the outcome: if the Find is successful the next command is actioned. In the case of BP, it's an escape sequence (^[) leading on to the execution of the XQT called BPGEN. If the Find is unsuccessful, a message appears at the bottom of your screen saying ...

```
Press any key to continue — Cannot find xxx
```

(xxx is a formfeed in BP's case). Thus the escape ^[is interpreted as the 'any key to continue' and the

following text 'xBPGEN' simply as *text* — meaning it gets inserted into the document. The next seven ^H codes move the cursor to the front of these unwanted characters and the ^N deletes them.

The mixture of ^E and ^Ds cause a series of short beeps — ^E is an illegal command, thus causing SuperWriter to squeak a protest: ^D is a short delay (see the *Bandage & Plaster Report* in Release 2.5 for a detailed explanation). That's simply to tell you the boilerplating is done, and that you can now do any final editing or tidying up before you enter an ampersand to run the final XQT — BPOUT. Thus the found/not found outcome can direct your XQT down separate paths (via other XQTs); or it could stop or continue with the current XQT.

BPGEN — an XQT that writes XQTs

After finding a formfeed, BP runs BPGEN. The main business of this XQT is to sandwich the key name in the edit document with the necessary instructions to retrieve the related text section from BPLIB.

It does this by using the formfeeds either side of the key name in the edit document to position the cursor. So the first thing is to place a block marker at the front formfeed, move the cursor right (on top of the formfeed), then *include* BP1.XQT as 'text' in front of the key name. Notice the 'include' — it conveniently retrieves XQT instructions as easily as normal text.

The included BP1 pushes the front formfeed down the document, so we search for that again and this time delete it. We next find the back formfeed and put in a Return — needed to 'enter' the name in the 'Include section' menu. Then include BP2.XQT, insert a second block marker, and write the block as BPKEY.XQT out to disk. The block is deleted, the back formfeed found again and the newly generated BPKEY executed.

The back formfeed is finally deleted by the latter part of BPKEY, and BP is re-executed. So it goes on till no formfeed is found by BP; as described above, the whole process is then paused for final editing before BPOUT completes it.

Last words

Personally, I think this bunch of macros shows just how good SuperWriter software really is. On the other hand, it must be used carefully — prepare well, get into a routine, that sort of thing. This way of working may seem boring to you; but if it makes the boring bits quicker, or allows someone else to use it — someone perhaps less competent, less experienced or cheaper — it has its merits.

If you are doing boilerplating, or if you want to try it, please write and tell me about it; I'm particularly interested in other uses you may have, as well as experiences of the techniques described here. Further improvements, comments or gripes are most welcome. ◆

The techniques summarised

BP library	Leave a blank line at the end of each section in your boilerplate library so that there are always *two Returns* separating the end of the section text and the following key. The boilerplating XQTs rely on this format!
Printing	Once set up, you can print your keyed section file using the trio of XQTs introduced last month — CMD, FFDEL and PTEMP.
Key names	Make sure you place formfeeds *directly* before and after the key name within your text. Don't leave any spaces between the name and the formfeeds.
Formfeeds	Insert formfeeds by using the MicroScreen INS MARK key followed by 'F'; or use CONTROL-Q, then '2F'.
No key	If the key name from your document doesn't match with one in your boilerplate library, little harm is done. The main effect is to close up the words either side. If preceded or followed by a Return, these will also be deleted.
Final edit	The boilerplating stops by *not* finding a formfeed. This results in an effect like 'no key' above: the text may be closed up, or a Return deleted. If it happens, it will be after the last section included and at the point where the cursor is located when BP is paused for the final, manual, edit.
Timing	A complete BP > BPGEN > BPKEY cycle takes about 14 seconds on a hard-disk Apricot; the lease document example took just over 2½ minutes to boilerplate. Add the printer time and an allowance for final editing and you could calculate a work rate, if that's useful to you.

Figure 1: The document as printed

THIS LEASE made the first day of April One thousand nine hundred and eighty seven BETWEEN Robert Iain Smith of Castle Barum, Barnstaple (hereinafter called 'Lord Bob') of the one part and F. Harry Mulligan (hereinafter called 'the Lessee' which expression shall where the context so admits include the persons deriving title under the Lessee) of the other part

WITNESSETH as follows :-

1. In consideration of the rent hereinafter reserved and of the covenants on the part of the Lessee hereinafter contained, Lord Bob HEREBY DEMISES unto the Lessee ALL THAT plot of land, buildings and premises known as the Hen House at the bottom of my garden, together with the rights to be enjoyed in common with Lord Bob and all other persons similarly entitled, for the term of five years from the first day of April One thousand nine hundred and eighty seven.

YIELDING AND PAYING to Lord Bob

the yearly rent of Six Thousand Pounds payable without any deduction by equal monthly payments in advance the first payment to be made on the date hereof

2. The Lessee hereby covenants with Lord Bob in manner following that is to say :-

(1) To pay the rent hereby reserved and at the times and in the manner aforesaid

(2) Not to exhibit on the outer walls or roofs of the Demised Premises any placard, sign, advertisement, fascia, skysign or lettering, whether electric, illuminated or hanging, except where approved in writing by Lord Bob

(3) Not to use the Deemed Premises or any part thereof for any illegal or immoral purpose or for the carrying on of any noxious unusually dangerous or offensive trade or business, except where approved in person and anonymously by Lord Bob

SIGNED SEALED AND DELIVERED by the said F. Harry Mulligan

Figure 2: The document as entered

```
THIS LEASE made the first day of April One thousand nine hundred and eighty seven BETWEEN !RIS! of the one
part and F. Harry Mulligan !LESSEE! of the other part
!WIT!
1.      !GIVE! the Hen House at the bottom of my garden, !RIGHTS! for the term of five years from the first
        day of April One thousand nine hundred and eighty seven.
YIELDING AND PAYING to Lord Bob
        !RENT1! Six Thousand Pounds !RENT2!
2.      The Lessee hereby covenants with Lord Bob in manner following that is to say :-
        (1)  !COND1!
        (2)  !COND6!
        (3)  !COND15!
!SIGN!
F. Harry Mulligan
```

Figure 3: The boilerplating sections — BPLIB

```
!RIS    Name of lessor~ Robert Iain Smith of Castle Barum, Barnstaple (hereinafter called 'Lord Bob')~
~
!LESSEE~
(hereinafter called 'the Lessee' which expression shall where the context so
admits include the persons deriving title under the Lessee)~ ~
!WIT~
WITNESSETH as follows :-~ ~
!GIVE~
In consideration of the rent hereinafter reserved and of the covenants on the part of the Lessee
hereinafter contained, Lord Bob HEREBY DEMISES unto the Lessee ALL THAT plot of land, buildings and
premises known as~ ~
!RIGHTS~
together with the rights to be enjoyed in common with Lord Bob and all other persons similarly
entitled,~ ~
!RENT1~
the yearly rent of~ ~
!RENT2~
payable without any deduction by equal monthly payments in advance the first payment to be made on the
date hereof~ ~
!COND1  Lessee conditional clause 1~ To pay the rent hereby reserved and at the times and in the manner
aforesaid~ ~
!COND6  Lessee conditional clause 6~ Not to exhibit on the outer walls or roofs of the Demised Premises
any placard, sign, advertisement, fascia, skysign or lettering, whether electric, illuminated or
hanging, except where approved in writing by Lord Bob~ ~
!COND15  Lessee conditional clause 15~ Not to use the Deemed Premises or any part thereof for any
illegal or immoral purpose or for the carrying on of any noxious unusually dangerous or offensive trade
or business, except where approved in person and anonymously by Lord Bob~
~
!SIGN~
SIGNED SEALED AND DELIVERED by the said~ ~
```

BP.XQT — starts the boilerplating

^Q3f	[find formfeed]
^[xBPGEN~	[execute BPGEN]
^H^H^H^H^H^H^H	[cursor seven chars left]
^N	[delete remainder of line]
^E^D^D^E^D^D^E	[three short beeps]
&	[pause the action]
^[xBPOUT~	[execute BPOUT]

BPGEN.XQT — generate BPKEY

^[/	[insert a block marker at cursor position]
^L	[cursor one char right]
^[iBP1.XQT~	[include BP1 into document]
A	[all of it]
^Q3f	[find that front formfeed]
^G	[delete it]
^Q3f	[find back formfeed]
~	[insert CR (carriage return)]
^[iBP2.XQT~	[include BP2 behind key name]
A	[all of it]
^Q3f	[find back formfeed again]
^[/	[insert second block marker]
^[cwBPKEY.XQT~	[write out BPKEY to disk]
Y	[confirm write over existing file]
^dby	[delete block just written out]
^H	[cursor one char left to ensure success of next find — needed because after deleting block, cursor returns to start of line on which first block marker was placed]
^Q3f	[find back formfeed again]
^[xBPKEY	[execute BPKEY]

BP1.XQT — front part of BPKEY

^[~	[go to 'Main Menu']
^K~	[up one — select 'UTILITIES']
~	[select 'LIBRARY' option]
^J~	[down one — select 'Include document']
~	[select 'Specify Include file']
BPLIB	[name of file]
^K~	[up one — select 'Include Keyed section']

BP2.XQT — back part of BPKEY

A	[accept all of section text]
^[	[return to 'Library file access']
^[	[return to 'Utility selections']
^[	[return to 'Main Menu']
^[	[return to document under edit]
^Q3f	[find back formfeed]
^H^H	[cursor two chars left]
^G^G^G	[delete two CRs at end of section and that back formfeed]
^[xBP	[execute BP]

BPOUT.XQT — save and print final document

^[s~	[save the document]
^J^J~	[down two — select 'PRINT document']
~	[name defaulted to edit file]
~	[select 'Format and PRINT document']
^[	[return to 'Main Menu']
^Z	[stop XQT dead]

The Easy Way

If any reader would like a copy of the techniques and sample files described in this two-part mini-series, please send a formatted diskette with a self-addressed, stamped (20p) disk mailer to the address below. If you don't have a mailer, just send the stamps (26p for first class) and your address — but please enclose your disk in some form of protective padding, at least wrapped in a piece of cardboard.

R I Smith
13 South View
Chaddiford Lane
Barnstaple
North Devon
EX31 1RD

Review

On-line Attendant

A toolbox for your Apricot: Ansible's ACOLYTE version 1.4

By Terence Green

ACOLYTE is basically a memory-resident utility from Apricot afiçionados Ansible Information. The disk is also packed with useful extras — plus a couple of pleasant space-filling time-wasters.

What you get

A complete list of the files provided is given in the panel; they come on an Apricot 3.5in single-sided disk which is rudimentarily copy-protected.

(The intention isn't so much to stop registered users making backups, as to discourage unscrupulous multiple copying. There isn't much anyone can do to stop determined hackers; but there aren't very many of the small Apricot software suppliers left, and users would do well to support the few remaining. And after all, if it's good enough to copy for friends, it must be worth paying for at £50 or thereabouts.)

The volume label on this disk is neatly underlined when you do a DIR, but if you send that directory list to your Epson it will turn on 1/8in and then 7/72in line spacing and leave your printer there. I habitually DIR > PRN for documentation; and ESC codes that provide a screen effect but act differently when sent to the printer are a nuisance. Perhaps it's just part of the protection, or maybe a visual treat. Either way I can do without it. (As it happens, the printer intercept program supplied with ACOLYTE could be useful here.)

Read on

The ACOLYTE manual is clear and straightforward (which isn't always the case with professional writers, as Ansible's directors are); the information is presented in a logical fashion, and you're advised to "work through a stage at a time". Within minutes, following easy-to-understand instructions, I had programmed a key to fill my screen with the subtle message 'Ansible Information'. Gosh, these sci-fi chaps are clever!

A READ.ME file on disk supplements the manual and is commendably short on last-minute bug fixes. Much of it concerns files added to the latest version of ACOLYTE (which is 1.4) — a Good Thing, since printed manuals can be tediously out of date when a product is at such an early stage of its life.

The main reason

The principal offering on the disk is ACOLYTE itself, in the form of a utility called AC.COM. This provides a prompt on the 25th line and a range of main options called from MS-DOS; it's invoked by hitting CONTROL and the ENTER key on the numeric keypad.

You can install AC.COM in your AUTOEXEC.BAT file so that it loads automatically on boot-up; that way you have it available at any time. Or you can simply install or remove AC directly from the A> prompt as and when you need it. A screen message informs you when ACOLYTE is installed and reminds you of the main options available, which are ...

- time and date on the status line
- monitor rescue
- loading and saving ACOLYTE settings.

The clock can be toggled on and off in 12- or 24-hour formats, with or without seconds; the date appears next to the clock, with the option of having the weekday in 'Mon' style. And these all know what to do at midnight — I stayed up late so I could confirm the changeover for you (well, actually the Editor just laughed when I asked about the overtime rate, so I had to cheat).

The 'monitor rescue function' is a very useful facility indeed. If invoked, it switches off the screen image without any effect elsewhere if there is no keypress for around 10 minutes. This will stop the image of those high-intensity reverse-video screens 'burning' permanently into the screen's phosphor coating when you have to answer the door or the phone and end up gone for half an hour. Sophisticated techniques were used to check that operation continued after ten minutes when the monitor went blank — a WordStar search/replace operation! This monitor-blanking facility doesn't operate on the F series LCD screens, by the way.

The last main option is a load-and-save choice for up to 10 different configurations of your chosen key settings with a default option. These saved settings can either be entered into AUTOEXEC or other batch files; or, again, they can be loaded from the A> prompt as required.

Personalisation

At almost any time — even while you're using other applications — a further selection of utilities can be accessed from the ACOLYTE prompt on the status line, by pressing CONTROL and the numeric keypad ENTER key.

The numeric keys are used with SHIFT and CONTROL to give 26 configurable keys each with a maximum of 79 keystrokes. You too can simulate a BBC *Micro Live* presenter using keyboard macros,

now that *Micro Live* has come to the end of its life (pity the BBC micro hasn't).

Certain applications programs with smaller keyboard buffers may override the 79 character limit — trial and error is required. The key configuration can contain print controls, returns and tabs as well as characters, though all these facilities are temporarily suspended while the built-in calculator is active; they return as soon as the calculator is switched off.

The key configuration can be toggled off temporarily if you use an application that preempts the 25th line, or if you use graphics-based shells such as GEM and Apricot ACTIVITY. In the latter case you are into something else entirely and probably won't understand why some people are determined to cut down their key-pressing and mouse-chasing. F Series owners will also need to toggle the clock off when they use ACTIVITY or the on-screen calculator.

You can still communicate with ACOLYTE and use the programmed keys while in graphics screens. The snag is that you can't see the status line; so you're deprived of confirmation of your keystrokes as you enter them, which some people find unnerving.

Any key on the keyboard can be 'twinned' simply with any of the keys you have already programmed. This provides a quick and easy way of setting up function keys, for instance, from within your application.

Printing

Another status line selection allows entry of ASCII characters from the 'high-order' range that is not available directly from the keyboard — very useful with the printer intercept utilities.

For in addition to AC.COM you get the Ansible Printer Program, APP.COM, designed to intercept your output to printer. There are three versions that utilise intercept techniques in increasingly complex ways — Ansible suggests that by using only the version containing the facilities you require, you can avoid the increased memory requirement of the later versions.

The APP utilities will intercept output to the printer and convert any character you nominate to another one, or to a string of characters. This is useful for accessing the full ASCII character set on your printer and when you want to use occasional control character strings, particularly when you'd like the convenience of entering a single character in your document rather than the whole string each time. The ASCII character entry facility mentioned earlier comes into its own here.

But the printer intercept program needs careful thought — and an understanding of your printers's control codes. If your printer has a hex dump mode and you haven't sorted out software controls, this could be worth getting into. If you want quick answers and have an Epson-compatible printer, try using the public domain program EPSON.COM (on DEALS FOR READERS disk AP002) in conjunction with your printer manual.

Files on the disk

THE MAIN FILES	
AC.COM	ACOLYTE main program
ACF.COM	ACOLYTE for F Series
APR.COM	ACOLYTE independent printer port toggle
APP.COM	printer intercept program V.1
APP2.COM	printer intercept program V.2
APP3.COM	printer intercept program V.3
OTHER UTILITIES	
DONTINIT.COM	suppresses printer initialisation signal
INIT.COM	restores printer initialisation signal
CP.COM	CONTROL-P toggle detect
CAPS.COM	Caps Lock toggle
CLICK.COM	keyclick volume control
HOLD.COM	adjusts duration of beep
TONE.COM	frequency control for beep
LOUD.COM	volume control for beep
REP.COM	toggle auto-repeat on/off
TIM.COM	clock only function for PC/Xi/Xen
TIMF.COM	clock only function for F Series
AICLEANF.CHR	clearer font for F Series
INTFIX.COM	fix for early BIOS versions
HARE.COM	speeds screen output on monochrome monitors
TORT.COM	slows screen output
EXTRAS	
AAM.COM	anagram generator
FOG.COM	Fog Index readability assessment
SEEK.COM	locate text strings in files
MODEL.COM	detects Apricot model
READ.ME	disk documentation

You can also select these facilities;

- an on/off toggle for the clock on the status line
- an option to enter the date in full 'dd month year' format at the cursor position (for letters and so on)
- switching between serial and parallel ports (you can cancel a long printout by squirting the output to the serial port and back to the parallel port, or vice versa)

Finally there's a bundle of extras — some useful, some fun, some downright irritating for the average user.

The latter group includes TORT, which slows down screen handling and, for Xen users, visual

input: it's provided because some programs for the 8086 Apricots zip by too quickly when driven by the Xen's 80286.

And AAM produces anagrams for crossword buffs — not only what you might call the 'legal' ones, but most possible combinations of letters in a given word. We're told that ACOLYTE now includes an optional text file that will screen out a thousand 'illegal' letter combinations (like 'AAA'), but this will still mean picking up the 'legal' anagrams from some dross.

For inclusion in batch files there are three potentially handy utilities:

- MODEL which checks which Apricot the program is running on and can be set up to perform conditional choices
- CAPS which toggles upper case on and off and
- APR which toggles between output ports

And there's more!

- HARE speeds up the monochrome monitor (an alternative to the DEALS FOR READERS utility FAST or OmrC's QUIK)
- CP gives an on-screen status report if the screen output is toggled on by CONTROL-P (but doesn't help while you are using a word processor with CONTROL-P print controls)
- REP toggles on and off the auto-repeat for key depression
- DONTINIT suppresses the printer initialising when certain applications load (so you don't lose your preset formats)

And a blessing to dogs, dog-lovers, and those like me too inept to program the 'bell' — LOUD, HOLD, and TONE to customise its volume, period, and pitch. Put them in your AUTOEXEC along with CLICK (key click volume), if Simon Potter has left you any room.

Fools and tools

My dad used to say "never buy a tool until you need it" when he wasn't chasing me round the house with a hammer in his hand. Meaning that if you buy a complete socket set because it is cheap and comes in a bright red tin box, there will come a time when you need to use a socket and your nice shiny red box won't have one that fits. It will have metric sockets instead of imperial, or the largest socket will be too small, or ... well, you know what I mean.

We all have programs that do everything we want them to do except the last thing we thought of. From that point on they cease being useful and started to become frustrating.

With this collection, Ansible Information has provided a bundle of tools which will prove useful for many of those boring and repetitive tasks we all have to do. These tools will also, on occasion, provide you with some timely assistance. The product seems to have been designed to answer everyday needs.

A few caveats though; memory resident programs may cause problems with some of your applications. If you use GW-BASIC the monitor will blank for all time. Well, until you reboot. The same holds for programs running with the BIOS 3.1 Extended Graphics loaded, but not for MSBASIC with GSX (which returns, as expected, with a keypress). On such matters, there is some additional guidance offered in an appendix to the manual to supplement that sprinkled through its main body.

Oops; nearly forgot. FOG.COM grades your writing style for readability on the basis of an established test for counting the number of words and syllables in the sentences of a document. Ansible Information cleverly doesn't run it against the manual: but I, for one, had to put in a lot of Big Words to get the grade I thought I deserved. ◆

ACOLYTE

Description:	Memory resident utility
Supplier:	Ansible Information
Address:	94 London Road Reading Berks RG1 5AU
Telephone:	0672 62576
Price:	£57.50, inc VAT and despatch (available from DEALS FOR READERS)
Configuration:	Any (non IBM-compatible) Apricot
Conclusion:	A collection of useful utilities that can be invoked and adjusted from within your applications. A few quibbles, but nothing that outweighs the gain in convenience — keyboard macro function especially useful.

Although David Langford of Ansible Information writes a regular column for this newsletter — for which he's paid (usually) — we don't have any particular brief to recommend Ansible products. We treat the products of all contributors just like anything else the Apricot user is being invited to buy. So we were prepared to be critical of ACOLYTE; it's pleasant to be able to carry such a favourable report, and we have added this product to our DEALS FOR READERS mail-order lists as ref no AS035.

Free Small Ads

The full list of Classified Ads is featured in the DEALS FOR READERS catalogue published every other month (ads remain in the full list for around three months, or until they start yellowing at the edges, or until you ask us to delete them). This page summarises ads new this month, plus useful-sounding services (including our own!) and most non-sale entries.

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 and very quick! Call Roger Gann at TP Group on 01-833 3501.

■■■ TURNKEY CONSULTANCY: we can recommend and/or implement systems using popular packages or custom-written software. We can also convert you from Sirius or Apricot to another species. Sensible and knowledgeable advice, no particular dealership ties or other axes to grind, modest prices, guaranteed work. 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 turnround, guaranteed quality, low prices. You're reading an example of our work right now ... Call Roger Gann at TP Group on 01-833 3501.

Apricot Sales

■■■ APRICOT PC with twin double-density drives, 12in monitor, 768KB memory, Intel 8087 maths co-processor, Microsoft Mouse; with Mannesmann Tally MT180 printer (three spare ribbons) and sheet feeder, voltage regulator, head cleaner kit, dustcovers — plus all the standard software (SuperCalc, SuperWriter, SuperPlanner, CP/M-86, Concurrent CP/M) and DR-Draw, GSX, WordStar 3.3, SWAP (Mercator), dBASE II, PolyPrint, Open Access, various APRICOT FILE disks, a host of other useful utilities; plus manuals for several languages. All with original documentation and packaging, bought from UK in June 1985 and used sparingly. Offers to G S Jayaram, PO Box 2642, Doha, Qatar (phone 417282 6-9pm UK time) — free air-freight despatch to any UK destination.

■■■ APRICOT PC with twin double-density drives, 9in or 12in monitor, and 512KB memory, the usual 'Super' software, COMM, CARDBOX, and utilities. Call Peter any evening on 0483 277169.

Apricot Add-on Sales

■■■ 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.

■■■ MICRO FOCUS LEVEL II COBOL (high-performance version) and FORMS-2 utility for sale. Develop applications using the world's most tried and tested language! Call Steve Budd, 0903 65287 (work) or 0903 68400 (home).

■■■ APRICOT PC 9in MONITOR. Brand new and unused (I bought a 12in screen instead): £100. Also APRICOT PC MOUSE, virtually unused: £75. Phone Robin Leaver on 09662 4104.

■■■ APRICOT PC SOFTWARE — OPEN ACCESS integrated system (database, spreadsheet, 3-D graphics, word processing, comms, appointments diary): virtually unused, a real bargain at £250. Sapphire's REPORT MANAGER (3-D spreadsheet, over 60 functions); £100. THE ULTIMATE DIARY (up to 5,000 entries): £15. MICROFILE (superbly simple electronic card index for up to 32,000 records): £20. NUTCRACKER SUITE (simplified text editing with typewriter-like functions): £15. Contact A Leigh on 01-542 4872 evenings.

Other Sales

■■■ COMMODORE MPP1361 printer; 136 columns, 150cps. As new (hardly used). Offers around £350. Phone Robin Leaver on 09662 4104.

Wants

■■■ APRICOT PC with twin double-sided drives, manuals and monitor (would consider an Xi). Prompt payment. Phone 0525 376339 any time.

■■■ 10/20MB EXTERNAL HARD DISK required for Apricot PC with BIOS 2.4. G S Jayaram, PO Box 2642, Doha, Qatar (phone 417282 6-9pm UK time).

■■■ APRICOT 9in MONITOR (cream) wanted for PC. CARRYING CASE also needed. Phone 01-908 4631.

Help

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

Misc

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

■■■ COMMODORE CLEAR-OUT — FREEBIES! A few items left over from a previous venture — copies of Dennis Jarrett's Complete Commodore 64 book and many cassette dust-covers. FREE to a good home — like a school? Contact Roger Gann on 01-833 3501.

Holidays

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

Use the service!

The Classified Ads service is free for all subscribers to APRICOT FILE newsletter.

Just let us have a sheet of paper with the words — not too many, please, and we do retain the right to edit ads and to refuse contentious material.

Send your ad to us via electronic mail (Telecom Gold 81:JET767) or by land mail to:

APRICOT FILE
TP Group
PO Box 509
London N1 1YL

AWD update

David Langford of **Ansible Software**, purveyor of the **Ansible WordPerfect Utilities** disk we reviewed favourably in Release 2.8, dropped us a note with an update ("after anguished demands from millions of picky users", the WP macro feature has been amended so that hard copies of macros can be printed) and one correction — the MicroScreen keys really *are* operative, even though they aren't labelled. Seems the problem is that the labels ought to be reset when leaving WordPerfect; but then you could *temporarily* exit the word processor with the Shell command to find on your return that the MicroScreen labels had been zapped.

APE that Apricot!

Incidentally, Ansible Software has a Beta-test version of an **Apricot emulator for IBM-compatibles**. As yet it's restricted in how much Apricot screen control can be reproduced on the IBM, but it sounds like A Good Idea — especially if you'd like to move SuperWriter on to a PC-lookalike without pausing for breath. A full working copy of the program (called APE.COM, of course) should be ready shortly; meanwhile Ansible has a few copies available at **£30** for would-be guineapigs — drop a line to **94 London Road, Reading, RG1 5AU.**

Cheap Racal by Tandy

Tandy has slashed the price of its intelligent V21/V23 modem; at £229 it wasn't much of a bargain when compares with the likes of the Miracle WS4000, but at **£74.95** it looks a great buy. That's particularly the case when you look under the Tandy livery to discover a rebadged **Racal 2123** — which Racal is selling for around **£350.**

Yellow peril

The **EEC** is polling two dozen Japanese-based printer manufacturers with a view to assessing claims that printers are being dumped at uneconomic prices in the Community. This has long been a shibboleth of European printer and typewriter makers; but with the current economic mood starting to favour protectionism — at the end of April EEC foreign ministers announced new powers to curb Japanese imports — this time the EEC may well be inclined to impose **anti-dumping levies**. For other goods, the EEC is politely threatening 100 per cent import surcharges unless Japan responds by relaxing its own curbs; but that hardly applies to computer printers, since Japan is the predominant source. The Japanese say the European printer manufacturers are simply less efficient, which is why they have less than 20 per cent of the total market here. Anyhow, **Epson** and the up-and-coming **Citizen** will both be manufacturing in Europe by this time next year and should escape any penalties.

One To One for sale

The electronic mail service **One To One** has been put up for sale by its US parent company **Pacific Telesis International.**

Officially, Pacific Telesis has "strategic reasons" for the sale (specifically to prepare itself for further deregulation in US telecomms next year). But one report talks of "sustained losses", though these are said to be within budget forecasts — "it can take years to make a return ... we have been building up a user base".

With between 15,000 and 19,000 users (depends whose figures you believe) and a turnover claimed to be something like £4 million a year, One To One is not a small company — indeed, it's probably second to Telecom Gold (albeit a long way second) in the British e-mail market. But Pacific Telesis is huge; it's a long way up the *Fortune* 500, and by turnover it will probably be in the top ten US companies this year. The potential of the alternative ventures for Pacific Telesis can be counted in the billions; One To One is relatively small beer for the Americans, particularly in a general-purpose e-mail market now increasingly dominated in this country by Telecom Gold.

One To One is worth keeping: it has pushed hard for interconnection of e-mail services, and it pioneered the sensible 'check with no charge' policy which gives users 30 seconds to check their mailbox without being charged.

We're told that there are "upwards of five" bids under consideration, two of the options being management buy-outs. Some staff have been given their cards already; Pacific Telesis probably wants to present a leaner company as a more saleable package. But more services are planned, broader gateways are under discussion, and price restructuring is rumoured. The rumours apply to Telecom Gold too, as it happens; they suggest a conversion to Telex-style charging by character and/or by usage rather than by time on the system. This type of pricing is common for commercial e-mail purveyors in the States.

More about this in the May issue of our sister publication COMMS FILE.

Many micros

According to the **National Computer Centre**, there were **584** different single-user micros on sale in Britain as of November 1986 ...

Apricot 40-days bug-fix

There are **problems** for SuperWriter users with NEW_BIOS2.7, the 'official' Apricot fix for the 40-days bug (see APRICOT FILE Release 2.8). **Neil Mathers** dropped us a note starting "I am able to type this letter on SuperWriter because I am *not* using the fix supplied by Apricot"; at the very least, it seems the patch screws up the MicroScreen keys pretty comprehensively, putting fraction characters where the likes of INS MODE and INS MARK should be — "and I don't know what else", adds Neil.

It seems that the NEW_BIOS issue disk features the American version of KEYTAB.SYS, the keyboard table file. Quite how the boffins at Apricot Advanced Tech achieved this masterstroke defies belief, but there it is.

The American keyboard is, of course, subtly different from the UK one and certain characters and editing keys will be absent — so if you want a pound sign, tough luck. The version we're selling as DEALS FOR READERS disk AP022 has been patched by us to overcome this problem and features a pukka UK keyboard. Anyone experiencing keyboard problems on NEW_BIOS2.7 is welcome to send us the disk to us with a stamped addressed mailer and we shall do the rest.