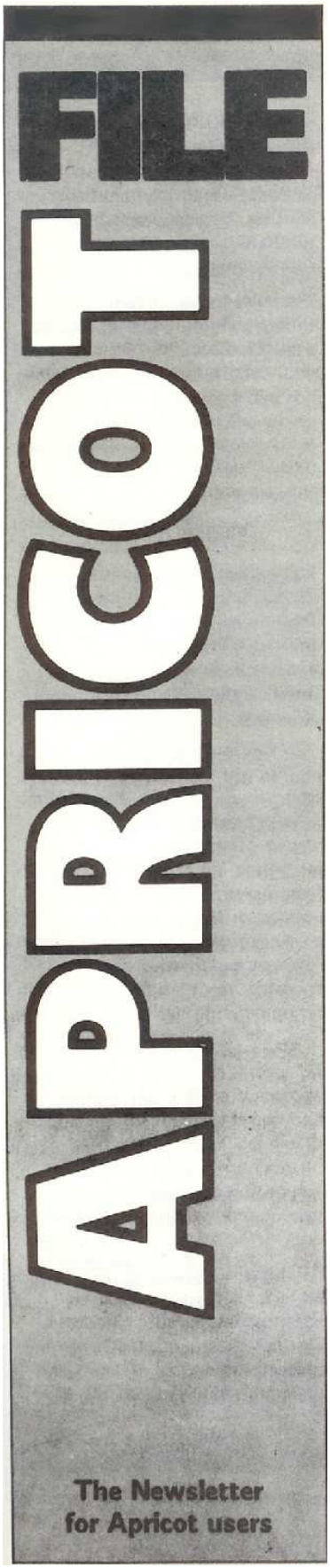
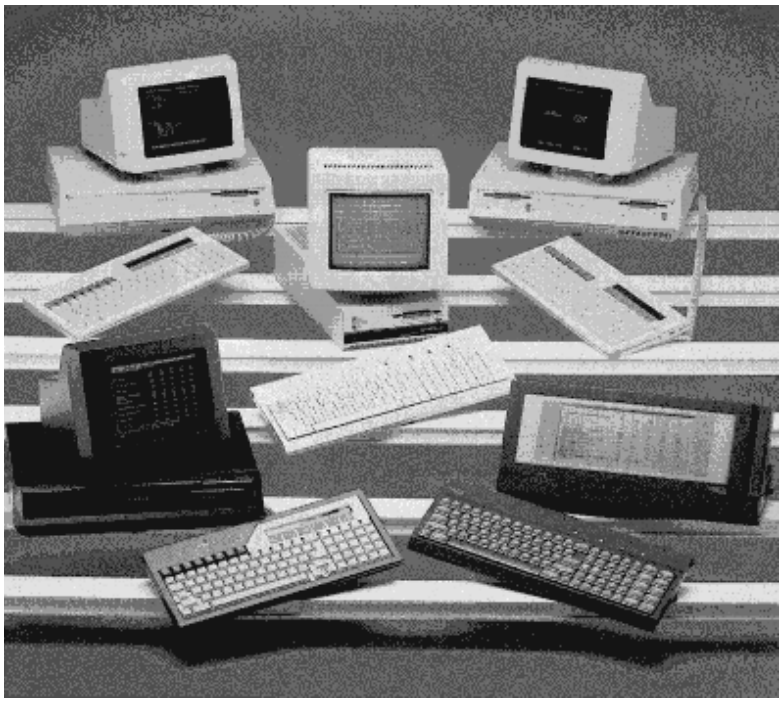


Apricot File 3.12



Scans courtesy of www.actapricot.org

Copyrighted content republished with thanks and permission from:

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

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

Release 3.12

APRICOT FILE

The Newsletter for Apricot users

Final Issue

The Last Word

By Dennis Jarrett

It is with a considerable sense of sadness that we have to break the news that you are reading the very last issue of APRICOT FILE.

Why? Well, there are a variety of reasons. The most obvious, and probably the most pressing, is commercial: the newsletter is barely breaking even, and every month loses more subscribers to the beckoning lure of IBM-compatibility.

A second consideration is editorial — what is there left for us to write about? That inevitable drift away from the Apricot world is matched by the maturity of the remaining subscribers, and for some months now it has become harder and harder to find articles that aren't too esoteric but which don't talk down to the readership. Obviously you can't please everyone all of the time; but we know we have already lost some subscribers because they felt the newsletter was getting too technical — and we have also received some criticism about the lighter articles we've carried.

In short — we feel that APRICOT FILE is becoming less and less *useful* to its subscribers. We don't want to con anyone: if we can't do it right, if we can't produce a newsletter that meets our own standards, we'd rather not do it at all.

And frankly, I'm getting a bit tired. TP Group is a four-person company, three directors and a bookkeeper, and one of the directors is taking an 18-month sabbatical right now. Basically the functions split between Roger's DEALS FOR READERS service and technical queries while I look after the desktop publishing and newsletters. I've been doing APRICOT FILE more or less single-handedly for three years now, and I'm afraid the last few issues have demonstrated that my publishing batteries need a recharge.

So it's time for the obituaries. We haven't got rich from APRICOT FILE (that was one of the original intentions back in 1985) but we have enjoyed the newsletter — we've learned a lot about Apricots and about desktop publishing, we've met some interesting people and faced some interesting questions. On balance we're quite proud about the world's only independent publication specifically for the Apricot.

And we certainly don't want to lose touch with the Apricot world. We intend to continue the DEALS FOR READERS service and to develop it, with better and more frequent mail-order catalogues offering wider and more useful support services. We'll be taking stands at a couple of computer exhibitions during the year, too, so please look out for us.

The last word: without your support we wouldn't have been able to produce APRICOT FILE for the past three years. We would like to offer our thanks to all contributors, past and present: to all those people who manfully ploughed through *Q&A* queries for us: and above all to you, the subscribers. ◆

In this issue

Disinformation: On oddments and ANSI 2
CAL at work — Apricots and interactive video 7
Readers' Writes 10
Free Small Ads 11
Review: Future imprefect? WordPerfect 5 12
Diary dates 13
The Workshop: A Turbo Pascal Primer — Part Nine 14
Review: GABBS — the Apricot Bulletin Board 17
News Analysis: Apricot goes MCA 22
News and views 24

On oddments and ANSI

By David Langford

Does FORMAT.EXE work with PC/Xi BIOS 2.7? I have met conflicting reports — and a weird statistical anomaly whereby older, more experienced Ansible customers were generally *less* able to make FORMAT work. Sometimes, even more bafflingly, it would work for a bit and then start issuing rude "Wrong BIOS version" messages. Odd.

The set-up

The culprit was Apricot's own SETUP disk configuration utility, the old PC/Xi version with ladder menus, as supplied to early customers and used only by the brave. Although initially confusing, this seemed to work well ... but secretly it was saving one wrong byte. In fact, SETUP configuration changes the 'major' BIOS version number to 1 before writing its bumf to the disk label sector. (What does this unnoticed error suggest about the BIOS version prevalent when SETUP was written?) BIOS 2.7 is thus heavily disguised as 1.7, and so on.

At boot-up time the bad information is copied to the BIOS configuration table in RAM, which is read by FORMAT. This table starts at a double-word address stored in the usual Microsoft offset-then-segment format at 700 hex in the zero segment (address 0000:0700 in DEBUG display format): offset at 700H, segment at 702H.

Using this information you can read the actual configuration table, which starts with ... well, Peter Rodwell's *Advanced User's Guide* is strangely silent on the table's early bytes, I suspect because of the cock-up. You *should* be able to read the minor and major BIOS version numbers directly from the first and second bytes of the table — for example, 7 and 2 respectively for version 2.7, 1 and 3 for BIOS 3.1. To make life more difficult, the late BIOS 3.1.2 is reported as though it were 3.2. After SETUP, these version numbers come out as 1.7, 1.1 and 1.2. Perhaps Rodwell thought it better not to document such a seemingly unreliable feature.

Apricot have provided a cure with the disk label-sector editor LABEL.COM (on DEALS FOR READERS disk AP027), which elsewhere I've also seen renamed as DSKLABED.COM to distinguish it from the LABEL.EXE volume label utility supplied with Apricot MS-DOS 3.20. The third edit screen of LABEL allows you to modify the bytes 'Version No. LO' and 'Version No. HI', correcting SETUP's error and also allowing the potentially dangerous experiment of fooling BIOS-specific programs about the version under which they're running. Be warned! Despite what some people think and a few will loudly tell you, altering this version number information does not (of course) alter the BIOS version which actually boots up.

Since macho power-users avoid such almost-friendly utilities as LABEL, I've been curing afflicted floppies more swiftly with DEBUG. With the disk in drive A, you can load the label sector with L 100 0 0 1. The command D 180 shows the hexadecimal contents of the disk's configuration table, starting with the two bytes already mentioned and going on with a diagnostic byte, then a byte which is zero if the parallel printer is selected and 1 for the serial one, then a byte for beeper volume (0 full, 15 off), and on into information about MS-DOS, keyboard, screen, comms, etc. If your BIOS version is 2.7 and the D display begins with 07 01, you can fix it with E 181 2 and write the corrected sector to disk: W 100 0 0 1.

Assuming the BIOS number is now reliable, your own programs might usefully read it to learn what facilities are available. Getting it from disk would be folly: a floppy could have been changed since boot-up. Instead, imitate FORMAT and read the RAM configuration table. In Turbo Pascal the information can be had as an Integer (or Word in version 4.0) via this:

```
MemW[MemW[0:$702]:MemW[0:$700]]
```

Or in assembler:

```
XOR AX,AX
MOV ES,AX
MOV BX,ES:[700h]
MOV ES,ES:[702h]
MOV AX,ES:[BX]
```

In both cases the major and minor numbers will appear as the high and low bytes of the integer result (Turbo) or AX register (assembler). When constructing assembler code in DEBUG, the H-for-hexadecimal is omitted from numbers, and the ES: segment command is deported to a line of its own before each of the last three MOV commands — for instance, ES: and then MOV BX,[700].

And in Basic? I wish you wouldn't ask such tactless things, but just this once ...

```
10  DEF SEG = 0
20  BOFS% = PEEK(&H700) + 256*PEEK(&H701)
30  BSEG% = PEEK(&H702) + 256*PEEK(&H703)
40  DEF SEG = BSEG%
50  MIN% = PEEK(BOFS%)
60  MAJ% = PEEK(BOFS%+1)
```

The last line shows how to read any byte of the table — just slot in the appropriate internal offset instead of the '1'. Similar offsets can be inserted in assembler; the last line there would become MOV AL,ES:[BX+3] to get the printer flag byte already mentioned. The equivalent Turbo expression is

```
Mem[MemW[0:$702]:MemW[0:$700]+3]
```

... using *Mem* rather than *MemW* to extract a single byte.

A dictionary utility

Speaking of disks, the F2 and F10 floppy drives are widely rumoured to be untrustworthy. For a while I noticed no trouble, but eventually found that the F10 sometimes formatted disks less than perfectly.

A knowledgeable pal claimed that early Apricot FORMATs would skip the odd sector — and that to be on the safe side you should format every floppy two or three times. Doing this, and switching to the latest FORMAT, gave a slight improvement but not 100 per cent reliability.

I suspected that the stepping mechanism for the drive head had got old and creaky, for the problem seemed connected with head position. On rare good days the drive wrote to double-sided disks and verified ok; on average days only single-sided disks would work; on bad days the words 'Disk Error' and 'Data Error' were endlessly repeated like mantras. When this happened I used the empirically-devised Bad Day Disk whose low-numbered data sectors were filled with about 50K of junk file: once this much had been written to a blank disk, the rest would usually copy safely.

Then Steve Gimblett of Western Computer Maintenance (0752 228564, plug plug) suggested the Dictionary Utility. The F10 is now handling floppies perfectly, with a copy of Webster's *Seventh New Collegiate Dictionary* stuffed between the main box and the monitor. With the monitor standing directly on top as usual, magnetic leakage from the CRT can interfere sporadically with disk operation — depending, presumably, on screen brightness and tilt, the head's physical location, and even the bit pattern being written to the floppy. Do try a dictionary before giving your drive heads the wire brush and Dettol treatment.

The WEBSTER.COM configuration utility also cured an F2 which would boot from drive B but not from drive A. Guess which of the vertically stacked drives is closer to the monitor?

The smartarse corner

In the last episode of Mark Whitehorn's thrill-packed Turbo Pascal primer (APRICOT FILE Release 3.11), I couldn't help noticing some oddities — doubtless deliberately inserted as a test for budding programmers. These are connected with the code which writes quantities of goodies into random positions of a game-location array.

The program GAMEDEMO.PAS defines an enumerated *Contents* type whose values include *Empty*, *Water*, *Gold* and *Penguin*, and the variable *Location: Array[1..40,1..30]* of *Contents*. (As you'll have guessed, although it wasn't stated explicitly, this array definition is Pascal shorthand for *Array[1..40] of Array[1..30] of Contents*.)

So far, so good.

After setting every element of *Location* to Empty, and then filling 20 at random with *Gold*, 20 with *Water* and 10 with *Penguins* (in that order), the program claims in comments that the array will now contain 20 or fewer water jugs, 20 or fewer sacks of gold, and *exactly* 10 penguins. Obviously there's a chance that some random Gold assignations will overwrite previous ones for Water, and that Penguins can overwrite either. The missed-out point is that there's a small but significant chance that with true random selection, Gold will also overwrite Gold and Penguins will overwrite Penguins. Statistically, the latter should happen in approximately 3.7 per cent of program runs. In other words, pedants should note that there'll be exactly ten penguins only 96.3 per cent of the time ...

Also, here's a useful tip. GAMEDEMO.PAS sets all the elements of *Location* to Empty by use of nested *For* loops. No harm in that; but should program speed ever become vital, with the Location array being cleared frequently, Turbo offers a much faster way:

```
FillChar(Location,SizeOf(Location),Empty);
```

Note that this statement works no matter how you change the dimensions of *Location*!

Those were minor points. The major one, showing the commonest pitfall of array usage, appears when values are assigned to array elements by such statements as:

```
For Count:= 1 to 10 do
Location[Random(40),Random(30)]:= Penguin;
```

Seems ok? Check the array definition and look again. The function *Random(N)* returns an integer value in the range *zero to N-1*. Not only will this statement never affect any array element whose first index is 40 or whose second is 30, it will occasionally try to change elements with a zero index — outside the boundaries of the array. In the absence of array bounds checking, this could write Penguins (whatever numerical value they might actually have) into the data preceding the array proper ... possibly even into the machine code. This is immensely dangerous, and a Bad Thing.

When developing a Turbo program with heavy array use, it's worth asking for index range checking with the compiler directive *{$R+}*. Just include it near the top of the program. Bad array calls, like an attempt to slot a *Penguin* into *Location[0,0]*, will stop the program with a 'Run-time error 90' message and a note of the PC (program counter) value. It's easy then to locate the offending statement — either immediately by pressing ESC if you're running the program in memory from within Turbo, or otherwise by asking for 'Find run-time error' on Turbo's Options menu and then entering the given PC value.

When all's well — and remember, some array indexing errors may show up only rarely — remove the *{$R+}* again to restore full (if reckless) program speed. Meanwhile, when trying last issue's GAMEDEMO.PAS, shift those random array positions into the correct range with, for example:

```
Location[1+Random(40),1+Random(30)]:= Penguin;
```

Further escape plans

Bright ideas for *Disinformation* columns have always been welcome, unless they implied hard work. Eddie Bromhead suggested that I supplement last year's *Complete Escape Sequences* series with a piece on the ANSI (American National Standards Institute) screen controls, usually mentioned here in tones of dislike.

Let's take a look at the subject, explaining in the process why this 'IBM-compatible' screen handling fails to impress me with its efficiency. First I'll expand on the original series's background, and mention display methods where ESC-sequence controls *don't* apply.

As detailed last year (APRICOT FILE Releases 2.9 to 2.11), 'screen' escape sequences are strings of two or more characters which always begin with ESC, and which when 'displayed' don't appear but are intercepted by the Apricot screen driver and interpreted as special BIOS commands. These usually affect the screen (erasing or deleting all or part of it, moving the cursor, setting attributes like underline or inverse video, etc) but can in principle do anything which came into the clever little heads of the BIOS authors (like keyboard modification).

All this is tackled by what Apricot call the Interrupt F1H handler, through which all MS-DOS screen output is passed (this includes 'file' writing and copying to the CON display device). The famously undocumented MS-DOS Interrupt 29H sends stuff to the display by the same route. Note that there are other routes! The Control Device screen handler (Int FCH, Device 31H) doesn't intercept ESC sequences and treat them as special: the ESC just appears as the left-pointing arrow which is its nominal font character. If an Apricot program using ESC sequences runs at all on a bog-standard IBM, there's usually a rash of these arrows, introducing ESC sequences which the IBM tactlessly displays without acting on.

ESC sequences are likewise ignored by Apricot's final BIOS 3.1 screen handler, working through Int ECH. An article on this is perhaps due: Apricot recommended it strongly for all 'generic' applications, but although it's fast the user has to do a frightful lot of work. 'Displaying a character' through Int F1H involves no more than sticking the character code in the AL register and calling the interrupt: the BIOS automatically puts the character at the current cursor position, making it bright, underlined or inverse-video according to the current default, and goes on to reposition the cursor and scroll the screen as necessary. Writing a character via Int ECH requires that you load CL with command code 4 ('display character'), CH with a screen ID, DX with the character *plus* attribute codes, AX with display position coordinates, SI with optional font information and DI with an 'image ID' laboriously obtained from earlier calls ... and after all that, if you want the cursor moved or the screen scrolled up you must do it yourself with more Int ECH calls. Help!

Finally ... displaying text by the disreputable method of writing straight to video memory will naturally also bypass the ESC sequence handler, and ensure incompatibility not only with IBMs but also with other Apricots. The F series and Portable keep their video memory at one address; the PC, Xi and Xen use another location and a different format. In the latter case the video memory at segment F000H is the same as that for the Victor/Sirius: but Sirius programs which write straight to video memory produce gibberish on the Xi and Xen, because the Sirius character coding isn't compatible with Apricot's system of font pointers. Rule of

computing: it's *always* more complicated than you thought.

Into the ANSI world

Once understood, Apricot's own 'VT52' ESC sequences are straightforward. ESC Y is a good example. When this sequence is received, the BIOS 'knows' that the next two characters passed to the screen driver will be coded row and column coordinates for cursor repositioning. The characters are intercepted and the cursor moved as appropriate.

My fundamental objections to the committee-designed ANSI screen controls are that they're invariably longer — wasting time by requiring more characters to be sent to the display — while their weird back-to-front syntax requires more processor overhead. Thus ...

All ANSI sequences start with ESC [(ie ASCII characters 27 and 91 decimal, 1B and 5B hex). This tends to be followed by a number or numbers, in the long-winded decimal ASCII format whereby the value 64 is sent as two characters, 6 and 4, rather than being compressed Apricot-fashion into the single character @ whose ASCII code is 64. (Already, processor time is being wasted converting raw numbers into numerical strings.) The rule is that the first *alphabetic* character in the sequence constitutes the 'code' which finally tells the BIOS what the command has been all about. For example, the cursor positioning code is H (or f). To move the cursor to the tenth row and fortieth column of the screen, you send the eight-character string ...

```
ESC [ 1 0 ; 4 0 H
```

Or, if you prefer:

```
ESC [ 1 0 ; 4 0 f
```

(The screen's first row and column are taken throughout as being number 1, not zero. Semicolons are used to separate multiple numerical ANSI parameters.)

The format accounts for my nasty remarks about processor overhead. Obviously, eight characters take twice as long to send to the screen driver as the four in Apricot's normal ESC Yyx. Perhaps less obviously, the fact that the command code comes at the end means that the screen driver has had to spend time testing each of the six characters after the ESC [lead-in, to see whether it's a command letter (with the character being stored for future reference only if it isn't).

To be fair, the ANSI sequences needn't always be quite this verbose. Numerical parameters can be omitted and, if so, a default value of 1 will be assumed. All the following sequences and their ESC [..f equivalents will home the cursor.

```
ESC [ 1 ; 1 H
ESC [ 1 ; H
ESC [ ; 1 H
ESC [ ; H
ESC [ H
```

... the last being only mildly less efficient than Apricot's coincidentally similar ESC H.

In what follows, I'll use capital N to stand for a numerical ASCII parameter as described above. A value of (say) 10 for N means sending two ASCII characters, 1 and 0, as in the cursor positioning example. (0 and 1 are characters 48 and 49 in the ASCII code table, remember!) Where there are two Ns, for cursor positioning, we'll call them X and Y for column and row. Don't forget that omitting any N, X or Y will have the same effect as setting it to 1, and that the semicolon separator can be omitted when there's only one parameter or none at all. The actual capital letters N, X and Y are not part of any valid Apricot ANSI sequence, and neither are the spaces I've inserted for clarity.

Group 1: Cursor Control

cursor up N rows	ESC [N A
cursor down N rows	ESC [N B
cursor right N columns	ESC [N C
cursor left N columns	ESC [N D
cursor moved to (X,Y)	ESC [Y ; X H
cursor moved to (X,Y)	ESC [Y ; X f
save cursor position	ESC [s
restore saved position	ESC [u
cursor position report	ESC [6 n

Notes:

- Single-step cursor movements with ESC [1 A or ESC [A (and so on) bear an uncanny resemblance to the Apricot ESC A, ESC B etc. cursor commands. The ANSI codes will similarly not move the cursor beyond the edge of the current screen window: rather than wrapping around, it won't move at all.
- Why two identical 'move to X,Y' commands? Apricot's BIOS 3.1 manual lists the second only as 'IBM-compatible', though both are supported on the IBMs to hand here. Older Apricot manuals claim a different syntax for the second sequence, with the row and column parameters swapped: ESC [X ; Y f. This is apparently just a mistake. As far back as BIOS 2.4, the commands work with identical syntax.
- The ESC [s and ESC [u cursor save and restore are interchangeable with the more usual Apricot

ESC j and ESC k: a position saved with ESC [s can be restored with ESC k, and so on.

- The 'cursor position report' function is documented as 'device status report' and is a real weirdie. What it should do, according to IBM ANSI documentation (required to make sense of the BIOS 3.1 manual's incomprehensible reference to it), is to return the cursor position to the keyboard buffer in the format of the dummy ANSI sequence ESC [Y ; X R. What it *actually* does is to imitate Apricot's own position-report sequence ESC n, returning the position *not* in ANSI format but as an Apricot ESC Yyx sequence which can be read as if from the keyboard (see APRICOT FILE Release 2.10). No IBM-compatibility there.

Group 2: Erase screen text

erase to end of screen	ESC [0 J
erase from start of screen	ESC [1 J
clear whole screen	ESC [2 J
erase to end of line	ESC [0 K
erase from start of line	ESC [1 K
clear whole current line	ESC [2 K

Note:

- Erasures 'to' a given point begin at and include the cursor position. Erasures 'from' a point end at and include the cursor position. None of these commands extends beyond the current screen window or moves the cursor. (Some Apricot-standard sequences do; ESC E clears the screen and homes the cursor. Oddly enough, ESC [2J *does* home the cursor on IBMs. So much for the gesture towards compatibility!)

Group 3: Screen text attributes

turn off all attributes	ESC [0 m
turn on bold text	ESC [1 m
turn on underlining	ESC [4 m
turn on inverse video	ESC [7 m

Note:

- The attributes can be freely combined (not so on an IBM, where monochrome inverse-video text can't be also bright or underlined). But ANSI's lack of specific 'off' commands makes for clumsiness. If all three attributes are set and you want to turn off underlining alone, you must send ...

  ESC [0m ESC [1m ESC [7m

  (as opposed to the Apricot's ESC 1).

Group 4: Miscellaneous

set auto CR after LF	ESC [20 h
turn off auto CR after LF	ESC [20 l
set wrap at line end	ESC [7 h
turn off auto wrapping	ESC [7 l
set screen window (lines N1 to N2)	ESC [N1 ; N2 r
set origin mode	ESC [6 h
turn off origin mode	ESC [6 l

Notes:

- The 'wrap' and 'auto CR after LF' features behave just like Apricot's standard ones. The 'auto CR' feature works with all three of the 'move cursor down one line' screen control characters, LF (0A hex), VT (0B) and FF (0C). In each case, sending the character to the screen when this feature is on will also cause the cursor to be moved to the left of the current window.
- The 'screen window' sequence isn't as good as Apricot's: you can set the window's top and bottom lines only. The cursor is homed in the selected screen area. If an Apricot ESC , sequence has already been used to specify a window, ESC [..r will preserve its left and right margins while resetting the top and bottom. N1 and N2 are always measured from the physical top of the screen, not that of the current window, and the first screen line is counted as line 1 as above — except that on the F10 here, it's treated as line 0 for the purposes of this ANSI command only. Setting/clearing normal windows with ESC , or ESC . will also wipe out the effects of ANSI windowing. As with normal Apricot windowing, a window height of only one line, or two, can't be set on the F2/F10.
- 'Origin mode' doesn't feature in my IBM ANSI references, and Apricot has explained it only in obsolete Technical Reference Manuals — the super final BIOS 3.1 manual merely mentions these commands. The obvious implication is that they don't actually work. With origin mode on, the ANSI cursor moves supposedly behave like ESC Y by refusing to move the cursor out of the current screen window. With origin mode off, you can supposedly move outside the window with ESC [..H or ESC [..f. In fact it makes no difference; apparently you can always leave the ANSI window with these sequences though not with ESC Y, while none of them will let you leave a non-ANSI-compatible window — ie one set with ESC , which doesn't fill the whole screen width.

So much for the fabled ANSI sequences. On the whole I wouldn't bother. ◆

CAL at work

Apricots and interactive video

By Dr Stuart Bunt and Dr Mark Whitehorn

With the rapid rise of new technologies and the emergence of the 'enterprise culture', training and re-training have become hot topics. Any method which makes such training more painless and efficient is to be encouraged; and as in so many aspects of our lives, Information Technology has made an impact in the field of education.

In the early days of CAL, some very bad programs were rushed out to catch the boom as many computers were introduced into schools. These often consisted of little more than a 'book on the screen' or a testing automaton. All these programs succeeded in doing was confirm in the minds of many teachers that the computer had no place in the classroom.

It is now being realised that computers can form a valuable addition to the teachers armoury — when used properly. Nowhere is this more evident than in the field of interactive learning; here the flexibility and fast response times of the computer have revolutionised the ways we can teach.

Interaction

Instead of the information being presented in a linear fashion, as in a lecture or a text book read from cover to cover, interactivity means that students can follow their own course of study — they 'interact' with the course material to alter the way in which the information is presented. Non-linear interactive learning involves rapid feedback from the student and the modification of their progress based on testing; computer-based training programs are ideal for this role.

Sometimes the student influences the order or speed of the learning process, for example by deciding which topic to study and at which particular level. In other cases the tutor can use the student's input to alter their progress; for example, they can be tested on their understanding and/or advised to re-examine the topic if the test reveals a deficiency. Individual students learn in the manner most appropriate to their particular skills and knowledge.

Such teaching programs can benefit from the structure of text-based computer games. For many, these have a fascination which is created by employing intangible 'rewards' (such as the satisfaction of gaining access to a particular room in a fictitious castle). In education, we believe, it is foolish to ignore such an easily-generated enthusiasm for problem-solving. Interactive teaching programs have this potential because the student has to become closely involved in the learning process, constantly being asked to make decisions and receiving feedback and rewards for success.

Video

Most interactive programs can benefit from the addition of real video images, as opposed to those generated by computer graphics. For example, during a tutorial on cell division the computer might ask 'Would you like to examine mitosis or meiosis?'. On answering 'meiosis' the student would be shown to a time-lapse sequence of meiotic division. The computer would take care of all the details such as where the images were stored and how they were retrieved.

Simulations are particularly impressive when combined with interactive video. In one program used to teach cardiac care, a patient on screen responds realistically to treatment. It can dramatically effect the student's retention of facts when they actually watch an incorrect choice of drug 'kill' the patient on screen before their eyes!

Graphic material in CAL has traditionally been presented as pictures in a book alongside the computer program. Many school systems still work in this way; for example a geography lesson on the computer might involve a student piloting a plane around the globe, but the detailed maps will be provided in an accompanying work pack. (Computer-controlled slide carousels are good for demonstrations but have little advantage over good conventional illustrations in a book.)

Computer-controlled video cassette players have been used with some success but their slow mechanical response times have limited their usefulness. Further, only the most expensive professional video tape units produce good-quality still images, and the quality of the stored image deteriorates with usage. By the very nature of interactive work, the images cannot be removed from the tape in a predictable sequence; so it can take several minutes to find images, which may be at different ends of the tape.

It has required the advent of the computer-controlled laser disc, with its fast random-access times, to make interactive video a reality. Such players mean that interactive CAL programs can include high-quality still or moving images, slow-motion, reverse, and so on. The worst possible time to find an image is about six seconds, and the discs will outlast mere mortals — although recent worrying reports of disc corruption by dyes used in the disc labelling throw some doubt on this.

Philips started the revolution by inventing the large format laser disc for storing up to half an

hour of full colour video on one side of a 12in disc. Due to poor marketing and initial high pricing, these 'LaserVision' players had only a brief popularity; and what little impact they did have was soon displaced by VCRs which allowed recording as well as playback.

The Japanese have seen how this market is taking off, however, and are trying to muscle in. Their main challenge is smaller format discs on which large amounts of data can be stored by using data compression techniques. This is necessary because one full-colour TV-standard image can take more than a megabyte of storage, and these are produced at more than 25 frames a second for real-time video!

Several disc sizes have been proposed, from 3cm to audio-CD size and up. The compact disc size seems to be winning in the field of text storage, (telephone books on a disk and so on), but the video standard has yet to emerge. At time of writing the Philips LaserVision standard still has more discs available than any other format.

An interactive video system requires a computer and its associated VDU, a colour monitor for the pictures, and a laser disc player. Once these are assembled it is very tempting to take the system one step further and overlay the video pictures with test and graphics from the computer. The opportunity would then exist to outline features of interest on the picture, to label structures or to pinpoint them and require the student to identify them.

Several cards are available to slot into IBM-compatibles to allow video overlay but they are still relatively expensive at £800 and up. Cheaper Genloc boards (which synchronise the video signals of the computer with that generated by the disc player) are available for the BBC Model B: but then you are restricted to the colourful but low resolution graphics of the BBC.

Cost

The computer may be the cheapest part of the equation. Any IBM clone from Taiwan will do for a start — in interactive video the response time of the player and student are usually longer than any delay caused by the computer.

The cheaper clones are incapable of generating complex graphical images to overlay on top of video images (or incapable of generating them with sufficient speed) but could produce text.

The price of computer-controlled laser disc players remains high because the market is so small. It is only in the last few years that the LaserVision players have found a niche in education. The players required for interactive video are exotic beasts capable of still frames, multiple audio tracks and computer control which pushes up the price still further. They are all capable of using CAV (constant angular velocity) discs, where concentric circles on the disc contain one frame each. This is in contrast to the cheaper CLV or constant linear velocity players used to show films; these CLV discs have one spiral track rather like an LP and the speed of the disc is carefully controlled so that it spins slower as the laser reads more central parts of the disc. CLV players are cheaper, but they are practically useless for interactive video because they cannot show clear still frames.

Philips is still fighting to keep control of this market and its patents. The minimum price of a computer-controlled LaserVision player is around £1,100 (for the Pioneer LD–V4100). Complete turnkey systems are now coming on the market; for instance, Microvitec's single-user CUB workstation, which uses the LD–V4100 player, MIC 2000 interface, 602 or 702 infrared touchscreen and an IBM—compatible is expected to come in at under £3,000.

The main problem remains finding suitable laser discs to run on your CAL set-up. The range of discs is patchy. There are several in medicine and many for business training — but few in other fields. Some of the originating firms try to recover their development costs by charging several hundred pounds for each disc they sell. Some vendors treat them like LPs and charge for the reproduction (as little as £3 a disc) plus profit, making a total of between £10 and £35 per disc. A few firms use them as publicity gimmicks and give them away for free.

The cost of producing your own disc can be prohibitive. A professionally-produced half-hour video may cost from £10,000 to £50,000, but can be done a lot cheaper if you do a lot of the planning yourself. Even transferring stills to a master tape can be expensive; at 75p to colour-balance each slide, it can take more than £22,000 to fill a double sided disc with the 55,000 frames per side it can handle.

A lot depends on the video quality required. In the past, in order to maintain the reputation for high quality reproduction capable with laser discs, many firms would only make disc masters from high quality, wide bandpass video master tapes. This could involve hiring very expensive machinery; however many firms now accept lower quality tapes as input.

Once the master tape is ready it can be transferred to disc. There are basically three ways of doing this:

- A 'master' can be produced (for around £2,500) from which copies can be made. The cost for these falls exponentially with scale. The first few discs may cast several hundred pounds each; when production volumes get up into the hundreds, the cost per disc may fall to less than £3.

- If you are interested in only a few copies, they can be produced rapidly directly onto glass or plastic without a master; the cost then is about £300 to £500 each depending on quality. This is a good method for testing your video before entering large-scale production. Try Microvitec on 0274 390011 or Video Time on 01-439 1211.
- If you are unlikely to fill a whole video disc with your own material a French company called Futur*Vision will sell you part of a disc. A disc is issued every month. They sell the rest of the disc to other 'part' users. Your own piece can be erased on other discs or left in place if you require. The disc you end up with can have quite an eclectic collection alongside your own production! Prices depend on the quantity of material recorded and number of discs supplied; with three copies of the disc, six minutes of moving footage costs around FF15,500 (£1,550), while 750 stills cost about the same if supplied as slides or FF8,990 (£900) if already transferred to one-inch video tape. Futur*Vision is in Paris on (33) 142.03.00.43.

Hardware and software installation

The details will vary from machine to machine and the computer language used to write the program, but the principles will be the same. First you will need a suitable cable for serial communications. A cable for connecting an Apricot to the Philips VP835 laser disc player is shown below.

APRICOT		VP835
1 FRAME GROUND	(FRAME GROUND)	1
2 TRANSMIT DATA	(RXD)	3
3 RECEIVE DATA	(TXD)	2
5 CLEAR TO SEND	(CTS)	4
[computer pulls this high when ready to receive data]		
7 SIGNAL GROUND	(SIGNAL GROUND)	7
20 DATA TERMINAL READY	(DTR)	20
[player pulls this high when ready to receive data]		

The pin assignments for the Apricot serial port are fairly standard from machine to machine (except for the Xen-i IBM-compatibles) but you will have to consult your particular laser player manual — or phone the manufacturer if the manual does not list them! — for its pin assignments. The baud rate, parity and stop bits must be set correctly for your player; note that on some players these can be set to a range of values by DIP switches.

The method of setting up serial communications is an article in itself (and in fact has been: see APRICOT FILE Release 2.7 and 2.8). But with most laser players it is fairly easy, since most of the handshaking is done by software rather than hardware wiring. In other words, when one machine wants to send a command to the other (for example to pause), this is done by sending a software code along with all the other bits going to the peripheral along the data line, not by changing the voltage on a specialised control line.

If the laser player is fitted with an onboard microprocessor, this must be disabled (see the manual of your machine for the specific code to send) or it will try to control the player and get in the way.

The best way to control the player is to write a series of procedures, one for each control the player accepts — one for fast forward, one for stop, and so on. This is easily done in a procedural language (we use Pascal) but may be a little harder in some dialects of Basic. Each of these procedures can then be called from the body of an interactive teaching program as required.

```
Procedure KbdDump ;   {this procedure empties the
                       keyboard buffer}
Var
 Ch : Char;
Begin
 While KeyPressed do
  Read (Kbd,Ch);      {this leaves the buffer
                       empty}
End;

Procedure FWD;        {this procedure deals with
                       FORWARD PLAY}
Begin
 Write (aux,'N');{sends code "N" to aux port for
 forward play}
  Delay (1000);
  Write (aux, £13);  {sends CR to aux}
    KbdDump;         {empties keyboard buffer}
End;
```

The manual that comes with the player should list all the available codes. The only problems may come with codes which require a response or acknowledgment from the player; a few may require delays to give the player time to act. ◆

At Dundee University, we have produced a full set of laser disc control procedures for a VP835. One of us (Mark Whitehorn) is currently producing an authoring toolbox in Turbo Pascal 4.0 together with a book on CAL, which should provide all the basics needed by even the most computerphobic to produce an interactive video tutorial from scratch. All these products are available from the authors.

Readers' Writes

Your chance to have a say

Amending SDIR

While it does work correctly in MS-DOS 2.11, the DEALS FOR READERS utility SDIR.COM (version 2.50) cannot list an MS-DOS 3.2 directory without adding *.* on the end of the command — you can't use something like SDIR A:\BIN, giving just the directory name. This is due to two faults: SDIR incorrectly checks the return codes from MS-DOS, and MS-DOS 3.2 returns different values from 2.11.

This is easily corrected, however. Use DEBUG to Unassemble from location 4CD and you should find this:

```
A>  DEBUG SDIR.COM
    -U 4CD
    xxxx:Ø4CD B443  MOV  AH,43
    xxxx:Ø4CE CD21  INT  21
    xxxx:Ø4CF ØACØ  OR   AL,AL
    xxxx:Ø4D1 751B  JNZ  Ø53C
```

Replace that JNZ with this:

```
    -A 4D1
    xxxx:Ø4D1       JB 4FØ
    xxxx:Ø4D2       JB 4FØ
    -W
    -Q
```

If the code is not found, you have a different version of SDIR — try locating it somewhere else in the file.

The change corrects the checking of the return code from MS-DOS. Previously SDIR was checking AL (which was 0 in MS-DOS 2.11) for an error condition; MS-DOS 3.2 returns a value in AL even when there is no error (an undocumented feature). The correct procedure is to test the carry flag to see if an error has occurred, which is the alteration we have made here.

I have also come across an interesting bug which can apply when certain word processors are used to create batch files; it highlights the need to end all ECHO statements with CR/LF. The last line of my BAT file was an ECHO to display 'END OF BATCH FILE', and the word processor I use (PMATE) doesn't create an automatic Return with linefeed — so the batch file ended with the word 'file'. This caused an automatic ECHO ON before the ECHO was executed, so the result appeared on the screen as ...

```
ECHO End of BAT file
End of BAT file
```

G M Sturges
Trélex, Switzerland

SuperCalc dates in English

This macro puts today's date (in UK *dd:mm:yy* style) in two adjacent cells anywhere in a SuperCalc spreadsheet:

```
/GN/FE^[,R£
DAY(TODAY)+MONTH(TODAY)/100
^P/FE^[,L£
(YEAR(TODAY)-1900)/100
/GN
```

Line one assumes the default Next and sets the current cell to be right-justified with number formats (the ^[escape symbol is entered by ESC R). The formula presents the day and month of today's date as a decimal number. Line three moves the cursor right and sets a left-justified numeric format for that entry. The formula in line four turns the year into a two-digit mantissa, line five rests to Next. The net effect is that '20 August 1988' will appear in two adjacent cells as '20.08.88'

This is an alternative:

```
/GN/FE^[,I
DAY(TODAY)*1000000+MONTH(TODAY)
*1000+YEAR(TODAY)-1900
/C^[,^[,V
^VS^VS^VS/^VS^VS^VS^VS/^VI^VE"
/GN
^C
```

In this case line one formats the current cell as Integer. Line two converts the day into a number, with extra digits replaced by '1'. Line 3 copies the value on to itself, so that it can be edited, and deletes the formula. Line 4 does the clever formatting: CONTROL-S moves left one space, the slash overwrites a zero, CONTROL-I tabs to the left margin, and CONTROL-E changes value to text.

The result puts today's date in the style '20/08/88' into a single cell. The formula is deleted in this version, so there is no automatic update as before; this may or may not be desirable in a particular application.

Ted Webber
Papua New Guinea

Compatibility and FORMAT

For some time I have been using Apricot's FORMAT program version 1.5.0 (dated 21 Oct 85) to format 3½in disks in IBM/Microsoft format rather than Apricot's native version. The objective in so doing was to simplify the inevitable future upgrade to IBM-compatibility; I had every intention of stating with 3½in floppies rather than the flimsy creations sold in the 5¼in size, and I suspect a high proportion of Apricot users will share that view.

I have recently been using a Zenith 183 portable at work and that had no trouble reading disks formatted on the Apricot PC. I am now the owner of a Dell System 220 (makes greased lightning look slow!) and after setting up subdirectories I inserted the first ex-Apricot floppy to transfer data files. The response was most unwelcome: "Disk in Drive A has no label" (it had) and "File not

found" (there were dozens of them). A disk which had been formatted on the Zenith 183 was read correctly however.

Running CHKDSK provided the answer: disks formatted on the Apricot were being regarded by MS-DOS as having a 1.2MB capacity, with the files all shown as lost clusters. The Apricot FORMAT program obviously wasn't getting the disk boot sector quite right, possibly because it predated MS-DOS 3.30 and the 720KB or 1.44MB type of 3½in floppy drives as fitted to my Dell and the current Apricot Xen-i range.

The solution was simple but not terribly convenient. Disks formatted on the Dell worked perfectly in the Apricot, so it was merely the time-consuming matter of using the Apricot to copy files across to Dell-formatted floppies and then copying them to the appropriate subdirectory on the hard disk.

Moral: *always* run a new computer in parallel with the old until you are certain that the old system is genuinely redundant. The temptation to sell the old'un at the same time as you take delivery of the new helps cashflow but needs to be balanced against the frustration of trying to transfer files between not quite compatible disk formats.

Mike Florey
Croydon

PS: I think there is a later version of FORMAT that may not suffer this problem, but don't bank on it without some prior experimentation. Otherwise, the changeover to Dell has gone remarkably well even if boldface and underline have to be represented on screen by different colours rather an accurate display à la Apricot.

We have been using FORMAT version 1.5.4 for about a year now without experiencing much trouble. Oddly enough, however, version 1.5.5 is to be avoided — formatted disks are left with multiple bad sectors. The problem disappears if you revert to 1.5.4 — Ed.

The Last Word

My Apricot left me a few months back after a full and happy life. Like many other owners I have had to settle for a you-know-who AT-clone, and every day I realise what I lost in the change ...

But due to this unfortunate and untimely death, I now have no further use for APRICOT FILE and will not be renewing my subscription. I'd like to thank you for all the interesting articles, for the DEALS FOR READERS disks, and generally for all the invaluable help I found in reading APRICOT FILE over the years. I wish you the best of luck for the future.

Alain de Senarclens
Neuchâtel, Switzerland

Classified

Free Small Ads

Apricots for sale

■■■ XEN 1MB RAM, 20MB hard disk. Includes 12in green monitor, cables (two parallel, one serial) and standard software. Much-loved and trouble-free, but I have to downgrade to IBM in order to upgrade my software ... Offers over £1,750. Call Andrew Dodd 01-602 1876 anytime.

■■■ APRICOT PC (twin SS drives) with 512K RAM, 9in and 12in monitors, Qume Sprint 10X/35 (35cps) daisywheel printer, Microsoft Word version 2 (unopened). Will accept £690. Contact John Marcus, 0753 886882.

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

■■■ APRICOT PC: £300. Excellent condition: 256K RAM, 9in green monitor, two 3½in 720K double-sided disk drives, Super series with manuals plus other software, MS-DOS 3.2 and BIOS 3.1.2, assorted TP Group disks and public domain programs. Will pay half the shipping costs (needs only an inexpensive 220V converter to work in UK). Al Thibideau at 100 LaSalle Street 17a, New York NY 10027, USA, tel (212) 666-2749.

■■■ APRICOT F10 (10MB hard disk): £350. Lotus 1-2-3 plus comms software plus utilities plus APRICOT FILE back issues: £50. Ring Jack on -1-946 5611 (day) or 01-946 9779 (evenings).

■■■ GIVE-AWAY PRICE for APRICOT PORTABLE — 256KB, single 720KB drive, all Super software. Keyboard needs repair, hence £70 ono. Bob Birks, 7 Leonard Avenue, Baddeley Green, Stoke on Trent ST2 7HW.

■■■ MICROSOFT COBOL compiler on two 5.25in disks with manuals. Best offer secures. Bob Birks, 7 Leonard Avenue, Baddeley Green, Stoke on Trent ST2 7HW.

■■■ APRICOT Xi 20MB hard disk machine — excellent condition, plenty of software: £985. Day 0384 66312, weekends 061-766 6232.

Apricot extras

■■■ PHILIPS 12in green-screen 80-column monitor: £55. Call 01-642 5380.

■■■ ALMOST UNUSED — CARDBOX PLUS version 3 and SAGE BOOKKEEPER, both for Apricot. £30 each. Call Godfrey Hodgson, 0865 56962.

■■■ F10 EXTRAS — optical mouse, new: £65. Transformer, new: offers. Day 0384 66312, weekends 061-766 6232.

■■■ ALPHA DISK 5¼in external floppy drive for Xi/PC, reads/writes IBM PC format. Also three Internal MODEMS with generic COMMUNIQUE. All in perfect working order. Best offer secures any item. Call Isobel on 04675 210.

■■■ PRINTER — Commodore MPS 1000 (actually Epson LX80): £130. Call 01-642 5380.

Review

Future imperfect?

Hands-on experience with WordPerfect 5

By David St John Wallis

When you decide to sit down to try and do something a bit serious, like reviewing the latest version of the most popular word-processor in the world, you try to plan how you're going to approach the thing. Are you going to review *all* its features, or tell the reader what it's like to use, or talk about learning curves?

Well, I thought I'd do all these things. But then sheer frustration with the program got me whinging and I thought I'd have to moan about what is wrong and leave any other review until such time as Wordperfect Corp of Utah gets the thing right. So here goes.

As soon as I read old 'Fletch' Bromhead's article about laser printers I rushed out to buy an Epson dot-matrix because I *must* have text and graphics. And then, of course, one simply *had* to have the WordPerfect 5 program. An error? Maybe. Someone once said that you should never buy version 1 of anything. I won't bore you with what the program can do — professional writers can get on with that. All I'll say is that it seems to do practically *everything*. And at three times the price of two weeks of lust and Red Barrel in Benidorm, so it jolly well should.

... But it doesn't. At least, not perfectly. In fact, a l-o-n-g way from perfectly in a lot of respects.

Hands on

Here are some of the things which went wrong with my copy (purchased, you may be amazed to hear, rather than whipped off a wily Oriental for the price of the disks. Maybe old Fletch was right: nick it first, find all the bugs, and then wait for the authors to find them too.)

- The GRAB facility, for grabbing an image off the screen for incorporating into text is fairly moribund and yields pretty awful images. (On a scale of 0=horrendous to 10=super, this gets 1/10).
- There was no printer driver for my Epson LQ2500+ and in consequence I got 'feature unsupported' messages (5/10).
- Screen rewrite (WP-speak for losing characters from the screen) is sometimes very slow (or at least dependent on some keyboard input — to the point where you think the program has lost some of your text. Why have a rewrite of screen anyway? I never had this trouble with Superwriter (6/10).
- The CURSOR utility, which changes the cursor's shape, didn't work within WordPerfect 5. As a result, the cursor is often hard to find (5/10).
- INSTALL crashed, apparently because some spotty Mormon forgot to include the Help files (1/10).
- SPELL has taken all night to fail to add part of my postal district (GU35) into the dictionary. Sentinel: 'We're working on it' (2/10).
- The program failed to add in my own dictionary of some 23,000 words compiled from scratch within SuperWriter. This is a special drag since it has the names of all my friends in Benidorm (4/10).
- There is no facility for adding a default drive for generated files — in Superwriter you can put a /DDISK command into SW.DEF (7/10).
- There doesn't seem to be a specific command that works to control the *number* of characters on a line. This can matter if you're e-mailing material and the service doesn't like long lines of text (8/10).
- Indent sometimes doesn't work — just now it didn't on the second line of an indented block. And other things simply don't do what you expect or don't work. For example, pressing F3 for Help often doesn't help; it just dumps you back to where you were (6/10).
- And sussing out what's an A4 sheet of paper — only the most popular paper size in the known galaxies — requires a bank loan to pay the phone bills for the talk-through from Sentinel (8/10).
- Another thing which gives a lot of grief is the frequent failure of the search-and-replace function to search and replace. An example is having the program look for 'hard' returns — [HRt] in WP-speak — and replace them with a space. This is more or less essential if you're editing a piece of text saved in ASCII. WordPerfect 5 tends to replace two or three strings, let out a squawk and put you into the bit of text where it had the vapours over the immensity of the task. Oh dear! (2/10).
- That wild use of function keys which I have moaned about with at least two prior versions of the program makes it too easy to press say CTRL F3 for Screen when what you wanted was CTRL F3 for Spell (5/10).
- And some things are just irritating — for example, WordPerfect 5 will insert a BOLD code before and after a piece of text which is already written, but then just ignores output at run-time. To get the program to recognise such codes you have to BLOCK the text, get into the FONT menu and select APPEARANCE then the BOLD option. Or do something else. Woe! (6/10).
- Perhaps the worst problem of the lot is the program's willingness to crash. At first I thought this only happened because of the vast number of terminate-and-stay-resident programs that I use. But WordPerfect crashed when spelling (or rather when not spelling). It crashed when saving, yielding up a bright blue screen and nothing else. It crashed when previewing —

a real pain because I hadn't yet taught myself how to use the 'timed backup' capability and lost a lot of words (0/10).

Morals

Well, I suppose some of these moans are fairly trivial. But the Epson printer I use was until recently the top-of-the-line model; a punter would think WordPerfect Corp might have heard of it.

And word processors are all about words; words are all about spelling — and the spell-checker is *terrible*. Gosh! It keeps on failing to remember my name. And just now I was delighted to see that having corrected the spelling of 'WordPerfect' from 'Wordpefect', after a bit of screen re-writing (that WP-speak again) it reverted back to my misspelling!

So, right now you get a word processor which crashes and which can't spell and can't print perfectly. I expect that when I re-read this article I'll wonder why I was making all that fuss. But right now I am disappointed. If I'm forking out nearly £500 I expect all the propellers to be screwed on, the champagne chilled, the wing-flaps oiled and the flight to be on time. *This* plane has been hijacked.

Ok, the program's clever — *very* clever, in places. And once you've got the hang of things it makes churning out *Doodles* a lot quicker and more colourful. But maybe it's just *too* clever; and maybe it wasn't Beta-tested for long enough.

Moral: wait for version 5.1 — or at least for a copy of 5.0 with help files.

Stop Press!

My moans may be answered — the charming Kay at Sentinel has said she will send me a completely new and revised bunch of disks.

A few days later

I am beginning to feel like a chap who has bought a car and has found bits falling off and missing.

The latest one today is that the 'hyphenation' module calls a file called something like {WP}WP{WP}HY.HYC (lots of WordPerfect 5 files have names like this). Presumably this does something like check if your hyphenation decisions are as good as those of the WordPerfect wizards chewin' backy and supping Bud in the tumbleweed salt lakes of Utah.

The problem is that the file isn't there. And now my WordPerfect 5 disk count is up to *fifteen*; it made my arm ache bunging all the disks back into the floppy to look for the missing file.

But today's new disk has a revised speller on it which didn't mind my postal district number at all. It did however object to the word 'lost'. It suggested 'los' — as in Los Angeles. ◆

Diary dates

On with the Shows

A round-up of potential relevance

October 1988

24-28 International Business Show *London.* Big office show at Earls Court — around 200 exhibitors covering everything for the office. Tends to major on furniture and office consumables rather than computer-based equipment, though. *Organiser: AGB Exhibitions, tel 01-868 4499*

26 October — 2 November IBM '88 *London.* IBM's very own show, the company's major showcase for its products (and there are several new IBM machines this year). Last year's inaugural event was done very well, probably because it comes from the PR budget rather than as a straight commercial venture. *Organiser: IBM, tel 01-578 4399*

November 1988

14-18 Comdex Fall *Las Vegas.* By comparison with Comdex Spring, this one has nearly twice as many exhibitors for the noisier and probably more important half of the giant twice-a-year US show for the computer industry. *Organiser: Interface Group, tel [USA] 617-449 6600*

15-17 Computers in the City *London.* Financial services computing show at the Barbican. Gets bigger and better every year. *Organiser: Online, tel 01-868 4466*

13-15 Info North *Manchester.* The Northern version of BED's everything-for-the-office show. *Organiser: BED, tel 01-948 9900*

December 1988

6-8 Interactive Video '88 *Brighton.* Fledgling event for a fledgling corner of computing; could be interesting though, especially for users involved in training and education. *Organiser: PLF, tel 0733 60535*

January 1989

27-29 Videotex User *London.* Another of EMAP's 'user'-suffixed shows. Now that viewdata is finding serious vertical markets in the City, this event at the Barbican may have more relevance than appeared in earlier years ... *Organiser: EMAP, tel 01-404 4844*

February 1989

21-24 Which Computer Show *Birmingham.* Biggest and best of the computer shows on offer, but with the current state of the industry (and the purchasers) the trek to the NEC might be justified only to get a snapshot of how the manufacturers and vendors see themselves making money in the next couple of years. Still, if you want to go to only show in 1989, this should probably be it. *Organiser: Cahners, tel 01-891 5051*

A Turbo Pascal Primer

Part Nine: Strings

By Mark Whitehorn

In last month's episode we had a look at arrays. This time we will touch on *strings*, which are simply a special form of an array.

A string is a variable in which we can store a collection of characters. A programmer might write a program which began by asking the user for his/her name: the user's reply would consist of a collection (or string) of characters, such as "Penguin Penguinsson". This string could be stored in a string variable.

A string is normally declared like this:

```
Var
Name : String[30];
```

In this example we are calling the string *Name* because we're going to store the user's name therein. We have also decided that the user's name is unlikely to exceed 30 characters, so the length of the string can be limited to that.

In some ways we can think of this sort of declaration as just being a quick way of writing ...

```
Var
Name :Array[1..30] of Char;
```

If you try to replace the former with the latter, the compiler will actually compile the code — but it will not accept statements like *Readln(Name)*.

Manipulating strings

As I said, strings are a special case. The close relationship between string variables and arrays of *Char* allows us to manipulate the data in a string as if it were in an array.

Suppose you declare a string like so:

```
Var
Name : String[30];
```

Then you can put an actual name into the string, either like this:

```
Name := 'Penguin Penguinsson';
```

Or this:

```
Readln(Name);
  {in which case you are expecting the user to
  enter the name}
```

This is standard string manipulation. But can also treat the string as if it were an array, for example like this:

```
Writeln(Name[18]);
```

This would write the eighteenth letter of the string to the screen — which for the example above would be the letter 'o'.

String length

The first letter of the string is in *Name[1]*. What, you might be tempted to ask, lurks in *Name[0]*? Answer: a character which gives the actual length of the string. Remember that the string *Name* is capable of holding 30 characters. This doesn't mean that it is duty bound to hold that number, and usually it will contain fewer. In the example above, I have put in 19 characters (the space counts as a character); so the actual length of the string is 19 not 30. And if we ask for the contents of *Name[0]* ...

```
Writeln(Name[0]);
```

... we would get on the screen a rather peculiar character which looks like two exclamation marks side by side (it is ASCII character 19).

When manipulating strings, it is sometimes very useful to be able to find out the exact length of a string. We can decode it into plain English with *Ord*:

```
Writeln(Ord(Name[0]));
```

The *Ord* function gives the numerical value of the ASCII character referenced — which, lo and behold, is '19'.

But this is getting to look a little messy. So the boys and girls at Borland gave us a built-in function called *Length* to return the length of a string. It can be used thusly:

```
Writeln(Length(Name));
```

Of course we don't always just want to read the length of a string. Sometimes we need to change its length manually. We can then do this sort of thing:

```
Name[0] := Chr(14);
```

Be very careful when you start playing around like this. Don't be tempted to set this to a value greater than the declared length of the string (in this case 30) otherwise your program can come tumbling down around your ears. But it's fairly easy to be careful about this; and once you start manipulating strings, you will probably find it useful to be able to change this value.

Note also that ...

```
For Count := 1 to Ord(Name[0]) do
Write(Name[Count]);
```

... or indeed ...

```
For Count := 1 to Length(Name) do
Write(Name[Count]);
```

... can be translated into English as "for 1 to the length of the string called *Name*, write out each letter of the string". In other words, either of the above expressions could replace ...

```
Write(Name);
```

Clearly it is easier to use *Write(Name)*. All I am trying to show is that a string can be manipulated in much the same manner as arrays.

More on Index types

I said in the last episode that the *index* type (which is the one which tells the compiler how many variables are in the array) is usually of type *Integer*. This is quite true; but it has prompted some people to ask "When would it ever be anything else?"

An excellent question, if only because it enables me to demonstrate even more things to do with arrays on wet Sunday afternoons.

Suppose that you worked for a secret organisation such as the Masked Penguin League (which is pretty secret). Naturally you would want to keep your documents coded so that none of your friends knew that you were a member. You might want a program which would take an ASCII file from disk, go through it letter by letter, substitute a different letter for each original one, and then write the encoded file back to disk. Of course, you would hope to be able to reverse the process at some later stage!

The demonstration program ENCRYPT.PAS does this, except that (for the sake of brevity) it operates on a string which is keyed in rather than on a text file which is read from disk. (If you want to expand the code to read files from disk, see article 3 in this series — APRICOT FILE Release 3.3.)

This piece of code demonstrates the power of using arrays and of using index types other than *integer* — it is possible to write a neat, elegant piece of code which does a lot of work in a few lines.

The demo program

At the start of this series I indicated that I would avoid just this sort of short solution to problems because, by their very nature, such programs are more difficult to follow. I feel however that it is necessary here as a demonstration of how they can be used to their best advantage.

Since the code is a little cryptic (ho ho) I will pull it to pieces here, rather than plastering the program with comments.

Setting up an array:

```
Code : Array[Char] of Char;
```

This line sets up an array which can hold characters. How many different variables have we declared in the array? To answer this we look at the Index type — which in this case is *Char* (to be found in *Array[Char] of Char*). Since the index type is of type Char, and there are 255 different characters, we know that the array can hold 255 different values. Do not be misled into thinking that we used the word *Char* here just so that we don't have to write '255'; there are 255 different variables declared here, but they are not all called *Code[1]*, *Code[2]*, and so on. They are actually called *Code['f']*, *Code['g']* and so on — 255 of them, one for each of the different ASCII characters.

Filling the array:

```
For CharCounter := ' ' to '}' do
  Code[CharCounter] := succ(CharCounter);
```

This is where you set up the actual encoding system that is used. Here I have chosen a very simple one, substituting for each letter the next letter in the alphabet; the code puts values into most of the variables in the array, starting at ASCII code 32 (a space) and ending at ASCII character 125 (the closing curly bracket). Between these lie most of the normal writable ASCII characters. The *For..Do* loop runs through all of this part of the array doing ...

```
Code['a'] := 'b';
Code['b'] := 'c';
Code['c'] := 'd';
```

... and so on. So whenever we ask for the contents of one of these variables, for example *Code['b']*, we'll get the character 'c'.

Producing the coded message:

```
For Count := 1 to Length(StringToEncode) do
  EncodedString[Count] :=
  Code[StringToEncode[Count]];
```

Each letter of the string is individually changed to the next letter of the alphabet. Let's look at a specific example to see how this is done.

Suppose that *StringToEncode* was "Hello there". The first time this loop executes, *Count* is equal to 1. So the expression ...

```
EncodedString[Count] :=
Code[StringToEncode[Count]];
```

... simplifies to ...

```
EncodedString[1] := Code[StringToEncode[1]];
```

We know that *EncodedString[1]* is the first letter of the encoded message we are creating. *StringToEncode[1]* is the first letter of the message we want to encode, which in this case is 'H'. *Code[StringToEncode[1]]* therefore becomes *Code['H']*.

So now we know that the first letter of the encoded message will be equal to *Code['H']*. If we look up in the array, we find that *Code[H]* is in fact the letter 'I'. So the first letter of the encoded string is 'I'. Phew!

The loop then repeats for the remainder of the string.

```
EncodedString[0] := StringToEncode[0];
```

We have been messing about with *EncodedString*, throwing in characters willy-nilly. Meanwhile, *EncodedString[0]* has remained unchanged. If we don't set it to the new length of the string, we won't get the correct output. So this line sets the actual length of *EncodedString* to the actual length of *StringToEncode*.

```
Writeln(EncodedString)
```

This just writes out the encoded string.

I have made the code slightly longer than necessary, in order to give myself an easier task in explaining what is going on. We can save some time and effort by causing the encoding to occur in the actual *StringToEncode*:

```
For Count := 1 to Length(StringToEncode) do
  StringToEncode[Count] :=
  Code[StringToEncode[Count]];
```

Thus we can dispense with *EncodedString* altogether, and we don't have to set the length of the string manually since it will not have changed.

Clearly this program needs some work before it is useful. For a start, it can be made to read files from disk. And at present the program just removes a character and substitutes the succeeding ASCII code: the use of an array is overkill here, and we could find easier ways of doing this particular encryption — for example:

```
For Count := 1 to Length(StringToEncode) do
  StringToEncode[Count] :=
  Succ(StringToEncode[Count]);
```

The real beauty of using an array, as demonstrated her, is that you can set up the array in any complex and devious manner which occurs to you. Once it has been filled with characters, the substitution is very easy and very quick. (What about encryption based on a code word?)

Next time you find yourself with a wet Sunday afternoon which has got out of control, have a go at an encryption program — it's better than the cryptic crossword! ◆

ENCRYPT.PAS

```
Program Encrypt;
Var
Code :array[Char] of Char;
CharCounter : Char;
Count :Integer;
StringToEncode,EncodedString : String[255];

Begin
ClrScr;

For CharCounter := ' ' to ')' do
Code[CharCounter] := succ(CharCounter);

Writeln('Please enter a string to be
encoded.');
Readln(StringToEncode);

For Count := 1 to Length(StringToEncode) do
   EncodedString[Count] :=
   Code[StringToEncode[Count]];

EncodedString[0] := StringToEncode[0];

Writeln(EncodedString);

End.
```

Review

The Apricot Bulletin Board:

GABBS reviewed

By Dr Stuart Bunt with Simon Potter

When Apricot plc went over to IBM-compatibility, many of us with standard-issue Apricots felt abandoned — left with no option but to go cap in hand to the nearest branch of Dixons to buy an Amstrad and join the True Fraternity of Big Blue Worshippers. But fear not, someone (apart from Ansible Information and APRICOT FILE) still loves us! Deep in the heart of Hampshire lives GABBS, the Gosport Apricot Bulletin Board Service, set up and operated by Stephen Cole.

Run for a long time on an Apricot Xi, GABBS has always had a strong Apricot following. The board features lots of public domain software utilities which actually *run* on Apricots. Its message facility is also a mine of useful information for Apricot owners — and it has come to my rescue before now. I had a student who managed to destroy my original £350 copy of Microsoft Assembler before I had made a backup (and with her thesis due in a week). We left a message on GABBS, someone downloaded the program for me, I picked it up, and was back in action the next day!

The system

Stephen Cole runs a tight ship and is often on hand to answer voice calls. (There is even a 'yell for Stephen' option on the board's main menu — the computer will buzz him and if he is around he will answer the phone, often with background comments along the lines of "not the **** BBS again!" from family and friends.)

Stephen may also type his replies in real time. It can sometimes be a little disconcerting when the computer starts answering you back, particularly when you have got yourself into a knot with the menus; it's usually Stephen giving a helping hand!

The bulletin board has gone through several incarnations over the last few years. At present it runs on a Tandon PC with an 80286 accelerator card and a 40MB hard disk. It uses SEAlink QuickBBS software (shareware, registration fee $25).

GABBS has links with many other BBSs in Britain and the continent (Holland at present); so it can truly claim to be an international bulletin board, and must be a great boon to those many forsaken Apricot owners left in mainland Europe.

How to connect

Set your software to 8 bits per character, one stop bit, no parity — pretty standard for UK bulletin boards. GABBS uses a Dowty Quattro, one of the best all-singing all-dancing modems on the market, so you can set your own modem for 300/300, 1200/75, 1200/1200 *or* 2400/2400 baud. There's a lot of text to be read, of course, so faster speeds are preferred — in fact, to stop the BBS lines getting clogged up GABBS doesn't allow downloading (copying files from the BBS) at 300 baud.

Now phone Gosport (0705) 524805 and catch the line when it is not engaged — there were 12,000-plus calls to GABBS last year, an average of some 30 a day. Grab the carrier quick or the Gosport modem will drop the line; then send an ESC to wake it up and start the log-in procedure.

There is no charge for calling GABBS, but if you have not used the BBS before you'll be asked to register before you can be allocated a user number and a password. You have to leave your name, address and phone number; there are also questions about your machine and software. Until your application is processed you will only be allowed a short time on the system, but subsequently you can use a total of 60 minutes per 24 hours.

Stephen does his best to keep vandals and wallies off the BBS, and all graffiti is removed within 24 hours. As with all public bulletin boards, however, you should beware of viruses — so put a condom on all downloaded software before you run it!

I use a Digisolve ChipChat CC2123 modem at 1200/75 bps and the DATATALK comms package from Datasoft. DATATALK is one of the neatest pieces of software I have come across and I can highly recommend it. It has all the features you could ask for: forgot a file name? Press Return and you get a directory. Want to edit a file to send? No problem, call up the built-in editor.

DATATALK is expensive, however, and it runs well only on Xis or Xens with green screens; it doesn't seem to like the paper-white monitor on my Xen, splitting the screen in two with alternate lines going to each screen ... very confusing! To get around this problem I use a converted Apricot POINT 32 (take one POINT 32, add a keyboard and monitor, boot from a system disk, and you have a 20MB Xi. As they say, "not a lot of people know that"; certainly the people at Apricot plc aren't going to give anything away!).

Luckily there are some good public domain utilities about for comms. KERMIT (on DEALS FOR READERS disk AP021) is one of the most ubiquitous, but I would miss DATATALK's ability to echo input to disk or printer ...

Moving around the board

Basically, there are four main types of activity on GABBS (and on any decent bulletin board):

- **Private messages** can be posted and read. In this case GABBS acts exactly like a conventional electronic mail system — only the named addressee can read the message.
- **Public messages** can be posted in a kind of general mailbox for anyone to read. Or they can be posted in one of the 'echomail' areas, sometimes called 'conferences' or 'Special Interest Groups' (SIGs); these are for people with a shared interest of some kind, but anyone who cares to look into a particular SIG can read any non-private messages posted there.
- **Bulletins** of information from the SYSOP (system operator) and small ads can be posted and read.
- **Files** can be copied from the board (downloaded) or sent to it (uploaded). These are usually public-domain programs, and GABBS has a number of separate file areas associated with SIGs for people with particular computers.

We've summarised the principal options overleaf. They correspond more or less to the single-key selections offered by the main menu; that is the principal route-map for GABBS, though experienced users who know what they get can skip the menu before it appears. There's obviously no manual issued with membership of GABBS, so the first-time user (or old users with poor memories, like me) will find it helpful to keep an overview of the menu structure beside them.

As you can see from the panel below, the menu lets you communicate with SYSOP Steve Cole either by voice (the Yell option), by message (use **P** to send him private mail, or send him a message as part of the Goodbye procedure), or download a file (via the **S** option). The Utilities sub-menu provides for altering your set-up parameters, getting information on your present status, and checking for personal mail. **W** calls up a (small) small ads section. **H** scans all recent messages in all the 'message areas' at once; **P** lets you post a message for private or general consumption.

Apart from the **P** electronic-mail option, the one you're most likely to go for is **S** for specific interests. Apricot users are given their own spot on the main menu for direct access to the Apricot SIG area, reflecting their greater use of the BBS, but you can also get to the Apricot area via **S**.

GABBS runs under version 2.01 of SEAlink's public domain BBS manager QBBS, which is menu-driven — up to three menus before you get to precisely what you want. Type-ahead is always available, though; if you can remember the menus, you can type the requisite code letter at any time and the BBS will drop what it is doing and leap straight to that option.

Setup

When you first access GABBS, you'll have to run through a 'new user registration' question-and-answer procedure. In part this elicits from you a description of what your computer can and can't do in the way of displaying information; those parameters are subsequently associated with your password, so they apply when you next sign on.

The Setup option lets you change those settings, producing a sub-menu that indicates just what you can alter:

```
<P>  Password change
<C>  City and State
<L>  Length of screen
<A>  Toggle ANSI Graphics on/off
<S>  Screen Clear toggle
<T>  Toggle page pausing
<E>  Toggle full screen editor on/off

<Q>  Back to TOP Menu
<G>  Goodbye (Logoff)
<M>  Back to Previous Menu
```

Most of these are self-explanatory. The ANSI graphics used are only compatible with VT100 and true ANSI emulation, and in general it's safer to leave them OFF unless you have an IBM-compatible. The full screen editor is not implemented as it crashes on-line in the present version of QBBS.

```
                GABBS  Main  Menu

              <R> Re - Register with GABBS

<C> Change Setup                <U> Utilities i.e Check Mail
<B> Bulletins from Sysop        <F> File Areas
<M> Message Areas               <A> Apricot users Start Here
<P> Public and Private Mail     <W> Sales and Wanted Adverts
<Y> Yell for Stephen            <G> Goodbye  (Logoff)
```

Utilities

Selecting the Utilities option from the main menu also includes a 'change parameters' selection among its alternatives:

```
<V>  Display version of QBBS
<D>  Display system usage chart
<U>  Search for a user of GABBS
<T>  Time on line
<M>  Check for Personal Mail
<C>  Change your own settings on GABBS
<Y>  Yell at Stephen Cole
<G>  Goodbye (Logoff)
<Q>  Back to Main Menu
```

The usage chart is interesting. It shows a surprising amount of daytime use (company phone bills?) — it starts at 6am (argh!), rises steadily to a peak at 11pm (fed up with TV or programming?) and then falls off to nothing at 4am.

'Echomail'

The major part of the bulletin board is given over to 'echomail' areas for messages (at the time of writing there are 8,912 messages on the system!) and for up/downloading files. Select Message Areas and you'll see this:

```
<P> Electronic Mail (local to this BBS)
<N> Network Mail
<*> Combine message areas

 +++ Echo Mail Areas +++

<1> Sales              <2> Hardware
<3> Common Room        <4> UK BBS news
<5> DBASE Echo         <6> IBM
<7> Programmer 'C'     <8> Apricot
<9> Amiga              <A> Amstrad
<B> Amateur Radio      <C> Atari ST
<D> Binkley            <E> Communications
<@> All area Read...

<G> Goodbye
<M> Main Menu
<Q> Quit to previous menu
```

With the 'Electronic Mail' option you can send a private message to another GABBS user, or post a public message for anyone to read. 'Network Mail' allows you to send a message to someone on another bulletin board — clearly you have to know the destination, but associated BBSs are to be found all over Britain and much of the Continent.

The meat of the bulletin board lies in the 'Echomail areas', the SIGs. Most of these are computer-specific, but there are some general message areas — the COMMON ROOM is for miscellaneous non-computer chat which ranges from the banal (a lengthy conference on why fruit Polos are now extinct) to the intense (a debate on gun control and fox-hunting), UK BBS NEWS carries info from other bulletin boards and general inter-SYSOP chat. (You can also get at these general areas by selecting the H command back at the main menu; that way you won't be invited to look at the SIGs.)

DBASE ECHO is a database forum mainly for dBASE users, though other databases get a mention occasionally. The ATARI section is for ST users, the AMSTRAD area covers the CP/M-based micros (not MS-DOS — PC users should go for the Big Blue area). The PROGRAMMERS area is international and mainly covers Zortech C (formerly called Zorland C): lots of interesting files here. Comms is a special interest of Steve Cole's — and the files of comms programs available here include the very same QBBS software that GABBS uses. And there's an active and large SIG for amateur radio enthusiasts (want to track a satellite? Here's a program, written in MSBASIC no less!).

Each of these areas has the same options:

```
<E>  Enter Message
<R>  Read Messages
<S>  Scan Messages
<T>  Quickscan Messages
<F>  File list
<L>  Download files
<U>  Upload files

<G>  Goodbye  (Logoff)
<Q>  Quit to Previous menu
<M>  Main Menu
```

Most of these are self-explanatory. Entering a message can be done simply by typing it at the keyboard, or you can upload a previously-prepared text file; you can make the message private (so that only the named addressee can read it) or public (everyone gets the chance to see it).

As with commercial electronic mail systems, messages have an 'addressee' line and a 'subject' summary (the sender's name is automatically appended too). The scanning options show you the 'to', 'from' and subject lines, along with the first few words of the text.

On selecting one of the options for looking at Messages, you are asked whether you want to search forwards (from a past date to the present) or backwards (starting with the message most recently entered). You can also opt for new messages (entered since you last looked at GABBS) or for 'individual' messages — those addressed specifically to you.

This is the kind of thing you'll get from the 'scan' option:

```
Active Message Board: "Apricot Echo" (219
active messages)
System has messages numbered 1 to 8912.
Enter number to start at, [Enter]=Last:

Would you like to mark messages for later
retrieval [Y/n]? N

Press N for next message, S to stop, or P to
pause.

Scanning messages, please wait ...

Message #8448 "Apricot Echo"
Date: 13-Jul-88 18:12
From: Simon Hedger
To:   Andrew Bessant
Subj: Re: PORTABLE BIOS 3.1

Message #8447 "Apricot Echo"
Date: 12-Jul-88 07:52
From: Andrew Bessant
To:   Simon Hedger
Subj: Re: PORTABLE BIOS 3.1

Message #8438 "Apricot Echo"
Date: 01-Jul-88 13:48
From: Gerard Bavelaar
To:   Andrew Bessant
Subj: KERMIT

Message #8437 "Apricot Echo"
Date: 01-Jul-88 13:41
From: Gerard Bavelaar
To:   All
Subj: Kermit 2.29

Message #8339 "Apricot Echo"
Date: 16-Jul-88 21:46
From: Stephen Cole
To:   All
Subj: Trinity II

Message #7982 "Apricot Echo"
Date: 13-Jul-88 21:52
From: Pete Hurst
To:   Simon Hedger
Subj: Re: PORTABLE BOOT ROM
  .
  .
  .
```

Scanned messages can be marked or selected for later reading via the <S> and <M> commands.

Selecting the File list option from one of the echomail area sub-menus takes you into yet another menu and from that into a list of files on the system — mostly public domain utilities. If you want to download something (get a file from the BBS) or upload (send a file to GABBS), the relevant options will present a menu asking for selection of a file transfer protocol such as XMODEM.

Apricot EchoMail

The Apricot area has over 200 messages at the moment, some them scavenged from other BBSs — which brings Apricot questions and answers together from all over Europe, a very effective use of electronic mail.

To judge from the number of questions on the Portable, Dixons is not giving much support. Comms also features largely, particularly the infernal Apricot modem; this is not so surprising, of course, since each questioner must use comms to get to GABBS!

In the Apricot file area there is a good (but limited — 42 files) selection of public-domain programs for Apricots. Perhaps it is time for some Apricot owners to upload some more software? It is bad manners just to download all the time. Some BBSs keep a record of a user's download/upload ratio and frown on continuous poaching of free software with no contributions.

Files can be sent or received using XMODEM, WMODEM, and ZMODEM (but not the SEAlink or YMODEM protocols). Most of the larger files are 'archived' — compressed to remove spaces, repeated characters and the like; the ARC.EXE utility to 'unpack' these is also available on the BBS (or on DEALS FOR READERS disk AP018).

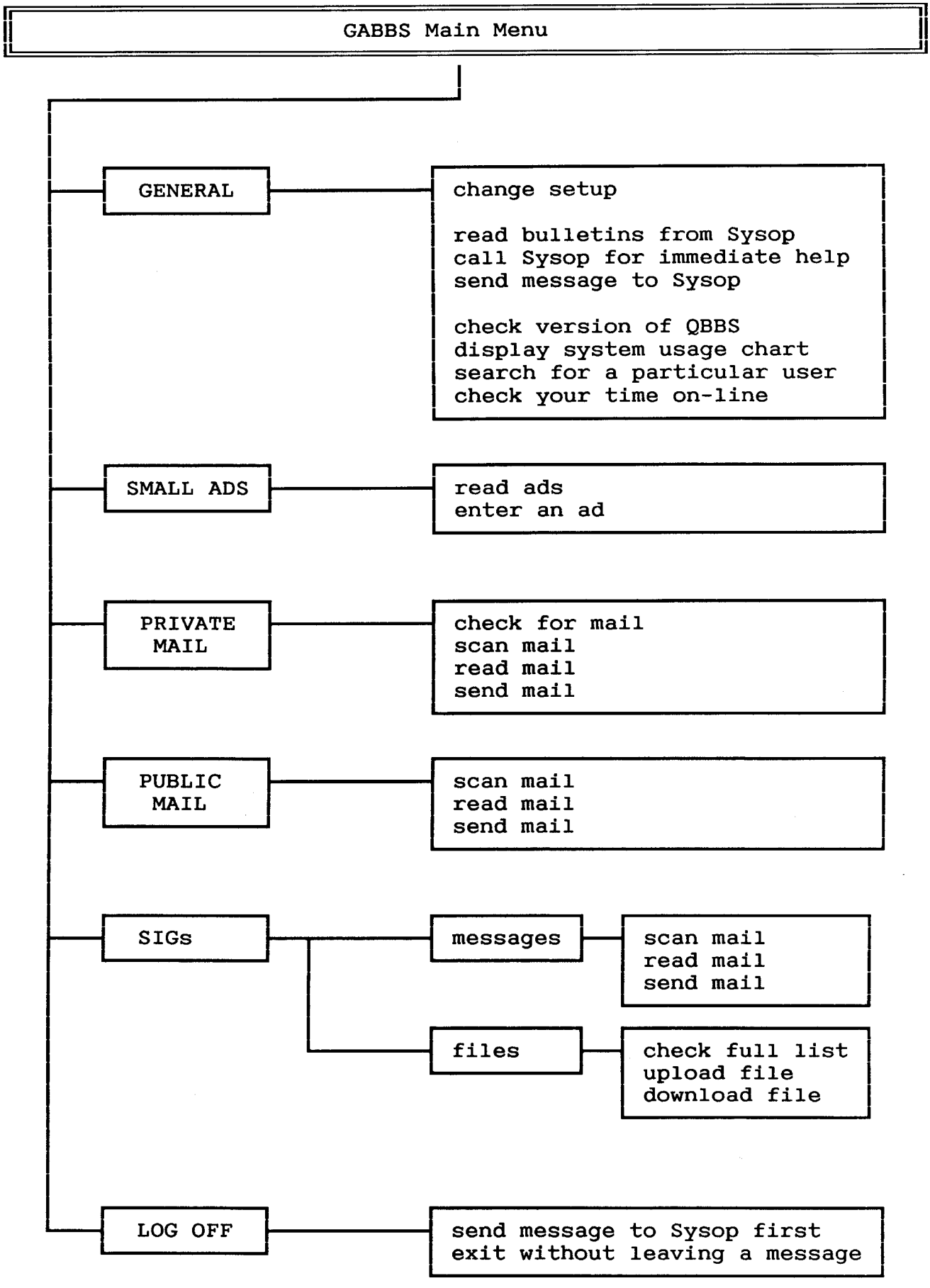
Log off

GABBS is a great bulletin board. It does everything a BBS should — it informs, it provides a centre for debate, it offers a very economical alternative to commercial e-mail services, it has public domain files of use to many users, it's simple to use, and it is well maintained by an experienced (and knowledgeable) SYSOP.

If you're an Apricot user with a modem, phone soon and register — ultimately a bulletin board is only as good as its users. The more you put in, the more everyone benefits! ◆

```
     ++++++++++++++++++++++++++++++++++++++++
     + Goodbye, Logoff and Disconnect etc!!! +
     ++++++++++++++++++++++++++++++++++++++++
     + <L> Leave mail to Steve then Logoff!! +
     + <G> Logoff NOW!                       +
     + <C> Changed your mind?                +
     ++++++++++++++++++++++++++++++++++++++++

OK Brains what'll it be? G

Thank you for your call Simon Potter
on 19-Jul-88

This was your 15 call to GABBS.
This call lasted  5 Minutes.

Press [ENTER] to Disconnect
```

The principal options in GABBS



Apricot goes MCA

New Qi computers aim to catch the corporates

"Qi spells death knell for AT bus" was one of the headlines, but the general feeling among the trade press is that Apricot is taking something of a gamble with the launch of its Qi range (the name was changed from 'Chi' when someone pointed out that Apricot intended to sell the computers in France — where 'chier' is a very crude verb referring to toilet functions).

On the surface, the two Qi machines aren't particularly remarkable by today's standards for upmarket microcomputers — a pair of fast 'personal workstation' computers using 386 processors (the slimmed-down 16MHz 80386-SX with its 16-bit bus for the 300s, the full 20MHz 32-bit 386 in the 600s) with lots of memory, VGA graphics, and so on.

True, there are some novelties — notably an infra-red security card which you can point at the computer to gain access (assuming you haven't lost the card). And there's some added value; the Qi incorporates built-in local network connections that bump up the price of its competitors. Apricot has always gone in for smart styling, too, so the Qi looks better than many 386-based computers. And the pricing is attractive — not bargain-basement, but significantly cheaper than the likes of IBM and Compaq with similar configurations.

All of which makes the Qi an attractive competitor in its marketplace. But the Qi's design also features a major technology decision that sets it apart from other non-IBM computers: it uses the MCA bus.

In practice the bus specification determines how all the electronic components are physically linked — how many wires or circuit lines are used in the standard connections, what signals each of them can carry, how quickly those signals can be moved around.

The standard for IBM-compatibles is usually called the 'AT bus' because it's the specification IBM used on the AT. Technically it is simpler than MCA (Micro Channel Architecture, trademarked and patented by IBM) but it's slower. Everyone who makes 80386-based compatibles is sticking with the AT-style bus for now. Apricot's Xen-i computer family uses the AT bus, and it continues alongside the Qi — Apricot has said it expects future sales to split evenly between AT-bus Xens and its MCA computers, and indeed there's a new Xen about to be launched. Even IBM has backtracked, launching an AT-bus model in the PS/2 line.

As an alternative, Compaq has persuaded several manufacturers to support its EISA proposals for a bus specification to match or beat MCA's performance. Compaq's partners in EISA include some of the big US clone-makers — Tandy, AST, Hewlett-Packard, Epson, NEC, Zenith, and Wyse — plus Olivetti. To those nine can be added several dozen 'supporters' who have paid a few dollars for a copy of the EISA spec.

The selling point for EISA is that it represents a low-risk upgrade of the existing standard without the need to buy new add-in boards — EISA will accept boards and software developed for the AT bus. It will also support more memory, more slots and faster data transfer speeds — faster indeed than MCA, according to the published specification.

The main drawback is that EISA doesn't exist except as a specification, and neither Compaq nor its fellow EISA enthusiasts has (as yet) announced plans to launch an EISA-using computer. There's also the pragmatic view from Apple's John Sculley that "EISA has caused confusion in a business marketplace that doesn't like confusion".

There is much scepticism about EISA's chances. Industry cynics point out that there are eight key IBM patents in the AT bus which will have to incorporated into or emulated by EISA; it's unlikely that IBM will grant any patent rights.

Meanwhile IBM responded to the announcement with disdainful remarks and a few delicate hints about litigation. There was also a clutch of US product launches, including the anticipated return to the AT bus for the entry-level PS/2 Model 30.

All this is red meat to the techno-freak, but for a change the estorica of computer design does have real implications for the end-user. Put crudely, slot-in boards and other add-on functions designed for one bus won't work with the other — the slots are a different size, for a start, and there are more connections in them.

So MCA means no access to the huge range of extras and add-ons that have been developed for the AT bus, but it does permit faster communication between system components and thus accelerates the overall performance of your computer.

But who needs MCA anyway? Both EISA and MCA are aimed at the kind of computing which needs the performance and can justify a premium price — like CADCAM applications and what's becoming known as 'workgroup' computing.

But MCA and EISA performance simply isn't required by most of the devices on offer right now. Tomorrow's devices, however, will be better able to utilise an MCA or EISA bus. That won't matter to the average user running accounts with a spreadsheet and a word processor on a single machine or a small network; provided the system does the job, the bus it uses is irrelevant. But a corporate buyer is always looking for

standardisation and future-proofing to simplify life and provide the best bulk-buy discounts; some very real purchasing decisions about MCA may have to be made in the near future.

Enter the Qi, giving the corporate purchaser an immediate alternative to IBM's PS/2 (the *only* MCA-based alternative at the moment, in fact). The Qi series — pronounce it 'key' — comes in two flavours, the 300 and 600. Between there are six models; all get the classic Apricot styling, low and slim to take up the minimum deskspace. They don't look as good as the Xen, because there's a large platform extruded from the top of the case to provide enough room inside for the MCA cards (other stylists might simply have made the whole case higher and left lots of airspace).

Standard features include space for an optional internal tape streamer, a 1.44MB 3½in floppy disk drive by Sony, from 1MB memory, and hard disks in capacities from 30MB to 120 are available. Around the back you get one serial and one parallel port, socket for mouse, the blanking plates for four MCA cards, and — unusually — a pair of EtherNet connectors plus a synchronous comms port.

A choice of on-board connections for one of the most common local networks is one way in which the Qi is ahead of the game. It's likely that a serious user will at least consider the 10Mbps EtherNet for networking; providing it as standard means the Qi user doesn't have to buy an EtherNet card and take up one of the MCA slots. It also simplifies things, with sockets provided for both the standard 'thick' EtherNet (connects to a transceiver box) and the 'thin' version (sometimes called 'CheaperNet' — uses a special T-piece that plugs into the socket).

The Qi user doesn't have to go with EtherNet, of course — Apricot's own VX offerings are NETBIOS-based, for instance — and the Qi 'environment' pack includes support for several alternatives.

Similarly, the synchronous comms port could be attractive to a purchaser with a relatively complicated set-up. It's obvious use is in communication with mainframes via bisync or X25 links, and the circuitry already provided will do most of the work of an add-in facilities like the IRMA card. It can also be used as a dual async port, however, which means two extra serial devices could be connected to it via a splitter cable. The computer can address then separately even though they share the same plug and socket.

One of the four MCA slots is already taken up by a Western Digital RLL disk controller, about the only piece of electronics that might reasonably have been sited on the main board — but then disk controllers are probably more likely to need replacement than any other functional module, so easy removal is handy (and it gives Apricot the option to switch to a higher-performance SCSI or ESDI controller). Besides, there isn't any room on the system board.

Of the remaining three slots, one has the extra endpiece that takes an enhanced VGA adapter. And given that Apricot has put so many extras into the standard specification, three spare slots is probably plenty — you might add memory, an internal, or a special comms board; but it's unlikely that the typical Qi user will run out of expansion capacity.

Elsewhere in the system unit there's an internal copy-protection device called the 'Dallas Dongles' after its US progenitor; this effectively prevents unauthorised duplication of the Qi 'environment' software.

That environment comprises an extra-cost package (£249) with Windows-386, a Logitech mouse, a comms package (includes file transfer, terminal emulations, and Apricot's 'telephone management'), network support (for Novell NetWare, 3-Com 3+Open, Apricot's own VX-Net, and Unix TCP/IP networks), and extended security features.

Those features can provide a very high degree of control over who gets access to the computer and at what times. It's based largely around the Qi card, about the size and shape of a swollen credit card which contains a small micro and some memory (plus batteries that should be good for at least 18 months). This is preprogrammed with a user's ID (so each user must have a card) and the computer's identification (so the card can't be used on a different machine). Point the card at the infra-red strip on the front of the system unit, press the button, and the data is transmitted.

With the security software in the Qi Environment pack, a system manager can restrict access to the whole machine, to individual devices, and to individual files. Access can be granted only within specific times and/or on specific days of the week. Unauthorised access will be logged, and the system can be told to lock up.

This is pretty impressive stuff. In practice it sounds a bit OTT, though, and the Qi card seems eminently loseable. So is it a technology-for-its-own-sake gimmick? Computer vendors take the view that it never hurts to give the salespeople something extra to say to the prospective customer, something that differentiates one machine from another. But the Qi card and the associated security are clever, do work, would probably deter any but the most ardent criminal, do allow sensible operating procedures to be impose, and don't necessarily have to be used.

Pricing ranges from £1,999 for a single-floppy Qi-310 (no hard disk) to the Qi-660i with 120MB hard disk at £4,999.

"Qi is Apricot's premium product, designed primarily for the corporate and large-company user," says Apricot. The company says its own market research indicates that such users are "intent on incorporating MCA in their strategies" while the AT architecture remains the norm for smaller user. So the Qi enables Apricot to hedge its bets for a relatively modest investment in R&D (£5 million, we heard). Apricot is shipping an MCA machine for corporate buyers who want it, and it's a good-looking machine at that. The existing Xen-i product line is among the most competitive 286/386-based offerings with the AT bus. And Apricot has even paid its $2,500 for a copy of the EISA spec — just in case. ◆

Glum MBS interims

MBS has reported a set of half-year figures that are even worse than most expected — sales down £3 million to **£61.6 million**, profits halved to **£600,000**. Poor performance in the IBM PC distribution business was cited, with several offices now to be closed and some 85 of its 900 staff given notice — including director Tim McGuinn, one of its prize captures from IBM. MBS's non-PC business has been doing well, however, and our view is still that the shares are worth holding — provided the immediate storms can be weathered.

Tandata vs Tandata

Comms specialist Tandata sent us a press release to point out that two of its directors are *not* facing corruption charges. They belong to a building firm also called Tandata. Ok, let's be charitable ...

Mini tape backup

Gigatape Ltd is a new UK subsidiary of a German company called Gigatape GmbH with a product called **Giga 1200**. You think they're trying to tell us something? You're right — Gigatape is a jolly impressive tape backup system utilising small cassettes (about the size and shape of dictation cassettes) with digital audio tape (DAT) encoding. The result is 1200MB of data on a 60-metre tape which can fully rewind in just 20 seconds. At prices around £6,000 for a complete add-on system, though, we're talking big users with big computers: at the press launch, Gigatape pointed out that its target audience currently uses ¼in 3M-type tape cartridges with capacities around 40MB.

Dull PC Show

The PCW Show used to be held at Olympia, and in the last couple of years it hadn't done badly with a potential awkward mix of customers — kids wanting games in the noisy section, a smaller but calmer business-oriented hall alongside. For 1988, the show organiser Montbuild changed the name and the venue; the **Personal Computer Show** moved to the scruffy and colourless Earls Court — and it lost just about all its interest en route.

There were few brand new products on show, and most of the big names stayed away. Instead the business section featured a clutch of European vendors (Dutch in particular, for some reason) and Asians (Singapore, Taiwan and Hong Kong each had their own little sections). But while the show may have been dull for the visitor, by all accounts business wasn't bad for the mainstreamers like Victor — Montbuild claimed attendance of 72,000, some 40 per cent higher than last year. Frankly it didn't feel like that many people.

The most active stand — apart from Amstrad's with its middling-interesting 286 and 386 IBM-compatibles — had Walters offering discounts on just about everything, including the cheapest 386 we've seen yet: £1,260 for a basic 16MHz system with keyboard and monochrome monitor but no hard disk (allow another £300 or so for that).

As usual with computer shows these days, there were a host of prizes — these called 'the British Personal Computer Awards'. Business PC of the Year went to IBM's PS/2 Model 70-A21, which is arguably neither the most exciting 386 machine around nor even a PC. The *PCW* Home/Small Business Computer trophy went to Dell's 220; curiously, the runner-up in the category was Cambridge Computer's Z88. How do you compare a laptop and a 286-based desktop machine?

Other awards were less contentious — Microsoft's fancy Excel spreadsheet, Zortech C, SageSoft's cheap 'n' simple MainLAN network, and the HP DeskJet — an inkjet printer than emulates a laser printer and isn't far off the same print quality.

Telex time

A **British Telecom** survey of 200 firms in import/export and freight forwarding suggests that half regard **Telex** as "essential" for business in general and over 90 per cent rate it as vital to their own operations. Almost two-thirds reckoned Telex had no disadvantages at all, though, which probably says something about British exporters!

Wear your Apricot

The true Apricot enthusiast will already have called **Ocean World** in Cowes for a copy of the Apricot designer-wear leaflet — T-shirts, sweatshirts, jogging pants, gilets, all emblazoned tastefully with the Apricot logo. And to provide the finishing touch, how about belts, neck towels (sic), holdalls ... and of course the Apricot wallet, the essential finishing touch. It's all part of the promotion for the Apricot-sponsored *Spirit of Apricot* ocean racer; and actually the gear looks quite nice. Some of the stuff is silly (the Apricot key fob?) but it looks well-made and prices aren't bad. Ocean World is on 0983 291744.

Bad lines

The quality of **data transmission lines** in Europe is pretty poor, according to the 1988 Annual Survey from the **European Association of Information Services**. Overall, 25 per cent of calls failed; France is the worst place in Europe from which to send data, with a third of calls failing. Britain's failure rate was at 28 per cent, though this might reflect a disproportionately large number of responses to the survey.

Poor MS-Pascal

"Fairly disgraceful" was the conclusion of the **British Standard Institution** on **Microsoft** MS-PASCAL. BSI's software certification people tested five Pascal compilers for conformity to ISO 7185, the only official Pascal standard there is. "Few in their right minds would use Microsoft's Pascal compiler to develop applications" said the man from BSI, who also criticised TURBO PASCAL as "restricted" despite Borland's claims that it is 'standard' Pascal. PRO-PASCAL by **Prospero Software** came top of the evaluation.

Watching the detectives

According to the on-line industry newsletter *Newsbytes*, a private investigator in Phoenix has set up an electronic mail service for PIs. Some 150 customers have already paid their $150pa for a mailbox, a Telex number, and a weekly on-line newsletter. ◆