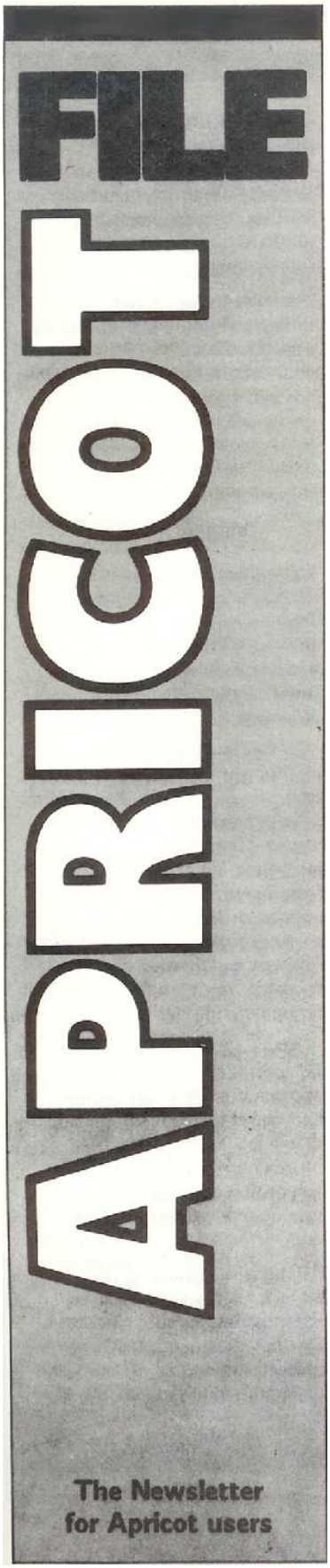
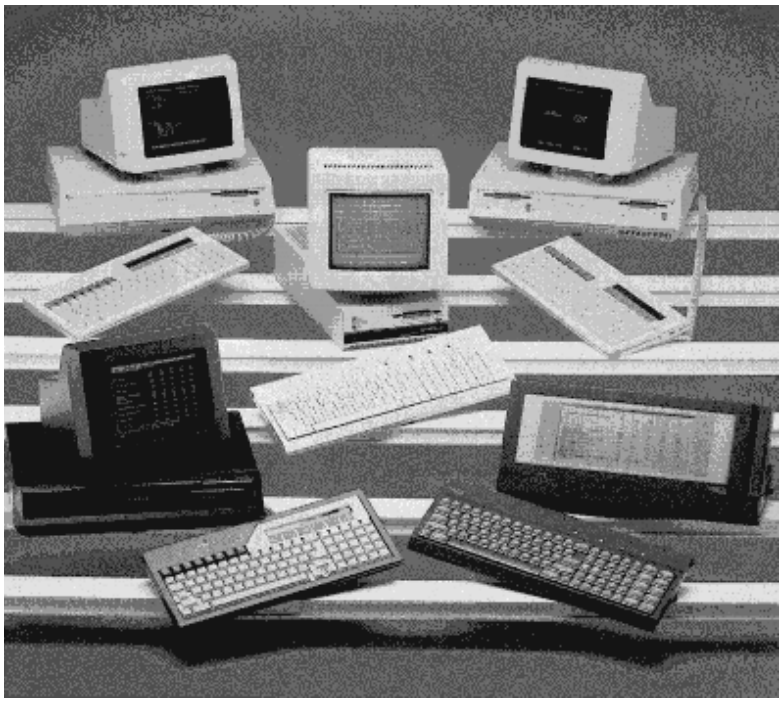


Apricot File 3.4



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 3.4

APRICOT FILE

The Newsletter for Apricot users

January 1988

The State of the Art?

The 1988 Which Computer Show reviewed (and survived)

by Garret Keogh and Simon Potter

We've seen the future of computers at the Which Computer Show ... and nothing much has changed. The 1988 show was, to be perfectly honest, rather dull.

Except of course for APRICOT FILE, which had its standard space on the Islington Council stand. We had the usual mixture of old friends, enthusiasts and boffins visiting us; and we also met some examples of that rare animal, the Apricot user who was not a subscriber to APRICOT FILE. Star visitors included David Langford, with a brilliant new version of SQUIRE.COM (insists on putting '*, squire*' at the end of every sentence — can now be user-customised) and the first working implementation we'd seen of Ansible's APE.COM Apricot emulator (run your favourite Apricot software on an IBM-compatible).

But having escaped from the time warp that is the NEC in Birmingham, there's a moment or two to reflect on the show — to see who was there (more than 400 companies) and who wasn't, to figure out why they were there or not, to assess what interest the Show had for us, for the visitors, and for the computer business in general.

Impressed with Apricot

First, who was there? The **Apricot** stand was one of the most stylish, biggest and busiest; it also had one of the two prime positions — next door was **Compaq**, which matched it for size but outpaced Apricot in sheer vulgarity with a gloomy dark-grey fake-Roman façade that recalled nothing so much as Christians being thrown to the lions.

So back to the spacious, clean Apricot acres. Where we witnessed the spectacle of **Roger Foster**, Top Dog at Apricot, and industry minister **Lord Young** hob-nobbing with smiles for the flashing cameras. Lord Young was there to open the show by telling us all how wonderfully well British industry is doing (and the computer industry in particular, of course). He sang the praises of British-made computers and then admitted to using a Toshiba. 'Nuff said.

Perhaps the most interesting thing for old-style Apricot users on the Apricot stand was the presence (somewhat muted, but definitely there) of the PC-286 upgrade for old-style machines. Apricot at last seems to be quite serious about this promotion, now known as the 'PC-286 Advance plan'. Some small but glossy literature has been produced and an upgraded PC appeared on the stand among the 386 machines and the high-tech VX9000 series.

CONTINUED ON PAGE TWO

In this issue

Using the new Apricots 5
Inside MS-DOS — Configuring CONFIG.SYS 6
Notebook: Microsoft Mouse and DOODLE 9
Keyboards and Fonts — why bother? 10
FORMAT funnies on the Apricot 14
Review: Fortran update — Prospero's Pro-Fortran 16
The Q&A Clinic 18
The Interrupt and how to use it 20
Batch File Recipes — three for Super macros 24

Which Computer Show report

CONTINUED FROM PAGE ONE

The PC-286 appears as reviewed in APRICOT FILE Release 2.12. Apricot will carry out the upgrade using "fully trained Apricot Computer Maintenance engineers " who will carry out the conversion on-site. Apricot's unit had Windows running with the Apricot trackerball mouse on a colour screen, and very impressive it looked too.

Big deals

Apricot also launched its parallel processor at the show. The VX9000 series derives from the SYMMETRY range produced by **Sequent Computer Systems**, a moderately-sized name in the world of parallel processing technology. The systems have front panels reminiscent of the Starship *Enterprise*, run a version of Unix, and use closely-coupled 80386 processors.

If all that sounds a bit much, wait till you hear the prices: the entry level system starts at £40,000, which is actually extremely competitive in this high-level market.

The deal with Sequent obviously extends the Apricot product line into 'departmental' computing in ways that probably aren't amenable to a LAN solution. Two related markets in which Apricot has some penetration and a good reputation are health and local government; with a minicomputer-style product to sell against the likes of the DEC Vax, Apricot should do well. Another prime target will be the customers of Apricot Financial Systems, which currently sells large Vax-based financial management systems to the City; the division's two key products, Quasar and CityDesk, are both being ported to the Sequent.

Apricot will be taking Sequent's basic processor hardware — the systems Sequent itself sells under the name Symmetry — and repackaging it at Glenrothes with peripherals and enhancements. The full Sequent product line covers a huge range, with top-end models capable of servicing a thousand on-line terminals quite happily; the low-end system uses a couple of linked 80386 processors (Sequent uses multiples of 386s to get extra power) with a minimum of 8MB memory on a backplane designing for simple expansion or reconfiguration (the quick 'n' easy slot-in expansion boards of the PC world are a constant source of surprise to people in the minicomputer business).

Implications

For Apricot there are a number of strategic advantages in this move. For one, the Sequent approach feels like the right one in terms of multi-user technologies — the best available hardware components, the maximum protection for users, the sane way to handle large on-line systems with maximum flexibility.

And Apricot has a ready market — both in terms of existing users, many of whom have had to shop elsewhere for their more substantial systems, and because there is still a prejudice in favour of buying British where possible (and where pricing permits).

A third plus: the Sequent approach fits the current product lines — no direct competition with existing products, a natural Vax replacement for the financial systems, a natural upgrade path into 386-based Unix-driven multi-user environments. The Sequent utilises the same operating system and the same processor families as Apricot's other products; the choice for Apricot's users (and the options for the salespeople) are increased by this move, but at the same time Apricot can offer the user an applications environment that is basically the same — from an entry-level 286-based £2,000 desktop computer to a 30-processor VX9000 Sequent running several hundred VDUs. What's more, Apricot doesn't carry any commitment to protect any current development work or products: many companies simply couldn't afford to switch to something like Sequent because of the time and resources they have invested in alternative approaches.

In short, the Sequent deal extends Apricot's chances of selling into the corporate market. The only potential difficulty will be whether Apricot can actually realise the potential. Can it sign up the right VARs? Can it establish a successful direct sales force in more markets than the City? Can it provide the kind of support that the whole business of selling, designing, installing, and servicing a complex system necessarily entails?

Our view is that Apricot probably can handle the Sequent business. It has done pretty well with its move upmarket from the PC/Xi and Gateway days; some extensive and complicated Apricot networks are up and running; there have been no notable defections among blue-chip customers and several useful acquisitions. What's more, the largely unsung Apricot Financial Systems has demonstrated the existence of a resevoir of competence and sales talent within the group. And Apricot does have the cash to invest in an expanded tech support organisation.

Apricot DTP

Apricot also showed some interesting new desktop publishing products — and the demos were among the most impressive systems on view in a show that was a big disappointment on the DTP front. Apricot are retailing a complete PAGEMAKER-based system including the PS8/2 laser printer 2 and a Xen-i 286 at £5,499 ex VAT: not bad for a ready-to-go POSTSCRIPT package.

The PS8/2 is in fact the first rebadged Apple LaserWriter 2, probably the most impressive laser printer on the market. It features 2MB RAM, a megabyte of ROM, and 35 built-in typefaces in 11 font families. There's a go-faster option, named PS8/2 Plus, which has the Motorola 68020 (the standard PS8/2 gets the 68000 chip). The RAM on this machine can be extended to a breathtaking 12MB — remember, it's only a *printer*, for heaven's sake — and you can attach up to seven add-on hard disks. The PS8/2 is £3,295, the Plus is £4,495.

Interestingly, Apricot has also signed up for Ventura Publisher — which makes it at present the only supplier to offer both the market leaders in DTP software. Apricot's recently-appointed DTP overlord, **Keith Henley**, told us there was a clear distinction in applications between multi-page documents probably of a technical nature (like books and manuals) and more design-based jobs like fancy newsletters; Ventura is good for the former, PAGEMAKER for the latter.

It hurts a bit to say anything too nice about Apricot. But the Which Computer Show stand at the show was exemplary — without a doubt the best-designed, well organised, and, in terms of product, most innovative. 1988 looks like being a good year for Apricot; for a company that came from near ruin to this high position in the industry, it deserves all the praise they will undoubtedly get. *(That's enough kudos for this year — Ed.)* If you don't know by now, Apricot is on **021-456 1234**.

Takeaway Amstrads

Elsewhere, the **Amstrad** stand was as busy as usual. One of the key products on the stand was a £399 24-pin matrix printer called the **LQ 5000di**; it bangs out draft copy at 288cps, NLQ at a speedy 96cps.

But Amstrad's star this year was the new PPC portable. It won't be to everyone's taste, but technically it looks good; and it does have some important design features that make it a fair computer — and a good data comms computer. It uses an 8086, Amstrad's proven IBM-emulating BIOS, an LCD display with a correct aspect ratio (so circles are round rather than ovoid), and a full-size ATE-type keyboard with 102 typewriter-like keys. Two 720KB 3½in disk drives are provided; so are a parallel connector, one full-size 25-pin RS-232 socket, and a pair of monitor connectors (one for a standard RGB display, one that can take the monitor from an Amstrad PC).

The only expansion provided is for a modem; you can save a few quid and buy the modem-free 512KB version for just under £400, but the 640K model with twin floppies and built-in modem looks to be the bargain — the modem itself has a spec for which some modem manufacturers are asking nearly the PPC's price. The modem is Hayes-compatible; it can auto-dial and auto-answer; it will do V21, V22, V22 *bis* and V23 — speeds from 300 to 2400bps (and for a bit of world-compatibility, there are also Bell speeds). To drive it, Amstrad has packaged a version of Softklone's MIRROR II. And the price for that seems pretty good to us — **£599**.

But on the portable front generally, it was clear that the excitement has died down. Portables were few and far between, except on the stands of the majors in this market — **Compaq** and (especially) **Toshiba**. The revitalised **Victor Technologies** did have an attractive little Toshiba-like 286-based number nestling among its good-looking small-footprint 286 machines; and its V286P portable is priced quite competitively at £2,499 for a box with a 30MB disk. But then you can get a non-portable desktop 286 computer for a grand less, no-one on the Victor stand knew very much about the V286P ("what's that socket for?" "Er ..."), and certainly none of the salesmen had any idea who might actually want to *buy* it.

Anyhow, Victor is on **0494 461600**. The only other portable interest came from the perennial Sir Clive Sinclair and the **Z88** — which finally seems to be just about ready for sale, given that the the thing actually has some usable software now. Its manufacturer **Cambridge Computer** discreetly showed a portable Hayes-compatible modem the size of a fag packet, switchable between Bell and CCITT standards for comms at up to 1200bps; BABT approval Real Soon Now (like March, maybe), projected RRP **£149**. And there was some file transfer and general comms software from Cambridge Computer and third parties.

On the subject of comms, it was rather a depressing show — nothing very much to shout about (or even to look at). True, **Amstrad** did its bit to drop the price of V22 modems — several Amstrad prices have drifted downwards, with its existing Pace-built V23 modem down £50 to **£99**. But the only other modem-makers there were showing high-performance high-priced units — **Miracle Technology**, for instance, was majoring on a new line of V29 and V33 modems (9600bps and up) at prices from £985.

Nothing special in local network offerings, either. Market leader **Novell** had little to say — a couple of products for Apple Macs, and that was about it. **Torus** has a new version of its **Tapestry** LAN, but no radical advances. **British Telecom** added an X400 messaging application to its Novell-based **T-1000** LAN, but the only interesting feature was that it runs with a dedicated PC hosting the the X400 gateway rather than a minicomputer.

Telex options

As for services, the best news we found was **Mercury Communications'** demonstration of the **Mercury 7200** Telex service. It's not a direct connection to Mercury lines; you're routed through a standard BT connection to Mercury's network. So Mercury expects to sell the service on quick and easy changeover (no need to alter existing equipment or Telex numbers) — as well as cost savings of around 20 per cent for long-distance inland Telexes (and 10 per cent on direct-dialled international calls). As on all Mercury services, itemised billing is also standard.

For 7200 you pay only a modest £5 per month registration fee for each answerback. On top of that, of course, you still pay BT for installation and rental of a Telex line. So in practice, 7200 appeals to the smaller user because of the per-call savings.

BT has a two-tier rate — 3.5p per minute for calls of up to 35 miles, 10.5p for longer distances. Mercury charges a flat 8.5p per minute; since most Telex messages will go further than 35 miles, Mercury will almost always be cheaper. And Mercury offers a store-and-forward facility at no charge; your message is downloaded to Mercury, which handles all the dialling and retries automatically for you. This keeps your line usage to a minimum. BT has a similar facility now in the shape of Telex Plus, but imposes a surcharge for its use.

Details from Mercury on **01-528 2000**. The company also announced a couple more database gateways on the **Mercury Link 7500** e-mail service — Infocheck for company data, OAG for airline schedules — and the option of delivery to fax machines (send your text direct to anyone with a Group 3 fax).

Wind-down

The Show included a large and product-packed **Sentinel Software** stand — featuring a database package named DataPerfect, something called WordPerfect Office (local electronic mail facilities with desktop utilities like diary and calculator), and pre-release copies of **WordPerfect 5.0** — looks very neat, especially in its smooth integration of text and graphics. All for IBM-compatibles, of course; no-one on the stand wanted to know about the Apricot user. If you want to remark on this, Sentinel is at **0932 231164**.

Reflections

We could tell you that the **First Software** stand was shaped like a giant desk and that visitors sat in one of the desk drawers. We could tell you that the **Unisys** stand had someone dressed as a giant Popeye giving out leaflets. We could tell you about the models and the flash advertising, the free drinks and the glossy promos.

But underneath the tack and glitz there was really very little substance at all to Which Computer 88. Maybe the computer industry has reached a low point: companies are announcing increased profits, but in terms of innovation and originality nothing much is happening.

The big companies seem to be treading water, marking time, waiting for OS/2 and/or PS/2 and/or MCA. No-one seems to be sure what the next big thing will be; in the meantime, profits are to be made selling 286 and 386 machines into basically the same applications as Apricot found for the Xi and PC.

Sure, the quality of the software for these machines is being tightened up. The prices are coming down. Documentation, for a long time a major sticking point with most software, is getting better. Perhaps the overcrowded market means vendors are at last having to take notice of what the user actually wants and needs.

But if you were shopping for systems to last the next three or four years, you would have been disappointed. What we have been told were to become the new industry standards weren't reflected at all — except perhaps by default, in the absence of key names like Microsoft, Lotus and (crucially) IBM. IBM now has its own IBM Show (late Autumn) and Lotus too has a dedicated show (in March). But there were very few OS/2 and PS/2 add-on products on view; it looks as though the waiting game is still being played. Nobody wants to jump in the water too quickly.

The absence of Apple and Mac-related products suggests that the Macintosh is not yet accepted as a serious general-purpose business machine. It appears to be consigned to the single-role of desktop publishing, despite all the Mac-inspired enthusiasm on the other side of the Atlantic.

Specific areas of the market like DTP and CAD are showing some innovation — and price decreases too, both of which are good for the end user. But the specific 'villages' for these subjects at the Which Computer show just showed the same faces with (largely) the same products.

As a business user visiting the show perhaps wishing to buy a computer system, you would think what you needed was a 286 (or just possibly a 386 machine) with a 30MB hard disk and perhaps a small, simple resource-sharing network. It would be running accounts or spreadsheets or word processing, using a decent 24-pin matrix printer. You'd get a good deal on such a machine, and for most users that setup is enough.

But does it need a 'Which Computer' Show? How many clones can you assess at once? How many dot matrix printers can a man take? This computer show didn't leave us asking *'which'* so much as *'why'*. ◆

Notebook

The new Apricots

A user's view on PC/S

By Eddie Bromhead

About six weeks ago we received eight PC/S 286 Apricots in my Poly Department for CAD work. These are Apricot's latest IBM-compatibles with 30MB disk, on-board LAN support as standard, 1.44MB 3½in floppy, the no-MicroScreen keyboard, and the Apricot keyboard mouse which now plugs into the back of the system unit. Two machines came with EGA, the rest with green-screen Hercules video drivers; half of them had Apricot's 5¼in add-on drive.

First impressions (as they say, you never get a second chance to make a first impression) were that the keyboard had got back the old PC's feel, which I prefer to the Xen's; but the machines had a lot of spaghetti out of the back — the monitors have separate power inputs. Three of the machines wouldn't boot or wouldn't run; their problems related to the expansion cage (IBM cards of the half-length variety lie flat in the slimline case which now has an internal power supply) or with the floppy drive. Other problems were experienced with the system configuration information which is held in CMOS memory even with the power off. Subsequently, one machine crashed badly after a CTRL-BREAK and now won't boot. Jeez!

The add-on 5¼in drives handle 360KB or 1.2MB format, but suffer if disks formatted on a high-capacity drive are written to on a low-capacity drive — the sort of complaint PC/AT owners were making two years ago.

Unpacking

When you get the machine, it's highly likely that your first job is to get the back off. Why?

(a) to set the monitor scan rate to match your monitor — it is different for green or white monitors
(b) to set clock speeds for the processor and co-processor
(c) to disable the on-board Hercules video if you have an EGA card
(d) to set the LAN address if you are going down that lunatic path.

All this is helped by having the instruction page from the manual stuck on top of the power supply (which these days is internal) — but upside down! Get any of these switches wrong, and it's unlikely that a machine will boot; or if it does, it will possibly not drive the screen properly.

Once over those hurdles you will certainly need to set the EGA card's DIP switches, focus the monitor, adjust picture size, call Apricot Maintenance twice, and run the setup program.

Using the machine

System operations like disk copying, formatting and the like seem slower than the old-style Xen. The monochrome display is very similar to the older machine's, except the fonts are a little unfamiliar. EGA is decidedly slower still; and most software switches from mode to mode, making the Japanese-built EGA monitor resound with disconcerting clunks. The EGA is much poorer in black-and-white monochrome; but in full colour it is extremely attractive, and can be switched to green-on-black with a button on the front.

Development software which crashes the machine sometimes needs a complete switch off to restore normality. There is a reset button as well as CTRL-ALT-DEL, but it isn't so effective as you might expect. Switching off isn't a good idea, since the Winchester needs to be settled down by running the head-parking program first.

The system comes with MS-DOS 3.2, Windows 1.03, a disk to boot the system as a LAN workstation, GW-BASIC and not a lot else. For the real Apricot devotee, there's no Super software, no alternate fonts and keyboards (or FONT, KEYS, FONTEDIT, KEYEDIT or LOGOEDIT), no GEM, ACTIVITY, or MANAGER, no GSX, no MicroScreen, and no dumbo demonstration programs of what MS-BASIC can do.

On the other hand, the machine runs IN*A*VISION graphics (under the version of Windows supplied), Ventura Publisher and GEM DRAW+ under GEM 2.2, Prodesign 2 and AUTOCAD — quite impeccably, and on the EGA in full-colour glory.

The one failure has been GSX — MS-BASIC and GSXBAS run fine on an EGA monitor with the CGA driver in either 320x200 four-colour mode or 640x200 black-and-white, but the Hercules driver produces junk.

Apricot-format single- and double-sided disks read fine, although the FORMAT program won't format disks to the old Apricot spec. Formatting to the IBM/Microsoft 720KB style is slow, the default is the full 1.44MB format.

On boot-up, the INT EC text driver (effectively ANSI.SYS with enhancements) and the appropriate INT EC graphics driver (like 800400EG.EXE) are loaded. Running text and graphics through interrupt EC appears to be a BIOS 3 specialty, and might even give some backwards compatibility with the older machines — but who's writing Apricot-specific stuff these days?

Conclusions

Nice if you have to have IBM-compatibility, but nowhere near as nice as a real Xen. ◆

Inside MS-DOS

Configuring CONFIG.SYS

By Garret Keogh

Everyone has their computer set up exactly as they themselves like it. Users with hard disks have their own subdirectory tree structure and peripherals attached and configured as preferred. CONFIG.SYS allows you complete control over the configuration of your computer.

When you boot up any 8086-based machines, a small piece of microcode in ROM is executed; this reads the 'disk bootstrap' from a disk into memory and transfers control to it. The disk bootstrap checks to see whether MS-DOS is present — whether the IO.SYS and MSDOS.SYS files are on the disk. If they aren't there, you'll be prompted to try again. Otherwise they're read in memory and control is passed to the entry point of IO.SYS — which contains the BIOS (built by Apricot) and SYSINIT (by Microsoft).

SYSINIT is called by the BIOS, determines the amount of memory available, relocates itself into the top of memory, moves MSDOS.SYS from its initial load location into its final position (which will usually mean overwriting the original copy of SYSINIT), and calls the initialisation code in MSDOS.SYS. That does a number of things in quick sequence:

- it initialises work areas and tables
- sets up interrupt vectors 20H to 2FH
- calls the resident device drivers one by one (which in turn will perform any hardware initialisation required)
- sets up the disk drive parameters
- displays the MS-DOS copyright message
- ... and then passes control back to SYSINIT.

SYSINIT then looks for a CONFIG.SYS file. If one is found, it is loaded into memory and processed a line at a time.

CONFIG.SYS is the 'system configuration' file, and the one on my computer looks like this:

```
FILES    = 20
BUFFERS  = 20
DEVICE   = C:\SYS\MOUSE.SYS
DEVICE   = C:\SYS\INTECTXT.SYS
DEVICE   = C:\SYS\MSCALC.SYS
DEVICE   = C:\SYS\AMDRIVE.SYS
DEVICE   = C:\SYS\ATMODEM.SYS
COUNTRY= 44
SHELL    = C:\COMMAND.COM C:\ /P/E:1024
```

(My machine happens to be a Xen-i, which is why the file specifications in CONFIG all refer to drive C. But the principle of CONFIG.SYS is the same for all MS-DOS computers, including old-style Apricots.)

After CONFIG.SYS has been executed, SYSINIT reopens the system devices for standard I/O. Then it calls the MS-DOS EXEC function to load and run COMMAND.COM — which either searches for and runs AUTOEXEC.BAT, or displays a prompt for your input. MS-DOS is now ready for use and SYSINIT can be discarded by the system.

CONFIG allows the user to customise the MS-DOS environment. The various statements in the CONFIG.SYS file effect certain aspects of MS-DOS. And the useful thing is that CONFIG.SYS is a plain text file, which you can easily edit with EDLIN or a word processor that produces ASCII files (like SuperWriter using System format); so it affords you the opportunity of tailoring those aspects of MS-DOS to suit your own personal needs or quirks. Next time you boot up, and thereafter, those personal preferences will automatically be included in the boot-up procedure.

Let's look at what the various configuring commands mean.

Files

Not all the possible CONFIG commands need to be included in a CONFIG.SYS file, but this one should. It defines the number of 'file handles' that can be open at any one time. A file handle is a two-byte number which the system assigns to the file or device that a program uses. The number can be from 1 to 255 for MS-DOS 3, and 5 to 255 for MS-DOS 2. The system sets the default at eight, including five that are predefined for the standard I/O 'files' ...

- the console
- standard input
- standard output
- auxiliary output
- printer output

Any one program can have up to 20 file handles open at once, and in a multitasking environment *each* process can have 20 handles. In such cases it is clearly useful to set the number of handles larger than 20, the exact setting depending on the number of processes and the requirements of each process. It must be born in mind, though, that each file handle over the default requires an extra 48 bytes of memory.

Buffers

This is another command that should probably be in every CONFIG.SYS. The BUFFERS command sets the number of disk buffers to be allocated. That number can be anything between 1 and 99; the default varies with machines and configurations, but it might be anything from 2 to 10.

This is one of the most important lines in the CONFIG.SYS file, because the configuration of the buffers command can alter your disk speed and performance quite significantly. Why?

Well, a buffer is a small area of the memory that MS-DOS specifically sets aside for use as a sort of holding area for information on its way to or from your disk. When MS-DOS first searches the disk for requested data to put into the program, it is loaded and stored in a buffer area and sent to the application program as and when needed. When more data is required by the program, the system will check the buffers *first* — if the desired information is there, MS-DOS won't have to check the disk at all and the loading into memory for processing is much faster. If the information isn't in a buffer, MS-DOS will empty the least-used buffer and refill it with new information from the disk.

The more disk buffers the system has, the more information it can hold at any one time without having to access the disk. There is, of course, the inevitable penalty: each additional buffer above the default requires 528 bytes of memory.

So how many buffers should you specify? The default includes three disk buffers if you're using a hard disk machine, two for systems with floppy disk. Thereafter, the number of buffers you add will depend on the amount of memory you have and the type of programs you run.

On the whole, you'll need to keep the buffer number low if you're short of memory. And if your machine is used mainly for applications like a spreadsheet or word processing, where disk accesses aren't so frequent, a modest buffer configuration — say 15 or so — will ensure that the maximum memory is available for your applications.

But if you're running programs that require frequent random access to the disk, such as a database, a higher number of buffers will increase disk performance. It takes a little experimentation to get the balance right between the number of buffers that will speed up disk access and the amount of memory you need to run your application properly.

Device

The DEVICE command installs a file called a 'device driver', a software facility that enables the system to utilise the specific characteristics of a device.

When MS-DOS is loaded it contains its own device drivers for the disk drives, monitor and keyboard, and serial and parallel ports — all devices that MS-DOS assumes you will have. For anything else you may well have to specify a driver and where MS-DOS can find it. Device drivers come as patches of code with a .SYS suffix; so to use a particular device, the appropriate file is stipulated in the 'filespec' part of this line.

In short, the DEVICE command allows addition device commands to be added. For instance I have added drivers for a mouse, a calculator, and a modem, along with a couple of others. You can specify a path for the device driver if it is not in the root directory — you can see from my CONFIG file that my device drivers are stored in a subdirectory called SYS (to keep the root directory as tidy as possible).

When your system boots up and MS-DOS looks at CONFIG.SYS, it will automatically invoke any necessary software and prepare itself to handle the device you've told it about.

Don't forget that the computer is inherently stupid — unless you load a mouse driver before attempting to use your mouse, the computer simply will not recognise the mouse and won't know what to do with it. You have to explain.

ANSI.SYS is a device driver you may have heard of; it appears on IBM machines, and provides capability to handle the display and keyboard. It's not needed on the Apricot because this is handled in the BIOS. Apricot-specific device drivers include the Portable's SPEECH.SYS, which installs the speech recognition facility (should you be determined to use it). To use Apricot's mouse you'll need MOUSE.SYS in CONFIG.SYS, the internal modem requires MODEM.SYS. And other extras may also require one or more device drivers to be installed.

Country

COUNTRY rather cleverly uses the international dialling code to customise the style of the date, time and punctuation used by MS-DOS so that it is appropriate to any of the countries listed here:

UNITED STATES	01
NETHERLANDS	31
BELGIUM	32
FRANCE	33
SPAIN	34
ITALY	38
SWITZERLAND	41
GREAT BRITAIN	44
DENMARK	45
SWEDEN	46
NORWAY	47
GERMANY	49
AUSTRALIA	61
FINLAND	358
ISRAEL	972

It can be entertaining to try different possibilities. This is the UK COUNTRY format:

```
Date:                dd-mm-yy
Time:                hh:mm:ss
Currency symbol:     £ to the left of the amount
Thousands separator: comma
Decimal separator:   full stop
Data list separator: comma
```

The default on many MS-DOS machines, especially IBM-compatibles, is the US setting:

```
Date:                mm-dd-yy
Time:                hh:mm:ss
Currency symbol:     $ to the left of the amount
Thousands separator: comma
Decimal separator:   full stop
Data list separator: comma
```

The French format gives a date style that many prefer, though it also has other effects:

```
Date:                mm/dd/yy
Time:                hh:mm:ss
Currency symbol:     $ to the left of the amount
Thousands separator: full stop
Decimal separator:   comma
Data list separator: full stop
```

The date and time formats are probably the only ones that you'll notice in practice — the other settings are typically reconfigured when a particular program is loaded to suit its own style. The COUNTRY command doesn't have any effect on the amount of memory MS-DOS uses.

Shell

SHELL tells MS-DOS where to find the command processor, which in most cases will be COMMAND.COM.

The phrasing of this command is somewhat complicated; and, to boot, MS-DOS is particularly fussy about the spacing and punctuation. So be careful when using it.

The form of the SHELL command has changed a lot through the various versions of MS-DOS. This is the syntax for MS-DOS 3.1 and 3.2:

SHELL = PATHNAME PATH [E: nnnnn]/P

The format for MS-DOS 2.11 and 3.0 is ...

SHELL = d:PATH\filename.ext d:path\ /P

The main difference is the absence of that */E:nnnnn*. But let's look at the possible parts of the SHELL command:

- *Pathname* is the drive, path and file name containing COMMAND.COM
- *Path* is the drive and path of the directory containing COMMAND.COM
- */E:nnnnn* is available with MS-DOS 3.1 and above. It specifies the environment size, *nnnnn* being a number between 160 and 32768 (the default is 160). The 'environment' is the area of the memory where you can store information to be read later by your program; this information is placed there by the SET command — see Roger's *Inside MS-DOS* Workshop in APRICOT FILE Release 2.9 for more on the environment. You would have to increase environment size if you regularly get an 'out of environment space' message
- */P* is a switch which indicates whether COMMAND.COM is to be used on booting. If this is not specified, the AUTOEXEC.BAT will not be run and a system crash is the likely result.

For those lucky enough to have the later MS-DOS versions it's worth talking about. Increasing the size of the environment increases MS-DOS's size. In MS-DOS 3.1 this is specified as the number of 16-byte sections of RAM reserved specifically for the environment; contrary to the remarks of another contributor to APRICOT FILE (Roger Gann's flawed but otherwise brilliant article) the environment-size parameter of the SHELL command *does* work, and it can make a substantial improvement to memory-handling.

Last words

The two other CONFIG commands are of limited use. The BREAK command is a simple on/off toggle switch which allows a program or command to be interrupted. The default is BREAK OFF.

The LASTDRIVE command comes with MS-DOS 3.0 and specifies the maximum number of drives on your system. The syntax is LASTDRIVE = *x*, where *x* can be any letter from A to Z. This is mainly used when adding external disk drives, tape backup units or in a network where another computer is used as an external drive.

As you can see, the CONFIG.SYS file can tailor DOS to your specific needs; and through the use of commands and device drivers, it allows you to customise many of the operational aspects of your computer. It's worth spending some time experimenting to get things right. Just edit CONFIG.SYS, using a line editor such as EDLIN or your word processor, and store it as an ASCII file. Remember that any changes *only* come into effect when the system is rebooted. Happy configuring! ◆

Notebook

Microsoft Mouse and DOODLE

Installation notes for an Xi

By Lindsay Doyle

If you have the CAD program DOODLE on your Apricot and want to install a mouse, its supplier Trilex will recommend a serial rodent supplied by Microsoft. (Apricot plc: please note that Trilex does not recommend the Apricot mouse and ask yourselves why.)

Why should you buy a mouse for DOODLE, even if you don't have any other use for the rodent? I'll tell you. The DOODLE keyboard cursor has the unfortunate characteristic that its velocity is not constant but increases logarithmically with elapsed time as you hold a button down. This must have initially seemed like a good idea to someone, but it is a non-starter. It constantly leads to overshooting the spot you want; and there is no way to train yourself to compensate, as the length of the next jump depends on how many jumps the cursor has taken previously. Also, it has to be said that the keyboard cursor is ill-suited to the freehand mode, where your every move is copied by the trace on the screen. Try drawing a map of the British Isles in this mode: not a chance. But with the mouse, a piece of cake.

So what must be done to install the mouse? Gather round while I tell you.

The box you receive will probably say 'for IBM Personal Computers and 100 per cent compatibles'. You know that the Apricot is not, especially when it comes to moving things around the screen. Fear not! All is well enough.

Open the box and you will find seven books and a License Agreement folder containing six disks. Discard or give to an IBM-owning friend all this material except the booklet labelled 'Customer Service Plan'. Next, examine the mouse cable and you will see that it is of the wrong sex. Its cable plug will not mate with the serial port socket on the rear of your Apricot. In fact, it will not even mate with the IBM, which is why an adaptor is supplied (which you can also give to your IBM-owning friend). Fear not #2! Procure or make up an RS-232 gender bender, nine-pin, unscreened, female-to-female (*you can get one for £13 inc VAT and P&P from* DEALS FOR READERS, *order number C002 — Ed*). Do *not* buy a null modem, which is similar but with crossover wiring instead of pin-to-pin connections.

The plotter I use is a Hewlett-Packard 7470A with serial option. Unfortunately, the Apricot has only one external serial port. I have found that you can unplug the mouse and plug in the plotter without the program losing touch with reality. As an unfrocked engineer, however, I must tell you that these plugs are not made to stand up to repeated insertions; the gold plating eventually wears off the pins and they become subject to the tarnishing which affects all baser metals. The best solution is to buy or make up a serial T-switch (available from Inmac and others at about £90). I tried simply paralleling the conductors at the connector but only wound up confusing the computer, mouse, plotter and operator beyond recovery.

Mousey requires the parameters of the serial port to be set up for 1200 baud in, seven data bits, two stop bits, and no parity. This seems compatible with my HP 7470A plotter which, although it can talk back, is not expected to do so by DOODLE. Leave the port set at 9600 baud out for maximum speed in transferring data to the HP buffer. Use the Apricot program SETUP.COM to make these assignments, and reboot the computer when finished. Incidentally, the HP plotter requires CTS to be implemented.

Now load the Trilex program CONFIG.EXE and follow instructions intelligently, selecting the Microsoft Mouse option and X-scale and Y-scale sensitivities of 50 points per inch. You are nearly ready for play.

It just remains to select the serial port before loading DOODLE. I do this with a utility called APR.COM from Ansible (*on the new ASD disk, available from* DEALS FOR READERS *again — Ed*) which toggles from parallel to serial or back each time it is called. APR is the first and the last entry in the batch file which selects my DOODLE subdirectory and loads DOODLE with the command DO [, the square bracket suppressing the title page display. There are several such utilities available, or you can just call PRINTER SERIAL and PRINTER PARALLEL.

Are you in DOODLE with the mouse suitably lashed to the serial port and a document displayed? Look at the bottom of the upper left corner's data. It reads 'keyboard cursor', meaning that you can curse about the screen using either the numerical keypad or the cursor keys. To toggle between key-driven cursor and mouse-driven cursor, enter the command CU. The display will change to 'external cursor' for the mouse.

You will not, of course, see the cursor crossbars yet because you have not evoked a cursor function. Type one, such as AD, which gives the freehand drawing facility. You will still not see the cursor crossbars. Fear not #4! Start to move the mouse and they will magically appear. The left button does the job that the space bar or Return key did before.

The right button seems to do nothing. Show no emotion, all is still ok. If you now type CU,C you will be in the mode where the cursor coordinates are displayed at the lower right of the screen. In this mode, the right mouse button resets the coordinates to zero.

All this palaver refers to DOODLE V.1.05C; and I will just mention that if you tried CU,X,Y, another cursor command in the DOODLE book, and it didn't work, don't be disheartened. I have never been able to get it to perform either. Maybe Trilex will have noticed it and fixed it by now. ◆

Keyboards and fonts:

Why bother?

By Ted Webber

Surely the font and keyboard utilities available with the Apricot are only of interest to afiçionados and commercial programmers? That was certainly my reaction when studying the list of extras which came with my machine. Let's get on with the word processing, spreadsheeting, or bank account hacking for which we bought the beast!

Then you come up against problems like not being able to print the ü — sorry, the pound sign. Well, I *can* if I put ü on the screen, because that character happens to correspond to the pound on my printer and home. But it is different again on two types of printer we have in the office.

Or you might like to have neat boxes around tables, instead of the somewhat ragged effect of using dashes and the broken bar symbol. The characters are all there on the computer, and probably on the printer too — but how do you get at them?

Similarly, writers of technical papers want access to mathematical symbols and the Greek alphabet — all in the standard font, but not on the standard keyboard.

We are told that only VUWRITER is adequate for this; but in fact other word processors can be adapted quite easily by editing the font and/or keyboard — though you must be careful with any of the high-end ASCII characters which the word processor might use as control codes.

In APRICOT FILE Release 2.6 Eddie Bromhead gave a few general ideas on changing fonts; and in November 1986 we had some thoughts on fixing up microscreens. I want to go through now the problems and solutions in matching keyboards, fonts, word processing software, and printers.

The simple one-off solution

For anyone needing one-off answers to these problems, Ansible Information provides excellent 'workaround' solutions on the ACOLYTE disk *(reviewed in* APRICOT FILE *Release 2.9 — Ed)* With ACOLYTE itself, you can specify a character by its ASCII code number; and it will appear at the cursor, actually within your document if a word processor is loaded.

But this facility requires five keystrokes to enter one character, so you won't want to do it as routine. And if you are using WordCraft, it won't work at all: WordCraft seems to obliterate any memory-resident programs loaded before it, or else they interfere with the word processing functions.

Remember also that ACOLYTE does not overcome the problem of the upper end of the ASCII range — many of the characters that you might want are pre-empted by word processors as control codes.

On the same Ansible disk will be found the APP suite of programs to interpret your keyboard so that your printer can understand it. In the example I gave above, the pound sign is ASCII 156 on the keyboard and screen but ASCII 129 on my printer; APP reads a customised data file, one line of which is '156 129' — meaning that whenever the computer tries to output ASCII 156 to the printer, APP will change it to 129. Hey presto, the pound sign appears on both screen and printer. No more L<BS>= to fudge it!

Since I have access to three different printers, I need three different APP.DAT files (or rather two, because the IBM character set on the NEC P5 printer matches exactly the standard Apricot font and keyboard — an important consideration when choosing a printer, incidentally. The other printers are a Centronix HS136 and an Epson BX80 clone). So here is a batch file that chooses the appropriate APP.DAT:

CHOOSPRN.BAT

```
ECHO OFF
:PRINTER
ECHO PRINTER MENU: redirect odd characters as
necessary
ECHO
ECHO     0  NEC-P5
ECHO     1  CENTRONIX HS136
ECHO     2  BX80
ECHO
CHOOSE
REM      See APRICOT FILE Release 2.10 for
         CHOOSE.C
IF ERRORLEVEL 3 GOTO PRINTER
IF ERRORLEVEL 2 GOTO BX80
IF ERRORLEVEL 1 GOTO HS136
IF ERRORLEVEL 0 GOTO SKIP
:HS136
COPY HS136.DAT APP.DAT APP3
GOTO SKIP
:BX80
COPY BX80.DAT APP.DAT APP3
:SKIP

... etc
```

The radical approach: fonts

Most printers these days can be programmed in dot graphics mode; and some can load a range of alternate fonts. But dot graphics are best left for graphics applications rather than the laborious business of fudging characters, and any alternate font probably won't match your keyboard requirements anyhow.

So in terms of finding a radical solution, it makes sense to assume that printers have a fixed character set — whereas the Apricot's 'soft' keyboard can be programmed.

This approach means you change the font such that (using the same example) the font table has the pound sign at ASCII 129; and you change the keyboard table so that the Shift-4 (pound sign) key implements ASCII 129. Now you have a computer which matches the printer, at least for this case.

The idea of separate font and keyboard tables is a little difficult to grasp at first. The font comprises the design of each of the 256 characters of the ASCII code. Characters 32 to 127 are generally standardised on all hardware; in general, they're the ASCII codes for alphanumerics and punctuation.

But characters 0 to 31 are control characters, only a few of which (like code 13, ^M or Return) have a universal meaning. They can also be used in the 'literal' mode to produce various graphics signs, but in that mode they are difficult to access from a word processor.

And characters 128 to 255 are a free-for-all. This higher end of the ASCII code set is reserved for specials, and is used by different hardware and different packages for their own purposes.

The font table has two somewhat distinct uses. First, it can be used to shift characters around within the ASCII table as in my example of the pound sign.

Second, it can be used to make the font more pleasing to the eye for a particular application. On my Portable I always use COMPACT.CHR, one of a number of different font designs available from Ansible; although smaller than the official Apricot/ACT font supplied, it is more legible because of the greater space allowed between lines.

So the first thing to do is to decide which design of font you like. Then you can start playing around with character sequences in the 128-255 range. How? There is always ACTIVITY, via the UTILITIES icon — which is possible, but slow and inconvenient.

I prefer the ladder-driven FONTEDIT.EXE, which is slightly less tedious. This sometimes loads with a split screen; the screen needs to be reset immediately before loading FONTEDIT, as in my FONTLOAD.BAT:

```
ECHO OFF
MXDUMP
Z
FONTEDIT
```

That works on an Xi with MS-DOS 2.11 and BIOS 3.1, incidentally. Use ECHO ^[z if you do not have the Z.COM utility. By loading a screen-dump program first (MXDUMP in this case) I can get a record of the changes very easily; an alternative is SHOWFONT.EXE, is a utility that came to me with MS-DOS 3.2. *(It's also on* DEALS FOR READERS *disk AP029 — Ed.)*

Some printers use ASCII codes in the range 160-254 for an italic version of codes 32-128, in which case your scope is generally limited to 129-159. Take care even there, because some printers duplicate control codes in that range. This is sometimes a useful facility, particularly for overstruck characters. If there is no ¥ sign on your printer for Yen, you might set up a key with Y<BS>= ... except that the BS (backspace) is ASCII 136 which has no effect on the screen. So in your font table, you'd put some sort of back-arrow in ASCII 136 as mnemonic of what it does.

Just what goes in the font table will obviously require a detailed study of your printer's technical reference manual, and of existing fonts which will be easy to manipulate with FONTEDIT.

Once you have saved your new font, in say NEW.CHR (all font files should have the CHR extension), load it with FONT NEW SAVFONT. That way, you can return to the default font with FONT SAVFONT.

Keyboard editing

So you now have a screen font that generally matches the printer. You probably want to use it to create a number of different keyboard configurations for various software applications, or to modify the key table supplied with the software. In particular, you'll often want to set up the Function keys and the MicroScreen to your taste.

The Apricot salesmen made a big song and dance about the advantages of the MicroScreen, but where can you find it documented in the manual "packs" issued with the machines?

The MicroScreen keys can be set up by ACTIVITY or EDITKB (see below), but there is no way to label the MicroScreen from within these programs. Nor do KEYS.EXE or LOADKB.COM (which comes with EDITKB) load labels for the MicroScreen when they load the key settings.

For this labelling, I use a text file such as MISC.ON (DEALS FOR READERS disk AP003) which

contains the ESC codes to send it to the MicroScreen when TYPEd. The MicroScreen can display two lines of 40 characters. As there are six keys to label in normal and Shift mode (Control-Shift is also available, but there's no space for the labels), each key label should be limited to six characters plus a space.

Alternatively, see EDIT.DOC on DEALS FOR READERS disk AP027 which describes how to use MS.EXE (also on AP027) to set up the MicroScreen and label the keys in one go. *(Roger also wrote this up for* APRICOT FILE *Release 3.2 — Ed.)*

In key table editing, you assign ASCII character numbers (for which the individual appearance has been fixed in the font) to Apricot key numbers. Alternatively you can assign a 'string' of characters to a key. In a word processing key table, for instance, you might notice that the same character is produced by both normal mode and Control-Shifting on the same key. So the Control mode is spare, and you assign 'Dear Sir' to that key — the semicolon, say. Whenever that keyboard is loaded, you can now get 'Dear Sir' with two keystrokes — Control and semicolon.

Attributes such as auto-repeat (which you might want for underlining but not for a vertical line) can be set. And there are a number of other options, set out clearly in EDITKB.DOC on DEALS FOR READERS disk AP027.

The Apricot has 96 keys, and with BIOS 3.1 each of them may be in normal, Shift, Control, or ALT mode — so there's plenty of room for all 256 characters in your font, plus some escape codes to drive the special keys such as the cursor arrows, and some strings of your own devising. The limitation in fact is not the number of keys, but the 1024-byte maximum size of a KB file plus a 128-byte header. *(In fact the version of* KEYEDIT *supplied with* BIOS 3.1 *overcomes this restriction and permits keyboard tables of up to 32KB — Ed.)*

Incidentally, if you want a single key to execute a program or batch file, just put a Return (it's ASCII code 13 and appears as a crotchet symbol in ACTIVITY!) as the last character of the key's string. For instance, to run something called TEST.BAT you could put 'TEST^M' on say the Control-B key for a single-keystroke execution of the program.

My policy is initially to install a modified default keyboard with MS-DOS functions such as DIR and COPY on the function keys. Then the batch file for each application loads an appropriate keyboard and restores the default on exit from the application.

Tools for keyboard editing include ACTIVITY again, and there is of course a version of ACTIVITY for every Apricot. A far better program, EDITKB, is available from DEALS FOR READERS — though unfortunately this won't run on the Portable or F Series machines. (KB files prepared by EDITKB on an Xi load perfectly on my Portable, however; the only key not catered for is F5, which is key code 97 on the Portable but non-existent on the Xi. EDITKB shows the Xi keys and their contents graphically; to use it for the Portable, remember that the eight function keys along the top row have become F1-F4 and F6-F9 on the righthand side of the Portable.)

You should also have handy a list of ESC codes (see APRICOT FILE Release 2.11) and control codes. These will enable you to understand what is already set up on your existing keyboard, and add others of your own design.

Undocumented EDITKB

When using EDITKB, note the following undocumented features:

1. LOADKB YOURKEYS automatically saves the existing keyboard in KEYTAB.SAV
2. EDITKB.DOC says 'If you enter the optional parameter *Knn* ... then key number *nn* will be ... the ALT key'. I found that this sometimes produced a KB file of more than 1152 bytes, which subsequently would not load. I think this happened when the keyboard to which I was trying to add an ALT key was prepared under a BIOS before 3.0
3. The NUL code ^@ (ASCII 0) is obtained from the numeric pad — enter '000'
4. When editing a key's contents, the BACKSPACE key does not delete — it is entered literally as ^H. But the DELETE key *does* delete and backspace. I usually CLEAR the existing key and start again to avoid getting muddled!
5. EDITKB$.KB has done something funny to the square brackets, so that the closing bracket is on the [key and ESC cannot be represented by ^[. Just use the ESC key instead. In many cases, the ESC will show in the 'prefix' box but not in the key table itself. So if you take a screen-dump record, write in the ESC codes you have inserted.

Word processor control codes

Writers of word processors like to think their products will be acceptable internationally acceptable, so they do not usually touch those peculiar Scandinavian items in the font. Instead,

they appropriate useful mathematical symbols such as the divide sign.

Within SuperWriter, you have some control of this business via Ansible's ACL.COM. Even so, SuperWriter will ignore ASCII characters greater than 240. So there is a clear case for doing Shifts in the font if you want to use them.

For WordCraft, I loaded WORDKEYS.KB into EDITKB and studied the characters assigned to the MicroScreen, Function, and Control keys. In many cases these are useful mathematical symbols or graphics. In fact, they have appropriated some of the single- and double-line border characters; it was impossible to use a complete set of either. The solution here lies in some redefinitions in WORDMSGS.005 and FULLMSGS.005, as an alternative or supplement to tinkering with the font, which you have already set up so nicely to suit your printer.

(See also Mr Griffiths' letter in APRICOT FILE Release 3.3 on alternative font facilities available within WordCraft itself.)

One word of warning about ACOLYTE: it appropriates the shift and control numeric pad, and anything you set up there (like BLOKGRAF.KB on AP027 for neat borders to tables) will be wiped out by ACOLYTE and won't be restored.

Final refinements

To install a font and/or a keyboard table into your system as default, thus speeding boot time, ACTIVITY has the necessary routines.

Alternatively, 'unhide' KEYTAB.SYS with one of the relevant utilities — try CHMOD, SWEEP or ATTRIB from the DEALS FOR READERS Public Domain disks. For instance, you'd use CHMOD via the command

```
CHMOD KEYTAB.SYS /SWNY
```

Once it is 'unhidden', you can then overwrite it with your new KB file *after its header has been deleted* — MS-DOS 3.2 has a utility for this. (I tried another way, which I do not recommend: DEBUG NEW.KB, then M 180 580 100 to overwrite the header, then W. It worked, much to my surprise.)

Don't forget to 'hide' KEYTAB.SYS again when you've finished, for example with

```
CHMOD KEYTAB.SYS/HRUX
```

To install a keyboard into an application such as SuperWriter, it is worth using RINSTALL.COM (DEALS FOR READERS disk AP007). I recently did this for WordCraft, and the batch file reduced from

```
KEYS WORDKEYS SAVKEYS LOADW
KEYS SAVKEYS
DEL SAVKEYS.KB
```

... to a single line, which is both shorter and quicker:

```
WC-LOAD
```

If all else fails, for instance because of SuperWriter's recalcitrance about ASCII codes 240-255, you can fall back on APP to redirect the character you have designated as the degree sign (ASCII 248, Control-5 on the keyboard) to the right place on the printer. But this won't help with WordCraft, which cannot work with TSR programs loaded (it wiped out SPRINT and HARE as well).

Keep a record of the new keyboards and fonts with screen dumps showing the keyboard layout for each mode — normal, Shift, Control, and ALT, if any.

Last words

I have indicated a plan of attack which accepts the printer as fixed, and so becomes the basis for the screen font, leading on to keyboard design for specific applications. But you'll be lucky to complete the process in one hit: it tends to be recursive, with applications such as word processing affecting the font and perhaps leading to a compromise involving APP. ◆

The toolbox

I have mentioned a fair number of software tools in this Workshop. Here is a collected list:

Program	Disk	Price	DFR ref
ACOLYTE	ACOLYTE	£57.50	AS035
ACL	ASD2	£57.50	AS006
ACTIVITY	[from Apricot plc]		
APP	ACOLYTE	£57.50	AS035
CHMOD	Disk No. 2	£15.00	AP002
or	ASD2	£57.50	AS006
COMPACT.CHR	ANSIBLEPATCH	£25.00	AS023
EDIT.DOC	Disk No. 7	£15.00	AP027
EDITKB	Disk No. 7	£15.00	AP027
FONT.EXE	[from Apricot plc]		
FONTEDIT.EXE	[from Apricot plc]		
KEYDIT.EXE	Disk No. 9	£10.00	AP029
KEYS.EXE	[from Apricot plc]		
MISC.ON	Disk No. 3	£15.00	AP003
MS.EXE	Disk No. 7	£15.00	AP027
MXDUMP	[from Sierra Systems]		
RINSTALL	Disk No. 7	£15.00	AP027
SHOWFONT.EXE	Disk No. 9	£10.00	AP029
Z.COM	ANSIBLEPATCH	£25.00	AS023
or	ASD2	£57.50	AS006

Notebook

FORMAT funnies on the Apricot

By Alan Solomon

Those who have reluctantly moved into the world of the IBM PC and its clones will by now have discovered the delights of the 5¼in diskette. It is floppy to the extent that it can be folded in half by enthusiastic postmen, it stores half as much as the old Apricot 3½in 720KB floppies, and you may notice that you sometimes get bad sectors when you format it — something that you never saw with the Apricot.

I've always ignored this last bit, and so has everyone else. But it turns out to be very important indeed.

FORMAT species

FORMAT comes with every MS-DOS version, but Microsoft doesn't write the program. The people who supply you with MS-DOS are expected to write parts of the operating system (the machine-specific parts). FORMAT is one of the machine-specific parts, and so ACT wrote the original diskette formatter for the Apricot. In MANAGER, it is part of a program called DISK; in ACTIVITY it is under the disk icon; and if you have MS-DOS 3.1, or if you're lucky, you'll have a file called FORMAT.EXE (for MS-DOS 2.11 users it's available from APRICOT FILE DEALS FOR READERS on disk AP020) which lets you format disks with the minimum of fuss.

Both on hard disks and on floppies, formatting is a two-stage process — though in practice, the two stages are combined in one program.

First comes the low-level format; this takes the circular tracks, formless and void, and writes down the magnetic markings that define and number the sectors. For example, a 3½in diskette has two sides (unless it only has one), 80 tracks on each side (unless it has 70), and nine sectors on each track. At this point, the diskette is covered with N codes (on the IBM PC it's full of ASCII 246s — there's no good reason for either of these).

Stage two is the high-level format. During this process, information is written on to the diskette — things like the boot sector, whether or not the diskette is copy-protected, a blank directory, and an empty File Allocation Table (the FAT tells DOS which bits of disk belong to which file). The Apricot formatter also puts on a volume name, which for some reason is always 'APRICOT'.

Bad sectors

On an IBM PC or clone, the formatting process identifies 'bad' sectors — which are parts of the diskette that have a manufacturing defect, and so cannot be formatted. On a PC-compatible, the entire track is considered to be bad; all nine sectors (4,608 bytes) is marked as unusable. The unit of space allocation on a diskette is a pair of sectors, called a cluster, so five clusters are actually unusable (5,120 bytes) and the formatting process reports this. Worse diskettes might have 10,240 bytes unusable, or even more.

Nothing is ever perfect; the important thing is to recognise this, and make sure you don't assume that all diskettes are perfect. So MS-DOS has a way of marking bad bits. Where the FAT entry would normally be 000 (meaning that the space is available for use), a bad sector is marked as FF7 — meaning 'bad cluster, do not use'. So if you peek at a DOS disk that you have just formatted where bad sectors turned up, check sectors one to four and you will see these FF7 codes.

But you don't get bad sectors with Apricots, do you? Oh boy, have I got news for you.

The tale of woe

I recently bought a batch of fairly crummy diskettes. The price I paid reflected this, but I found that of the 5¼in disks, about 95 per cent formatted up to the full 360K, 3 per cent had up to 15K of bad tracks, 1 per cent had more (so I used then for sending articles in to APRICOT FILE), and the balance I chucked away. Considering the price, they were a good bargain.

There were also a lot of 3½in disks in the job lot, and I was very pleased when they all formatted up to 720K on my Apricot. Then I started getting problems. I found that I couldn't always write to these diskettes; sometimes I got write errors, and sometimes the computer would hang. But they'd formatted fine, and I assumed my diskette drive was getting old, so I replaced it.

Then I acquired a Snap, an IBM PC clone with 3½in drives. I reformatted some of the diskettes that had given me problems ... and I found that some of them had a lot of bad sectors, some as many as 200KB. This wasn't entirely surprising, given the price I paid and the performance of the 5¼in diskettes in the same batch. What was surprising was that Apricot FORMAT hadn't noticed this and hadn't marked the bad sectors.

I took one such bad disk, and reformatted it again using MANAGER. Sure enough, no warning of bad sectors. But perhaps the Apricot drives are so

good that the diskette works fine in an Xi? I tried copying a bunch of assorted files on to that disk; and after about 300K, it started giving write errors. So the Xi couldn't write to that diskette. I took it out and reformatted it in the Snap — still lots of bad sectors.

My conclusion? The Apricot FORMAT does *not* mark bad sectors on diskettes with the normal FF7 entry in the FAT.

A fix

I think this is an appalling omission. Not everyone will be using the kind of rubbishy diskettes that I picked up for a song; but even the best-quality disk has the occasional defect. The Apricot won't mark that defect and avoid it — it will try to write to an unusable area on the diskette. The consequences can be pretty dire. At best you could have problems saving your data, and at worst, you could lose data. And by the way, does anyone know if the hard disk formatter marks bad sectors on a hard disk?

So what can be done? It turns out to be very simple. First, maybe I should buy better quality diskettes, and not try to save a bit by buying cheap ones. Cheap disks are ok on IBM PCs, where you can even buy single-sided diskettes and format them as double sided, but that would not be at all wise on an Apricot. I'm going to format all my diskettes on the Snap from now on, and only use the ones that format perfectly for the Apricot.

In the longer run, it would be a good idea if someone wrote a better version of FORMAT for the Apricot. One problem with that: most people think they already have a working FORMAT, so why should they buy another? Another difficulty: how can you tell all the thousands of Apricot owners about the problem and the solution? I guess that subscribers to APRICOT FILE are lucky, and the rest will have to rely on word of mouth.

But the best solution of all came to me after I had been through this process, and nearly finished this article. To add insult to injury, it was suggested to me by one of my customers — my God, it's really coming to something when I get free consultancy from my customers instead of the other way round. And to compound the fracture, the answer had been staring me in the face for a long time.

On diskette AP003 of the TP Group's collection of public domain disks there is (among other gems) SCAVEN, written by the great Tom Jennings back in 1983 *(and now fixed for Apricots — Ed)*. SCAVEN looks at each sector of the diskette, and if it is unreadable, it marks the FAT, so that MS-DOS won't use it. For £15, the problem is solved. ◆

Contributors in this issue

Dennis Jarrett is Editor of the newsletter. His magazine credits include being founder-editor of *16 Bit Computing*, *Office Automation Report*, *Practical Computing* and *Which Computer*. Among his books are *The IBM PC Book*, *The Sirius/Victor 9000 Book*, *The Electronic Office*, and *The Good Computing Guide* (*really* thick and very cheap too at £2.95 — or £3.00 inc P&P from DEALS FOR READERS).

Steve Allen works at a nutrition research unit in Cambridge as a sort of bureaucrat — financial planning, personnel and the like. Suckered by the cute looks of the PC/Xi (especially the dark grey one) he bought a whole fleet shortly before ACT dumped them. He now spends most of his time enjoyably wrestling with obsolescent software.

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

Lindsay Doyle currently writes for APRICOT FILE on a variety of esoteric subjects. Over the years he has contributed under various pseudonyms over 75 articles to the likes of *Personal Computer World*, *Microcomputer Printout*, 16 BIT COMPUTING, the Epson Users' Group, *Pet User Notes*, and various other Pet, CBM 64, and Vic publications. (Most of these are now defunct — is there a message here somewhere?) In addition to pounding away on an Apricot Xi of long standing, on which all the real work gets done, he is now playing with a new Amiga and HP plotter.

Garret Keogh is the staff writer for TP Group, a computer enthusiast and general dogsbody; since coming to TP he has acquired more knowledge about desktop micros and envelope-stuffing than he ever thought possible — at the grand old age of 19 he is adding to his computer education through a long process of trial and error(s). A music fan and committed socialist, his dream is to become a successful journalist ... these are his meagre first steps.

Alan Solomon is a writer and consultant who writes programs in Fortran, Pascal and assembler. They're marketed by his wife's software house, S&S Enterprises. He also runs a SIG in the IBM PC Users' Group.

Review

Fortran update:

Prospero's Pro-Fortran for GEM

By Eddie Bromhead

No sooner had I despatched my little missive on using GEM as a sort of super-GSX (APRICOT FILE Release 3.2) than I found Prospero Software had launched GEM versions of its Pascal and Fortran compilers. This review considers only the Fortran compiler; but the products' close links suggest that much of the comment will apply equally to the Pascal.

What you get

The GEM Fortran package has a sort of Borland Turbo feel to it, since it includes a complete developer's environment. Assuming you start off with GEM, this means opening the 'F-Bench' (Fortran Workbench) application F_BENCH.APP.

Menu selections from the workbench include the facility to open a Fortran source file and edit it; the editor is set up to look a bit like WordStar in non-document mode — but the mouse is integrated, and function keys are fully configurable.

Other options selectable from drop-down menus include compilation, linking, and running the resulting program under GEM (as an .APP); converting it to a GEM Desktop accessory (an .ACC file); or running it outside GEM altogether. The latter choice produces EXE files which may use GEM's underlying graphics, provided VDI functions are linked in.

Complicated link operations are done with the use of a 'script', and thus ease the compile/link/test cycle command sequence — it consists merely of setting some flags in the drop-down menus the first time around, and then subsequently just selecting the 'compile and link' option.

Also supplied in the package is the Probe debugging program. I rarely use utilities such as these, as I have ingrained habits from long-ago days — card decks and big mainframe batch processing. But they are useful if you can master the details.

The compiler

Apart from being set up to work under GEM and the Fortran Workbench F-Bench, so that all the files appear as overlays to F-Bench rather than as separate EXE files, the system appears similar to the fully-featured Fortran 77 compiler introduced by Prospero over a year ago.

This includes not only the full language specification, but also a number of useful subroutines and functions which extend the language — to get this degree of support even today with other leading compilers, like Microsoft's, you need add-ons like Harrison-Ward's utilities reviewed by me (APRICOT FILE Release 2.1).

Some things have been cut, however. The package does not support the 8087; and it has no 'big model' code — an application can only address arrays bigger than 64K bytes if these reside in blank common. (Prospero has allowed this in their compilers for some time, although data structures bigger than a segment are prohibited elsewhere. The restriction is lifted in so-called 'Jumbo model' programs, supported in the full system).

Support for GEM comes in a library of functions and subroutines, mostly translated from the C-bindings supplied by Digital Research but with some additions. The library covers both the AES and VDI operations, with some extensions to generate menu bars and the like; those normally demand the use of the DR Resource Construction Set to produce the .RSC files. Prospero's system can use .RSC files, if you have access to the Resource Construction Set (RCS) editor, or it can generate the necessary object trees itself via a series of subroutine calls. The library provides full coverage of the functions present in GEM, including those present only in GEM 1 (but dropped after the Apple lawsuit), those introduced especially for GEM 2, and those changed in scope or calling protocol between the two.

Why GEM?

Why not WINDOWS? The answer lies in three places. Firstly, there are a lot of Amstrads out there. Secondly, Prospero has 'professional' compilers for the Atari ST which runs GEM, and there is now a Prospero Fortran (or Pascal) for GEM on that machine too — it is source-code-compatible with the 8086 product. Finally, a lot of software is appearing for use under GEM, notably desktop publishing. No doubt Prospero has an eye on that as well (albeit via the Pro-Pascal compiler rather than the Fortran). In comparison, WINDOWS is appearing to be a much more difficult system to work with.

Overview

The compiler seems lightning-fast on a Xen. But I wouldn't be too happy about running the system on my twin-floppy (single-sided) PC; there are just too many required files.

The documentation is supplied in three paper-covered volumes in a slip case. Each of these is written in a readable style, and professionally typeset and printed. In fact, the documentation runs to over 600 A5 pages, of which about 500 are the descriptions of the VDI and AES bindings; and frankly the remaining pages are overkill — the system is so easy to use.

There are a couple of niggles, though. Once or twice, the documentation lets you see that the

Pascal product was developed first and the Fortran manual was written using search-and-replace on a word processor. And some of the demonstration files had rubbish at the end — SuperWriter sometimes does this to you — which stopped them compiling until edited.

Bindings are provided for functions which don't work although they are documented — VRQ_LOCATOR being the prime example. Knowing from DR that this function had been dropped from GEM long ago, I tried it. Sure enough, it does nothing!

The product could do with a catch-all 'fill in the arrays yourself and call GEMVDI' routine, just in case GEM 3 has a super-duper extra and the update is slow in coming from Prospero (friends tell me this isn't normally the case).

And finally, the editor doesn't let you know what line and column you are in. It is possible to find out which line, using the 'Go to line number' option in a drop-down menu; when this is selected, it shows the current line number in the selection box. The more useful column number is more difficult. Ideally, the display of these would be in a 'desktop accessory' showing not only the physical column number but also how Fortran interprets it; TABbing to column 8 initially is often interpreted as going to the traditional column 7 start of a Fortran statement. ◆

Pro-Fortran for GEM

Description:	Fortran compiler and development system for Apricots with GEM
Supplier:	Prospero Software Ltd
Address:	190 Castelnau London SW13 9DH
Telephone:	01-741-8531
Price:	£113 plus VAT
Configuration:	Any Apricot which uses GEM
Conclusion:	Well worth the money.

GEMVDI vs GSX — update

A couple of points to supplement my APRICOT FILE Release 3.2 piece. Doing single-key reads via GEMVDI to intercept cursor keystrokes is madness — I've given up. Either go the whole hog with the AES, or use MS-DOS services!

The GEMVDI single-key input function filters out the cursor keypress perfectly in GEM 2, so that the procedure I outlined only works on an Apricot (now where have we seen that before?) running GEM 1. If you use a routine to get keypresses, it catches cursor input if the screen is in TEXT mode; but when the screen is in GRAPHICS mode, they are filtered out. You have to designate an alternative cursor pad. Luckily, Apricot GSX suggests one for you — the numeric keypad, acting as a compass rose. This works perfectly on an IBM too.

Latest from the competition

RM-FORTRAN

Ryan-Macfarland is up to version 2.4. Changes from the previous version to 2.11 included more listing options and generation of 80286 code. It now has a no-co-processor library, and has dumped MS-LINK for a cut-down PLINK — a pity, because MS-LINK is much improved and only two-thirds the size even of the reduced PLINK. The full PLINK needs some getting used to, however, and the best training ground is RM's mini version. Also on the issue disks is an interface to allow calling of assembler-coded routines conforming to MS-FORTRAN calling protocols. Version 2.4 appears to be faster, and has a windowing debug environment like Microsoft's CodeView (see below).

MS-FORTRAN

Microsoft has released version 4.00 of the MS-FORTRAN compiler. Finally, it's nearly right! Full Fortran 77 at last, and fairly highly optimised — the code runs fast. EXE files are smaller, too.

On the minus side, many of the compiler options are invoked on the command line with difficult-to-remember case-sensitive codes. Also, the files and libraries are large, so that a lot of disk space is required. It isn't really feasible to use single-sided Apricot floppies any more, a hard disk is pretty much essential.

Support is there for 80186 and 80286 chips, and run-time diagnostics are much improved. More to the point, the package now includes a debugger which runs in windows (*not* WINDOWS!) but is decidedly not for Apricots.

A change in the way files may be OPENed is that formerly STATUS='NEW' meant 'may exist', but now this role is taken by STATUS='UNKNOWN'. STATUS='NEW' means that the file must *not* exist. Also, it is no longer possible to connect more than one unit to a file — a shame if you wanted to send two streams of output to the screen, or printer, or even NUL during testing.

Comparisons

Byte recently reviewed these two and **Lahey Fortran F77L**. The latter came out tops because of its compilation speed, even though it is 'co-processor only'. The Microsoft Fortran was found to have the edge at run-time. I'd agree.

The Q&A Clinic:

Questions by subscribers, answers by APRICOT FILE

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

Q. IBM format

I've been looking for a FORMAT program that will *write* 3.5in disks in IBM format. With BIOS 3.1 and greater it is possible to *read* such formats, and even to update them — but it is not possible to *create* them. The immediate solution has been to format all diskettes on an IBM, but if you run out of preformatted IBM diskettes no data can be transferred from the Apricot to the IBM.

A. There is a version of FORMAT which does all this (the command FORMAT B: /I formats the disk for IBM 720KB). Look for release number 1.5.4 dated 15 Jan 87; DEALS FOR READERS disk AP020 has this FORMAT on board.

Q. Bridge games

Does anyone know of a Bridge Game (quality more important that cost) for the Apricot Xi?

A. Sorry, but we have never heard of any Bridge games for the Apricot — has anybody else?

Q. IBM FORMAT option

The FORMAT option in MS-DOS 3 doesn't seem to work for formatting 360K IBM-type single-sided disks. Am I doing something wrong?

A. It won't work if you have BIOS 3.1 on a single-sided machine. That's because the IBM format requires a drive capable of formatting 80 tracks; single-sided floppies have only 40. (Single-sided 80-track disks are supposed to exist, but we've never seen one.)

Q. DOS 3 SORTDISK

Is there any way one can alter SORTDISK.COM (from your DEALS FOR READERS disk AP002) to work with MS-DOS 3?

A. Yes, and the solution is detailed at length in David Langford's *Disinformation Column* of December 1986 (APRICOT FILE Release 2.3). In brief, the problem arises because SORTDISK uses MS-DOS 2 functions not available in MS-DOS 1; and because it was produced for generic MS-DOS machines at a time when MS-DOS 1 was still quite common, SORTDISK prudently checks that it's running under MS-DOS 2 ... and it's a bit of a shame that the programmer didn't anticipate the possibility of an MS-DOS 3.

The Langford fix is easy enough; DEBUG SORTDISK.COM, and look for a patch of code containing MOV AH,30. You should find it at address 363, which unassembles like this:

```
XXXX:0363  B430    MOV AH,30
XXXX:0365  CD21    INT 21
XXXX:0367  3C02    CMP AL,02
XXXX:0369  7406    JZ 0371
... etc
```

MS-DOS function 30 is 'get DOS version number', returning the 'major' and 'minor' bits of the number in AL and AH respectively. The instruction at 0367 compares AL with '2', and the JZ 0371 skips to the program proper if and only if AL *does* hold '2' ... which means that things won't proceed if the version number is '1' or '3'. The neatest solution is to replace the JZ ('jump if zero flag set') with JNC ('jump if carry flag not set'). The CMP operation subtracts 2 from the contents of AL, setting the zero flag if the result is zero and the carry flag if AL is less than 2; that makes MS-DOS 3 acceptable.

In fact, the essence of the edit is to change one byte at 0369 from 74 hex to 73 hex. The quickest way to implement the change is as follows:

```
> E 369 73
```

Write that away, and the deed is done.

Q. DOS 3 revisited

Can you tell me what effect MS-DOS 3.2 and BIOS 3.1.2 would have on the software I use — SuperCalc 3, dBASE III, DataTalk, WordPerfect, etc — as well as the general system setup?

A. So far as we know, the new BIOS is highly compatible with most applications. We run some of the programs you mention and have experienced no problems; the only trouble we heard of involved those poor souls running BIOS 2.7 who changed to 3.1; Pulsar Accounts and Apricot Accountant didn't like the new BIOS and would require hacking. Subtle BIOS are however apparent to sensitive types such as programmers ...

One minor hang up concerns the MS-DOS utility programs such

as CHKDSK. Some of these check for the MS-DOS version number and refuse to run if they find the 'wrong' number, so they utilities won't work under 3.2. Brandishing DEBUG can put matters right though. Alternatively, you can get the pukka MS-DOS utilities from us on a forthcoming DEALS FOR READERS disk — some of them, like XCOPY (a superior COPY, halfway to DISKCOPY), are worth the upgrade by themselves.

Q. Laser vision

I am currently investigating the purchase of a laser printer. Apart from the confusing descriptions of the multitudes of machines on the market, I am puzzled by the term 'POSTSCRIPT-compatible. I have been told that if I don't buy such a machine I will not be able to upgrade it later (to *what* the dealer was unable to tell me). I have been quoted £2,499 for the Apricot (Kyocera) Laser without POSTSCRIPT and £4,999 for the Apricot (Apple) PS8. Please, in simple English, what are the relative merits of the respective machines (irrespective of price)? And, more important, what does the PS8 do that the Kyocera can't?

A. In performance terms, the Kyocera is a bit faster (a nominal ten pages per minute against Apple's eight); the Kyocera can't print bit-image graphics (it doesn't have the memory to hold all the digital information an image needs); the Apple has available a much wider range of typefaces; and the Apple can print much larger typesizes.

But POSTSCRIPT-compatibility is the key thing. POSTSCRIPT is a page description language, which is practical terms means a set of commands and controls that define exactly how a printed page will look — what goes where, what typestyles and sizes are used, what image shapes are printed. In practice, these commands are sent to the printer by the desktop publishing software — you set up the layout by using the DTP package's commands, it converts that into POSTSCRIPT equivalents to drive the laser printer.

The implication is simple: if you want to use a DTP package that employs POSTSCRIPT commands, you need a POSTSCRIPT-compatible printer. Apricot sells one of the two market-leading DTP packages, PAGEMAKER, and that does need a POSTSCRIPT-compatible printer (the other top name, Xerox's Ventura Publisher, uses a different page description language and so does not need POSTSCRIPT-compatibility in the printer — though it does demand a lot of memory in the printer to hold fonts and graphic images).

PAGEMAKER isn't the only DTP package, and it's not only DTP packages that can drive laser printers. Most top-line word processors like WordPerfect and Microsoft Word have drivers for laser printers, and you can do a good deal of fancy formatting using such software with a non-POSTSCRIPT printer.

In short, our advice is to forget POSTSCRIPT-compatibility and pay less for your laser — *unless* you want to use a DTP package that demands POSTSCRIPT-compatibility, such as PAGEMAKER.

Q. Keystrip encore

Some of the MicroScreen keys on my PC no longer work. Is there a repair that doesn't involve replacement?

A. The MicroScreen fails because it's connected to the keyboard PCB by a short length of flat ribbon cable which is relatively flexible and can eventually crack across a fold — thus breaking one or more of the connecting lines. There is no easy way to repair the ribbon cable, which is why our DEALS FOR READERS service supplies straight replacements for £25. It's an easy enough fit and we supply full instructions; all you do is peel off the flexible sheet with the keystrip on it, unplug the connector (you have to separate the two halves of the keyboard to get at the connector) and stick on the new strip.

Q. ACTIVITY alternative

I don't like ACTIVITY, but is there any other way to alter the meanings of individual keys? And is there any easy way of sending a control code to a printer?

A. Who *does* like ACTIVITY? You can use escape code ESC 4 to reassign keys; or there's an excellent keyboard editor on our DEALS FOR READERS disk AP027, price £15.

You can redirect a control code to the printer by ECHO code > PRN (you produce the ESC symbol by pressing ESC R — not ESC r). Alternatively you can use Basic in the form LPRINT CHR$(N), where N is the ASCII value of the control code. Simple, no?

Alternatively, there's EPSON.COM on DEALS FOR READERS disk AP002; this does basically the same thing in a simplified form.

Q. GSX screens

Does 800400EG.EXE have anything to do with GSX, and if so, what?

A. No, it has nothing to do with GSX — in practice it's a replacement for it.

Q. Access to Compuserve

I subscribe to the US on-line service CompuServe; but I don't know how I can access it more cheaply than by phoning the USA direct. I have obviously tried the CompuServe helpline but without any useful suggestions. Is CompuServe accessible via IPSS? If so, how? Or do you have any other suggestions?

A. Yes, CompuServe is available via PSS. Try the PSS Helpline on **0800 282 444** (a charge-free call) — they are genuinely helpful, and can tell you the address to enter for CompuServe when you get on-line to PSS. ◆

The interrupt

and how to use it

By Alan Solomon

The interrupt is one of the most useful facilities of a modern computer, but it's always treated with terrible awe. There's a mystique about it — it's a kind of *pons asinorum* that sorts out the Real Hackers from the Sunday Programmers. But the time has now come to dispel this illusion; interrupts are a piece of cake — they're just a grandiose term for 'subroutines'.

Why interrupt?

The interrupt was originally conceived as a way to handle peripherals; as a kind of warning that something somewhere needs attention, or requires some action.

Imagine you're sitting in your armchair reading a newspaper, and you are expecting someone to call at the front door *and* maybe someone to call on the phone, tool. Now, imagine the situation if your front-door didn't have a knocker, and your phone didn't have a bell. At regular intervals you'd have to pick up the phone receiver on the off chance that there might be someone on the other end of it who wants to talk to you, and you'd have to keep going to the front door in case anyone is sitting on your doorstep casually waiting for you to come and look.

This is going to make reading the newspaper a bit laborious to say the least. For a start you'd have to make a decision about how often you'll check the situation. Too often and you won't make any progress with the paper, and too rarely you run the risk of people hanging about interminably waiting for attention, *and* holding up processes which can't happen until after these consultations.

Then add to these two tasks the need to check whether the kettle has boiled, and if you are hungry yet, whether you ought to uncross your legs, and all the many other things that need attention, and you can see why the human brain has evolved as an interrupt-driven device. Life is much easier when things 'ask' for attention, whether it's a stomach or a front door. The alternative is a 'polling' system — where you keep going round all these various peripherals, seeing if any of them has anything for you.

Nesting interrupts

Notice also that while you are servicing one of these 'interrupts', another interrupt might demand attention (while opening the door and dealing with the caller, the phone rings). The chances are tlat you'll suspend processing the first interrupt, and deal with the second. You might not fully process the second before returning to the first — you ask the phone caller to hang on while you finish with the front-door caller. Or you might deal with the phone call, then go back to finish at the front door.

Interrupts can be interrupted, and the interruption can be interrupted, and so on to any level. You know how some days it seems like you never actually do any work? That's because you spent the whole day servicing interrupts, so you're cleverer than you thought!

Human beings can in theory nest interrupts like this indefinitely, but in practice there is a limit to the nesting — the limit comes when you try to resume processing a suspended interrupt, and find that you can't remember where you'd got to.

Computers work in much the same way. It would be ridiculous if the CPU had to check continuously to see whether, at any given time, any one of the ninety or a hundred keys had been pressed. It's much more sensible if hitting a key generates an 'interrupt' that services the keystroke and, usually, puts it in a buffer for dealing with later (although some keystrokes need immediate attention — such as a CTRL-ALT-DEL reset on an IBM-compatible). This way, when the key is hit, it sends a message to the CPU, 'Hey, I've been hit. Do something with me!' and the CPU is suitably prompted to get out of its armchair.

Hardware interrupts

The keyboard is only one of the devices that can interrupt the CPU; so can the diskettes and the hard disk. These interrupts are called 'hardware' interrupts and they can be disabled in order to prevent the CPU being interrupted while it's doing something critical, like changing the stack segment. Of course, if interrupts *are* disabled, the keyboard can't interrupt the CPU — which disables CTRL-ALT-DEL and the like — so the facility to disable should be used with care.

When you, in your human guise, are interrupted, you make a kind of mental note as to where you'd got to; and if the interruption is interrupted, you makes a second note of where you'd got to in that task ... and so on. When a computer is interrupted, it must do a very similar thing.

Before servicing the interrupt, it must store the current state of all its changeable variables (the registers amongst other things). Then, after the interrupt is serviced, it can restore things to how they were, and take up exactly where it left off before the interrupt occurred.

It's the programmer's job to make sure that this is done. Normally, the changeable variables (the registers and so on) are stored on the stack. The advantage of sticking them in there, rather than in a *fixed* location is that the interrupt can interrupt itself, as it can store the states of the registers for each invocation. If it used fixed locations, the second time it stored the registers, it would overwrite the first lot.

This leads us to one of the many and varied bugs in MS-DOS — it isn't 're-entrant', and this is a big

problem for interrupts. If MS-DOS is in the middle of doing something, and you interrupt it and ask DOS to do something else, you'll find that when it's done it dumps you somewhere in the middle of Kansas with no shoes. Or to put it another way, the computer hangs.

This is because MS-DOS wasn't written very well. Specifically, MS-DOS maintains no fewer than three stacks; but it *doesn't* maintain pointers to them on the main stack. So, if MS-DOS is in the middle of doing something, you'd better not ask it to do anything else.

Fortunately this only affects terminate-and-stay-resident programs, and most common or garden programs don't have to worry about it. By the way, the IBM BIOS *is* re-entrant, and the Apricot BIOS is probably. So, if you know what you're doing, you can get round this MS-DOS non-re-entrancy problem by using interrupt 21 hex, function 34 hex, which returns a pointer (in ES:BX) to the DOS-is-busy flag. This is not documented in the manuals.

Software interrupts

All of this may be very interesting, but it's not of enormous interest to you as a programmer unless you're writing a memory-resident program or a BIOS.

The *software* interrupt, on the other hand, is very interesting indeed. It is the key to using all the low-level routines offered by MS-DOS and all the routines in the BIOS. So how do you use it?

Well, interrupts are used rather like subroutines, and there are 256 different ones available in an Intel 8086 (or an 8088) — the bottom 1KB of memory holds 256 four-byte pointers and each of these pointers stores a single address reference in the usual SEGMENT:OFFSET form. Each given address is the start of a routine stored somewhere else in memory.

So if you call interrupt 0, the computer looks at the address in memory locations 0 to 3, which are the very bottom four bytes in memory, and it calls the subroutine that's stored there. If you call interrupt 1, addresses 4 to 7 are used; and so on.

You can call these interrupts from assembler by using the INT mnemonic, or from Turbo Pascal by using the predefined procedure INTR. Parameters are passed to the interrupt via the registers, and information is returned to the calling routine the same way.

Putting them to use

So what useful subroutines can we access via the interrupts? First, let's take MS-DOS (or PC-DOS; there is no real difference between them, contrary to what you may have heard). Interrupts 20 to 27 are particularly useful because they will work on any MS-DOS computer, and are *generally* useful for anything to do with disks and files.

Interrupt 21 (all interrupt numbers are in hexadecimal) is probably the most useful interrupt in the history of the world. It is called the MS-DOS function call. Disk activity is done via interrupt 21; open, close, read, write, delete, rename, random access, create subdirectory, check disk free space — anything to do with disks. Interrupt 21 also lets you do some screen writing and keyboard reading.

There are 57 varieties of interrupt 21, though MS-DOS uses some of these internally. I'm not going to describe each one; if your manuals are as bad as most, get *Programmer's Guide to MS-DOS* by Dennis Jump. MS-DOS 3 has a few more calls for networking, and each upgrade of MS-DOS 3 has a few more than the last.

If you're using MS-DOS calls the chances are that you're using assembler; most higher-level languages don't require recourse to interrupts because they usually provide most of the facilities you might need as built-in features. In assembler, calling an interrupt is a piece of cake.

For example, suppose you want to know what free space is available on a disk. You load the registers according to the manual. Put 3600 into AX, and a number that asks for the disk drive into DL — 0 means default, 1 means drive A, 2 means B and so on. We then do the interrupt; and lo, CX is the bytes per sector, AX is the sectors per cluster and BX is the number of free clusters. DX also gives you the total number of clusters on the drive.

```
MOV AX,3600H
MOV DL,1
INT 21H
```

These three instructions are all it takes to check on drive A. I told you it was a piece of cake.

Programming interrupts

If you are using a higher-level language, accessing the interrupts is a bit trickier. The reason for wanting to do this might be that the language doesn't provide all the facilities you want; very few would have an inbuilt way of telling you the disk free space, for example. The only way to call the interrupt from some languages is to call an assembler subroutine, and your manual should show you how to interface a language with assembler.

The simplest thing to do is to write a specific assembler routine to do precisely what you want and call *that*. But it would be nice to be able to write a more general routine that would call *any* interrupt.

The problem is that the 8086 requires the interrupt number to be hard-coded ('immediate' format) into the code. So you can't tell the 8086 to do an interrupt number that's stored in some memory location or register. This isn't a problem if you're doing an MS-DOS function call, as they are *necessarily* interrupt 21. But for generality, we want to be able to do any interrupt.

The disk accompanying Peter Norton's *Inside the IBM PC* (quite a good book) accomplishes this by offering about a million different assembler routines. But there is a neater way, and it involves

one of my favourite tricks: self-modifying code. The routine patches itself in the course of doing its job. (Of course, this will fail on an 80386 machine running a decent operating system.)

First, a warning to anyone who tries to use the code following. All of it is taken from working programs (either IBM or Apricot), but I've left out the uninteresting startup and ending code — the variable declarations, or ASSUME pseudo-ops, or whatever. The advantage of this is that I can get straight to the point; the disadvantage is that the code won't work if you just type it in as it appears here. But if you know enough about the language you're using to be able to take advantage of this code, you'll have no trouble putting in the necessary prologue and epilogue. And if you don't know what you are doing, you shouldn't be using interrupts.

```
NAME    INTRPT
FRAME   STRUC   ;  Define structure (DSECT)
SAVEBP  DW  ?   ;  Caller's BP
SAVERET DD  ?   ;  Four-byte return address
                   pushed by caller
DXREG   DD  ?   ;  Address of word with DX value
CXREG   DD  ?   ;  Address of word with CX value
BXREG   DD  ?   ;  Address of word with BX value
AXREG   DD  ?   ;  Address of word with AX value
NUMBER  DD  ?   ;  Interrupt to be performed
FRAME   ENDS
CSEG    SEGMENT 'CODE'
DGROUP  GROUP   DATA,STACK,CONST,HEAP,MEMORY
        ASSUME
        CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
INTRPT  PROC    FAR
        PUBLIC INTRPT
         PUSH   BP
         MOV    BP,SP
;                Point SP at FRAME structure
         MOV    AL,BYTE PTR [BP].NUMBER
;                Move the interrupt number into
                 the right place, patching the
                 subroutine
         MOV    CS:INTERUPT+1,AL
;                Set up registers for interrupt
         MOV    SI,[BP].AXREG
         MOV    AX,[SI]
         MOV    SI,[BP].BXREG
         MOV    BX,[SI]
         MOV    SI,[BP].CXREG
         MOV    CX,[SI]
         MOV    SI,[BP].DXREG
         MOV    DX,[SI]
         PUSH   BP
;                A bug in the BIOS makes it
                 necessary to save and restore BP
                 for some interrupts
INTERUPT EQU    THIS BYTE
;                Perform the interrupt which has
                 been patched
         INT    16
         POP    BP
         MOV    SI,[BP].AXREG
;                Save the register's VAR
                 parameters, so that Pascal can
                 access them
         MOV    [SI],AX
         MOV    SI,[BP].BXREG
         MOV    [SI],BX
         MOV    SI,[BP].CXREG
         MOV    [SI],CX
         MOV    SI,[BP].DXREG
         MOV    [SI],DX
;                Pop the parameters off the stack
         POP    BP
         RET    20
INTRPT   ENDP
CSEG     ENDS
         END
```

Then the calling program can treat the procedure as if it were declared thus:

```
PROCEDURE INTRPT(NUMBER:BYTE; VAR AX,BX,CX,
  DX: word); EXTERN;
VAR  AXR,BXR,CXR,DXR: word;
```

And it can call it thus:

```
        INTRPT(16,AXR,BXR,CXR,DXR);

PROCEDURE SetMode(Mode:word);
        {Set CRT mode status:
        0 = 40x25 B&W       1 = 40x25 Colour
        2 = 80x25 B&W       3 = 80x25 Colour
        4 = 320x200 Colour  5 = 320x200 B&W
        6 = 640x200 B&W     7 = Monochrome    }
BEGIN
        AXR:=Mode;
        INTRPT(16,AXR,BXR,CXR,DXR);
END;
```

Fortran can use exactly the same assembler scheme; only the interface at the Fortran end is different. But be careful — I'm assuming that your compiler pushes four-byte addresses on to the stack, and not all of them do. And I am also making an assumption about the order in which the parameters are pushed. In some cases, you can choose what the compiler does with a metacommand. Read the manual.

Microsoft Basic (the old compiler) can also use this method to call interrupts. But Basic pushes two-byte addresses on to the stack; so DXREG, CXREG, BXREG, AXREG and NUMBER must be defined as DW instead of DD, and the RET argument should be 10 instead of 20.

It's rather easier with the newer Basic compilers for IBM-compatibles. Turbo Basic has a subroutine called INTERRUPT, and there's a similar thing in Microsoft's QuickBasic.

There is a short cut for Microsoft Fortran and Pascal via a badly-documented function that you can call, DOSXQQ, which provides access to many of the MS-DOS function calls.

```
function dosxqq(command:byte, parm:word):byte; extern;
```

The first parameter tells MS-DOS which function to call; it is used to set the AH register. The second parameter is used to set the DX register. The value of CX can be accessed via a variable called CRCXQQ, and ES via CESXQQ. DX is available in CRDXQQ. This is covered, though very poorly, on page 119 of volume 1 of the Microsoft Pascal manual (version 3.13), and not at all in the MS-Fortran manual (but it *is* available, as an inspection of the library will show).

I have never used DOSXQQ in any of my programs, as I don't like undocumented features

(they have a nasty habit of disappearing with upgrades); but here it is.

```
PROGRAM DODOS(INPUT,OUTPUT);
FUNCTION DOSXQQ(COMMAND:BYTE; PARM:WORD):BYTE;
EXTERN;

VAR
 DOSCALL : BYTE; {DOS RETURN CODE (VALUE IN AL)}
 COMMAND : BYTE; {DOS FUNCTION NO.}
 PARM    : WORD; {DOS PARAMETER LIST (DS:DX)}
 MONTH   : WORD; {CURRENT MONTH}
 DAY     : WORD; {CURRENT DAY}

 MESSAGE : LSTRING(80);
 {CHARACTER STRING TO BE PRINTED}

 CRCXQQ [EXTERN] : WORD;
 {EXTERNAL VARIABLE — VALUE IN CX}

 CRDXQQ [EXTERN] : WORD;}
 {EXTERNAL VARIABLE — VALUE IN DX}

BEGIN
 {SET FUNCTION NO. FOR PRINT STRING}
   COMMAND := #09;
 {POINT DS:DX AT STRING}
   PARM := WRD(ADR MESSAGE[1]);
 {$ TERMINATES STRING}
   MESSAGE := 'TODAY'S DATE IS: $';
 {CALL DOS TO PRINT STRING}
   DOSCALL := DOSXQQ(COMMAND,PARM);
 {SET FUNCTION NO. FOR GET DATE}
   COMMAND := #2A;
 {CALL DOS TO GET DATE}
   DOSCALL := DOSXQQ(COMMAND,PARM);
 {MONTH IS RETURNED IN DH, DAY IN DL}
   MONTH := HIBYTE(CRDXQQ);
   DAY := LOBYTE(CRDXQQ);
 {FINALLY, USE CONVENTIONAL WRITELN}
   WRITELN(MONTH:2,'-',DAY:2,'-',CRCXQQ:4);
END.
```

All the fashionable hackers use C these days. I'm not too keen on C, but it has its uses; and you can certainly do an infinite amount of damage with a C program. It's fairly easy to get down to a low level, too.

One word of warning — get yourself a good C compiler. I started off using Zorland C (this is a rebadged Datalight C). The compiler works ok; but part of the deal was that Zorland had to rewrite the manual, and it isn't very good. In particular, I wasted an entire day's programming because Zorland didn't document the interrupt call correctly, and even the example given was wrong. When I realised this, I hurled the wretched thing across the room in fury and I haven't touched it since. Have a look at Turbo C, as it is rather good; or Microsoft C, as it has a source level debugger, CodeView. Microsoft C is the language for Windows, too, and there are a *lot* of toolboxes available.

```
/*  Register structure required for functions
    int86() and int86x() is defined below.
INT86( INTERRUPT, REGSIN, REGSOUT )
    ... performs an 8086 software interrupt.
    interrupt is an integer holding the interrupt
    number, regsin and regsout are pointers to a
    structure (defined below) which hold the
    values of the 8086 registers on entry to the
    interrupt handling routine and on exit from
    it.
INT86X( INTERRUPT, REGSIN, REGSOUT, SEGREGS )
    ... is similar, but with an additional
    pointer to the segment registers. */

typedef struct   { int ax,bx,cx,dx,si,di,cflag; }
                 REGS;
struct WORDREGS  { unsigned
                 ax,bx,cx,dx,si,di,cflag; };
struct BYTEREGS  { unsigned char
                 al,ah,bl,bh,cl,ch,dl,dh; };
union REGS       { struct WORDREGS x; struct
                 BYTEREGS h; };

    /* alternate def */
struct SREGS { unsigned es,cs,ss,ds; };

/* Define macros to get at the segment and offset
of a far pointer. */
#define FP_OFF(p)  ((unsigned) ((char *)(p) -
                   (char *)0))
#define FP_SEG(p)  (((unsigned *) (&(p)))[1])
doanint()

{  REGS reg;
   struct SREGS sreg;
   int i;
   char buffer1[512];
   char buffer2[512];
   char *p1;
   char *p2;
   p1 = buffer1;
   p2 = buffer2;
   eg.ax = 0;
   int86( 0x13, &reg, &reg);
   reg.es = FP_SEG( p1 );
   reg.bx = FP_OFF( p1 );
   reg.ax = 0x201;
   reg.cx = 0x1207;
   reg.dx = 0;
   int86x( 0x13, &reg, &reg, &sreg);        }
```

After the complexities of assembler, the contortions of Basic, the unreadability of C and the even worse incomprehensibilities of Microsoft Pascal and Fortran, it is a real relief to tell you about Turbo Pascal. Borland really makes it easy. First you declare a register type (and with Turbo version 4, that's done for you), then you declare a register variable.

```
type
 Registers =
  record
   AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags :
   Integer;
  end;
{With Turbo Pascal version 4, you don't need the
above — a Registers type is predefined for you}
var
  Regs : Registers;
```

Then you load the necessary data into the registers:

```
regs.ax := $3600;  {to get disk free space}
regs.dx := $0001;  {drive A}
msdos(regs);
{now regs.ax, regs.bx, regs.cx and regs.dx
contain information about drive A, and my program
can take whatever action is needed}
```

I really cannot understand why anyone uses any other language! ◆

Batch File Recipes:

Three batch-file Super macros — Variables in XQT files Alternative boilerplating Beating the subdirectory ban

By Steve Allen

I found that I was starting to write a number of very similar XQT files to run under SuperCalc. In some cases these performed quite specific tasks, such as merging portions from a number of spreadsheets. But often I required no more than loading, waiting (using the ampersand), saving and quitting.

The use of a batch file to write these routine macros automatically and then run them has proved a great time-saver. The file is called RUN.BAT; the syntax is RUN <FILENAME> where <FILENAME> is an existing CAL file.

RUN.BAT

Line numbers are included for reference — don't enter them when keying in this batch file.

```
1   ECHO OFF
2   CLS
3   IF %1@==@ SC3 CR
4   IF %1@==@ MENU
5   IF NOT EXIST %1.CAL GOTO ERROR
6   :START
7   IF EXIST %1.XQT SC3 %1
8   IF EXIST %1.XQT MENU
9   ECHO /L%1 >%1.XQT
10  ECHO A&/S%1 >>%1.XQT
11  ECHO OA/QY >>%1.XQT
12  GOTO START
13  :ERROR
14  ECHO FILE %1.CAL DOES NOT EXIST ^[B
15  PAUSE
16  MENU
```

The first two lines are probably familiar. **Line 3** is a test. If no variable is entered after the command RUN, SuperCalc loads itself and runs a default XQT file called CR.XQT — which stands, rather unsurprisingly, for 'Carriage Return'. That is all it contains, and using it means I'm presented with a blank spreadsheet instead of the spurious "Enter '?' for HELP or 'Return' to start" message.

Line 4 means that on leaving SuperCalc I am whisked back to my menu by MENU.BAT. This line is important; without it, the batch file continues processing unnecessarily. If you do not use a menu structure you could substitute something like GOTO END and have a label :END to close the file.

Line 5 tests for the existence of a CAL file with the variable name. If it does not find one, the batch file goes to the error sequence in the final three lines.

Line 7 checks to see whether there is already an XQT file of the specified name. If there is, it loads SuperCalc then runs the named macro to load the relevant spreadsheet(s). **Line 8** performs the same function as line 4.

Now for the interesting part. Having established that there is a spreadsheet with the variable name and that there is no existing XQT file to access it, the batch file proceeds to write one with the same filename (*sans* extension) as the spreadsheet. It does this by echoing a string — in this case a series of SuperCalc commands — to a file.

What **line 9** does is send the text '/L<NAME>' to a new file called <NAME>.XQT. As regular SuperCalc users will know, this command line means Load <NAME>.CAL.

As **line 10** of the batch file adds a new line to the freshly created macro by the use of *two* greater-than signs, the Return necessary for the success of the previous SuperCalc command is automatically placed at the end of *that* line. **Line 11** works in the same way, and the XQT file thus created looks like this:

```
/L<NAME>
A&/S <NAME
OA/QY
```

And that means:

/L	Load
<NAME>	name of a CAL file to load
<Return>	
/A	All of it
&	pause for keyboard Input
/S	Save
<NAME>	name of a CAL file to save
<Return>	
O	Overv rite existing file
A	save All of it
/Q	Quit?
Y	Yes

Note that it's important to use the %1 parameter form in the second line of the macro, even though SuperCalc allows the use of the Escape code ^[as a shorthand to re-enter a previous filename. That's because if you have, say, consolidated another spreadsheet, the filename summoned by ESC will not be one with which you started the XQT.

Line 12 of the batch file forms a loop back to the test in line 7. This time there *is* an XQT file of the specified name; so it can enter SuperCalc, having created and saved a basic macro for future use.

SuperWriter boilerplating

This one bears no relation to the boilerplating technique described in the SuperWriter manual and ingeniously automated by Bob Smith (APRICOT FILE Release 2.9). It does however perform the same function, namely the inclusion of standard phrases and clauses to order in routine documents — which in my case are employment contracts.

APP.BAT

Line numbers are included for reference — don't enter them when keying in this batch file.

```
1   ECHO OFF
2   CLS
3   :LOOP
4   IF EXIST B:%1.* GOTO WRITEXQT
5   ECHO ^T^E^D^D^E^[x^[ >>C:APP2.XQT
6   SWKB1 B:APP
7   MENU
8   :WRITEXQT
9   IF EXIST B:%1.$ GOTO SLOTTERS
10  ECHO ^B^[ib:%1 >>C:APP2.XQT
11  :RETURN
12  ECHO  >>C:APP2.XQT
13  SHIFT
14  GOTO LOOP
15  :SLOTTERS
16  ECHO ^[f¥ >>C:APP2.XQT
17  ECHO  >>C:APP2.XQT
18  ECHO ^G^[ib:%1.X >>C:APP2.XQT
19  GOTO RETURN
```

The technique for writing macros used here is the same as for RUN.BAT. It has a speed advantage over the Smith method, though at some cost in disk space. This is mainly due to a change of approach which involves keeping each clause as a separate file on a library disk or directory. Also on this disk is a template or skeleton of the basic contract into which the various clauses will be inserted.

The contracts contain two sorts of routine clauses which are suitable for automation in this way. Some will have to fit into pre-determined gaps in the skeleton text, others merely accumulate at the foot of the document. To cope with this, there have to be two types of filename on the library disk and two modes of execution in the batch file (and therefore in the macro).

The file is called APP.BAT (for 'Appointment letter') and the syntax is:

```
APP <clause 1><clause 2><clause 3> ... etc
```

... where each clause is the name of an existing file on the library disk.

Line 4 is the first test. Its primary function is to see if there are any more variables to incorporate in the macro — whether the list of specified clauses has been exhausted, in fact. If there is a clause to use (and an existing file in the library) control passes to the second test in line 9.

Line 9 establishes what sort of clause is to be used — one to slot in or one to put at the end. If the clause is a 'slotter' (one with a filename extension '$') the program moves to another label, to wit :SLOTTERS. It then writes the first part of the macro.

Note the use of the double greater-than signs again. It is essential to use this throughout; the same routine may be invoked many times, and a single greater-than character would create a new file overwriting any previous parts of the XQT.

Note also the use of a RAMdisk, drive C. This is done chiefly to speed up the operation, but also because the macro does not need to be kept once the job has been done.

In short, once in the macro **lines 16-18** find the first mark '¥', (CTRL-4) which I use in the skeleton text to denote the gaps for slotting, delete it and include the named clause. The next line moves up to the label :RETURN which inserts a Return to accept the inclusion of the imported file in total. This is followed by SHIFT, which moves the batch processing on to the next variable in the queue. The whole process then restarts from the :LOOP.

If the specified variable is not a 'slotter' but an 'accumulator', it will fail the second test. Instead of moving to the :SLOTTERS label, the program uses **lines 11 and 12** to add to the macro. Rather than search for a mark, these lines merely include the new clause at the foot of the document — hence the CTRL-B. Having written these lines, APP.BAT does another SHIFT; and the routine repeats until the supply of variables is exhausted.

(It's important to deal with 'slotters' before 'accumulators' when entering the command to run the batch file. If you try it the other way around, the cursor will be stuck at the bottom of the text unable to find any marks.)

Only then does **line 5** come into play. Its function is to return the cursor to the top of the document, emit what David Langford has described as a 'peevish beep' to let you know it has run its course, and finally close down the XQT. This may sound like an unnecessary frippery; but in fact some sort of end line or final line is essential. Without it, the normal end-of-file marker generated by MS-DOS — a long line of ^@ characters — will wreak havoc.

When this has concluded the writing of the macro, **line 6** loads SuperWriter (more specifically, the wonderfully enhanced Ansible-keyboard version of it) and runs the hitherto unmentioned and very brief APP.XQT which is also held on the library disk. All this contains is a Return to enter Edit mode, the name of the skeleton file, and the invocation of the just-written APP2.XQT:

```
<Return>
B:SKELETON.APP
^[xC:APP2
```

On leaving SuperWriter **line 8** rounds off the process by calling MENU.BAT to return to the main menu.

Beating the bans

Here's another one which might interest you. It gets around Superwriter's inability to deal with subdirectories, albeit not very speedily; it notes the current directory, copies the workfile to RAMdisk, edits it, and then returns it to whence it came. It lives in my root directory and is called SW.BAT.

To edit a file in the current directory the syntax is SW <FILESPEC> /S.

Note that the PATH currently set at the time *must* include the root directory so that SW.BAT can always be found.

SW.BAT

Line numbers are included for reference — don't enter them when keying in this batch file.

```
1   ECHO OFF
2   CLS
3   IF %2@==@ GOTO NEXT
4   IF %2==/S GOTO CHANGEDIR
5   IF %2==/s GOTO CHANGEDIR
6   :NEXT
7   CD\SW
8   IF %1@==@ MENU
9   IF NOT %1@==@ RUNSW %1
10  :CHANGEDIR
11  IF NOT EXIST %1 GOTO NOTFOUND
12  ECHO %1 BEING COPIED TO RAMDISK SO THAT IT
    CAN BE ACCESSED BY SUPERWRITER
13  COPY %1 C: >NUL
14  ECHO ECHO OFF > C:RETURN.BAT
15  ECHO A: >> C:RETURN.BAT
16  CD  >> C:RETURN.BAT
17  CD\SW
18  SWKB1 C:%1
19  SWKB1 AMEND
20  COMMAND/C C:RETURN
21  DEL C:RETURN.* > NUL
22  CLS
23  ECHO COPYING EDITED ^[0 %1 ^[1 FROM RAMDISK
    TO ORIGINAL DIRECTORY
24  COPY C:%1 A: >NUL
25  DEL C:%1 >NUL
26  MENU
27  :NOTFOUND
28  ECHO FILE ^[0 %1 ^[1 DOES NOT EXIST OR IS NOT
    IN THIS DIRECTORY ^[B
29  ECHO SEARCHING ...
30  WHEREIS %1
31  ECHO
32  PAUSE
33  MENU
```

This should be fairly clear following the previous two Recipes. The batch file MENU.BAT is invoked in line 8 and elsewhere; RUNSW in line 9 runs a batch file called RUNSW.BAT, which among other things calls SWKB1.COM. The error-check in lines 11 and lines 27-33 utilise the public domain utility WHEREIS.COM (on DEALS FOR READERS disk AP002). Line 19 has Ansible's SWKB1.COM calling my AMEND.XQT, a file which looks like this:

```
<Return>
C:RETURN.BAT
^J^J
CD ^[s
^J
```

Contribute!

Why not share your own useful utilities? Let us have your own home-brewed batch files, and we'll write them up for this *Recipe* series.
Just drop us a line at APRICOT FILE, PO Box 509, London N1 1YL.

About the newsletter

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

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

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

APRICOT FILE is a publication of the TP Group — which until we get the name changed is the trading style of Permapoint Limited, registered in England no 1929195.

Postal address

TP GROUP
PO Box 509
London N1 1YL, England

Personal calls and deliveries

TP GROUP
Unit 4, 47-51 Wharfdale Road
London N1 9SE, England

Phone, E-mail, Telex

Telephone	**01-833 3501**
Telecom Gold	**81:JET767**
[from overseas 10081:JET767]	
Easylink	**19011170**
The Source	**BDE566**
Telex	**265871 MONREF G**
quote reference 81:JET767	

The People

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

The Guarantee

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

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

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

■■■ ISSN 0268-6066 ■■■

Free Small Ads

Apricot Sales

■■■ APRICOT PC — 512K RAM, two 720KB floppies. ACT carry cases, disks, leads. £600 or offers. 0704 25744.

■■■ APRICOT Xi — 10MB hard disk, 512K. MS-DOS 3.1 plus Basic, SuperWriter, SuperPlanner, Communications (including Prestel), Assembler, C, graphics, etc. Home use only and immaculate. £795. Apricot MODEM also available. Phone Tony on 01-948 1603.

■■■ APRICOT PORTABLE: 512K, colour, speech, manuals and plenty of software, boxed as new — £365 ono. Also many other odds 'n' sods. Call Steve Howard, 0689 57433.

■■■ APRICOT MAINFRAME — two IBM-compatible Xen-i workstations, telephone on keyboard, tape streamer, mouse. £9,000 or reasonable offer. Mr Antoniou, 01-444 2607 (office hours).

■■■ APRICOT F1 with 256K, 12in BMC monitor, SuperCalc, SuperWriter etc, The Last One program generator. Available with or without printer — Epson LX-86 or Panasonic KX-P1091 (both have NLQ). All with original packing, has had minimal home use only. Best offer accepted. Call Robert Bailey, 01-979 1953.

Apricot Add-on Sales

■■■ DATA WARDEN tape streamer and five tapes — suitable for PC/Xi, hardly used (less than 12 months old). £700 plus VAT. Contact Mark Buckley, 0865 512168 (work) or 0865 882604 (evenings/weekends).

■■■ APRICOT PC/Xi SOFTWARE AND ACCESSORIES — Lotus 1-2-3, dBASE II, padded Carry Bag, Dust Covers, parallel Printer Cable. Software unused, accessories in perfect condition; all at bargain prices. Phone Mike on 061-428 5975.

■■■ CLEAROUT: MD10 (add-on 10MB hard disk with power supply and controller card) — £275 ono. 10in COLOUR MONITOR — £135 ono. 256KB EXPANSION RAM card — £55. Also many other odds 'n' sods. Steve Howard, 0689 57433.

■■■ FOR THE PC/Xi — FD5A1 5¼in floppy disk drive, interface, software: £260. CBASIC compiler (MS-DOS) £150. CBASIC compiler (CP/M-86) £60. CP/M-86 (five disks) £75. QUICKCODE and dBASE II £100. All little used. Offers to T Dawson, 0292 77588 (evenings) or 0292 79800 ext 2682 (office hours).

Other Sales

■■■ CLEAROUT: OKI MICROLINE 192+ dot matrix printer (80-col, serial and parallel, friction and tractor feeds) — boxed as new, £155. SIEMENS INKJET PRINTER (80-col, serial and parallel, friction and tractor feeds) — £175. CASE 400/213 multi-speed modem (300/300, 1200/1200, 1200/75) — £135. EPSON CX21 ACOUSTIC COUPLER (300bps, rechargeable) — £45. TANDY 200 laptop computer with built-in modem and software — £485. SERIAL T' SWITCH — £20. COLOUR MONITOR 14in — working, but may need recalibration: service manual supplied — £55. 2764 EPROMS, used but erased — £1 each. All prices ono. Also many other odds 'n' sods. Steve Howard, 0689 57433.

■■■ AMSTRAD DMP4000 132-column printer — still under warranty. Also seven Epson MX-80 RIBBONS. Best realistic offer secures any item. Call 06512 2667.

■■■ WATANABE WX4675 six-pen A3 flatbed plotter, little used. £450 plus VAT. Call B Stanier, Demic Scientific, 0740 20818 any time.

■■■ IBM CLONES supplied — PCs and ATs, 8-12MHz; prices from £750. Martin Filmer Computing Services. 01-761 0425.

Help

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

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

Wants

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

■■■ WANTED — Keypad for F2 (even a dead one will do). Robert Clews, Flat 6 Lindsworth School, Monyhull Hall Road, Kings Norton, Birmingham B30 3QA.

■■■ WANTED — dead/damaged/cheap Xis or PCs. Phone Derek on 01-650 4900.

■■■ APRICOT XEN (non-IBM-compatible) hard disk system. Also wanted — a good C INTERPRETER (to avoid having to link after every re-edit) or Apricot CODEVIEW debugger; and a TELETEXT TUNER/READER that can be connected to Apricots (is there such a thing?). Call Tony Bridges, 01-948 1603 any time.

■■■ F10 10MB hard disk system (with manuals if possible). Contact R Noakes, 051-336 6898.

■■■ IBM-COMPATIBILITY FOR F SERIES — either a fully-effective IBM emulator or a patch permitting the IBM version of Microsoft Word 3.0 to run. Also wanted — a WORLD MAP or GLOBAL MAPPING PROGRAM. I have scores of home-grown GW-BASIC programs to sell, exchange or share — including full documentation and sample programs for the infra-red Mouse. Contact Bob Mandel, Lewis & Clark College, Box 96, Portland, OR 97219, USA.

■■■ SECONDHAND OR EX-DEMO HARD DISK — internal or external, 10 or 20MB, for PC. Cannot afford £800+ for a new one. Write to T Orekoya, FAO (FIRI), Via della Terme di Caracalla, 00100 Rome, Italy.

■■■ MMS INSTRUCTION MANUALS AND INTERFACE BOARD required for MMS hard disk unit and tape streamer. Willing to pay for help (even telephone assistance) in setting this up. Call Roy Roberts, 0923 775381 (office) or 0494 775088 (evenings).

■■■ GOOD DEBUGGER wanted for PC/Xi. I hate using Apricot's; can anyone help before I'm forced to write my own? (Preferably with symbols and/or C source display). Phone 01-948 1603 any time.

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

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

■■■ WORD PROCESSOR wanted for Apricot PC with single-sided disks — MS-WORD 3.0 preferred. Also ANY OTHER SOFTWARE. Phone 01-908 4631. ◆

Government changes IT R&D

The 13 January **White Paper**, which redefines the **DTI's** role as 'The Department for Enterprise', exemplifies the Conservative's attitudes to a tee — and some people in the information technology business are slightly worried about its effects. The Government expresses concern that industry-funded R&D as a proportion of GDP is lower than for our competitors: opines that innovations close to marketability should not be funded by Government: and rejects most of the conclusions of the **IT 86** committee it established — notably a new five-year £1 billion IT research programme to follow Alvey, with Government putting up £425 million of that.

In fact, the Government stomps on the idea of specific support for information technology applications, one of the planks of IT 86's conclusions. Instead the White Paper has the UK contributing £200 million to the second phase of the EEC's ESPRIT programme, plus another £84 million from UK public funds to complement a similar essay into basic R&D in Britain.

Most of the grant-aid schemes for encouraging the take-up of IT are terminated, too — as are the product-development Microelectronics Industry Support Programme and the Software Products Scheme.

Instead there will be less cash overall — and more selective use of the money, particularly focusing on collaborative ventures and development-assistance areas. The DTI is obviously keen to pass the buck for R&D to investors, which is probably no bad thing; and it appears to recognise the importance of selective pump-priming. It remains to be seen whether the 'initiative' in the DTI's somewhat grandiose renaming comes primarily from Lord Young's ability to cut the PSBR rather than stoking the fires of industry.

Ad lib

US computer-industry advertising totalled a whopping $2 billion in 1987. **IBM** accounted for nearly a quarter — it spent **$467** million on ads during the year.

Apricots on the menu

XI Data Systems of Leeds seems to have carved out (as it were) a catering systems market in the armed forces. Already responsible for a large Army Catering Corps order for Apricot systems running its software, XI has now picked up an **RAF Catering Directorate** contract for 80 Xens. XI custom-wrote this system with RAF staff.

BT profits up

British Telecom announced a profit increase of more than 11 per cent for the six months to September 1987, apparently due almost entirely to increased demand for the groups services.

After a year's worth of bad press, BT is obviously over the moon about the figures; but in PR terms, a gain in profits does put the group under severe pressure to improve services this year.

The long-awaited introduction of itemised bills will be a welcome addition to the service — London subscribers should be getting them soon, at no extra charge, following a successful pilot scheme in the City. They will detail calls of 10 units or more.

Microlink goodies

Microlink, the cheap gateway to Telecom Gold, is wooing new subscribers with an impressive range of goodies — in particular, a selection of ready-to-go comms packages consisting of modems, software and one month's free use of Microlink and Gold. The modems are from **Pace**, and excellent value they are too: the entry-level Microlink offer provides 300 or 1200/75bps plus comms software for **£99**, or you could pay a modest **£169** for 1200bps Hayes-compatible comms. The first systems are available for the BBC micro as well as IBM-compatibles; Microlink is on **0625 878888**.

Word processor spoof

Just in case anyone was fooled, the 'Word Processor Evaluation' by 'West Engineering' included with several of last month's computer magazines is a promotion for **Microsoft's** Word 4. In the back of the report is a tear-out coupon for a £15 evaluation kit containing a working copy of the word processor and some documentation. So *that's* why Word 4 beats WordPerfect on just about every criterion ...

E-mail with a conscience

Poptel is neither a breakfast cereal nor a dial-a-record service: it's 'the alternative access to the new communications technology'. Poptel is an electronic mail service aimed specifically at non-commercial users such as trade unions, development and aid agencies, educational and community groups and campaigning organisations — but it is of course also available to individual users. Among Poptel's wide range of services are the usual e-mail facilities (including Telex), bulletin board facilities, access to news services (UPI, AP, and TASS among them), teleconferencing, and some specialised on-line databases.

Poptel users have access to the international Geonet network — over 25 host computers, around 10,000 subscribers. Poptel is committed to ensuring users freedom from interference by large commercial organisations and to demanding from suppliers such as Geonet the kind of facilities non-commercial users need; one result is a modest fee structure — registration costs £25 for the first mailbox and while there's a monthly subscription charge of £10 with usage charges of 8p per minute, you do get 60 minutes per month free.

Poptel is not a non-profit organisation; but the margins are obviously nowhere near those of conventional commercial services — and of course the motivation is totally different. Details and subscription forms are available from **01-249 2948**. ◆

IBM on Apricot?

We finally tracked down **Insignia Solutions**, purveyor of a product called **SoftPC** which emulates an IBM PC on other machines. Insignia told us that the software is designed primarily to provide IBM-compatibility on non-Intel machines, mostly Motorola-based systems and RISC computers; they reckon an Apricot implementation has never been produced and won't be. Ah well ...