.EXE

JUNE 1988 VOL 3 ISSUE 1

£3.20

The Software Develop... ...ine

IN THIS ISSUE
The UNIX Special Report

Design your own Fonts for the EGA and VGA.

The Software Development Life Cycle – We look at Liveware.

Ray Duncan on Alternative 386 Operating Systems.

Lotus Developer Tools: Write your own add-ins for 1-2-3.

Clear: Produce Flowcharts from dBase Source Code.

Why Write Programs in a Database Language?

Converting a Multi-tasking DOS Application to OS/2.

10 HOURS VIDEO PLUS 365 PAGE STUDY BOOK

The Complete C Video Course

Now you can learn C the easy way!

Become a C expert in the comfort of your own home with our ten VHS tapes and a beautiful 365-page workbook.

Teach yourself all you need to know about the C language with this unique video course.

- ***Comprehensive easy to follow 365-page workbook***
- ***Floppy disk containing all the example programs***
- ***C Programming standards and guidelines are covered with examples.***
- ***Not compiler or hardware specific***
- ***ANSI standards discussed***
- ***Additional work books available at £19.95 each***
- ***36 Lessons to cover topics at all levels of programming skill***

ZORTECH

YES!
Rush me the Complete C Video Course

☐ *VIDEO COURSE £129.95** ☐ *EXTRA BOOKS £19.95**

VISA, ACCESS OR CHEQUE ACCEPTED

Name ....
Address ....
.... *Phone* ....
VISA or ACCESS No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Exp. Date ....

To: ZORTECH LTD. 106-108 Powis Street, LONDON SE18 6LU.
Tel: 01-316 7777. Fax: 01-316 4138.

CALL THE ORDER HOTLINE 01-316 7777 NOW! CIRCLE NO 1

**Prices do not include shipping or VAT*

Editorial: We welcome articles from prospective authors and invite submissions from readers. Please write or phone for our contributors notes and rates of payment.

Subscriptions: .EXE is available only on subscription. An annual Subscription for 11 issues costs £35 in the UK and £45 for overseas. .EXE is published each month excluding December. It is an independent publication and is not affiliated in any way to any manufacturer or vendor of hardware, software or services. .EXE Magazine is published by Process Communications Ltd. 10 Barley Mow Passage, Chiswick, London W4 4PH, England. Telephone: 01 994 6477. ISSN 0268-6872

Editor
Robert Schifreen

Advertising Manager
Jonathon Howell

Advertising Executives
Steve Duck
Oliver Hill

Subscriptions & Promotions
Helena Adams

Design
Katey M.

Print
Chase Web Offset

Edlin

You may have caught a glimpse of the .EXE offices on the television news recently. The Beeb came to film an interview with yours truly, following my acquittal in the House of Lords after hacking into British Telecom's Prestel databases.

British Telecom had prosecuted me, and a friend, for hacking into Prestel late in 1984 and using various passwords. Among the passwords I used was that of HRH The Duke of Edinburgh. Although the Lords' decision means that there is now no legal precedent under which UK hackers can be convicted, the action has cleared the way for Parliament to pass a specific Computer Crime Act. It will be a while in coming, but its arrival will hopefully clear up the mess once and for all.

What I set out to prove 4 years ago was the incompetence of those whose job was to keep the Prestel hardware and software under control. Over a period of around 10 months, I discovered the System Manager id and password, because Prestel staff had left this information lying around. Armed with these numbers I could, and did, instruct the Prestel system to divulge the identity number and password of any subscriber or information provider on the system. Admittedly, the system to which I had such privileged access was only a development site, but Prestel programmers had used real, live copies of data on this test site, so the files contained real passwords and not dummy ones.

The story of how I managed to hack Prestel has been told many times by many people. Most accounts of the events contain at least a couple of mistakes, so let me explain what happened. Early in 1984, I was testing some Prestel terminal software at my office. The Prestel system was asking me for an identity number, so I typed the keys that were nearest to my fingers at the time. I used an identity number of 2222222222, and a password of 1234. I was surprised by 2 things. Firstly, the numbers that I tried let me in to the system. Second, the account was an internal BT account and, therefore, I had access to information that only BT staff should have been able to see. Among the information was the telephone numbers of many Prestel test computers, which I regularly called over the next few months.

Although I called the test computers regularly, I could not normally get into these systems as I did not have any passwords. However, in October 1984, I called up one of them and was amazed to find a password written on the sign-on screen. It was later admitted in court that Prestel programmers had put the numbers there as "an aide memoire". I typed in the numbers, and the system replied "Good evening, System Manager". The rest, as they say, is history. As I continued in my explorations of the System, little did I realise that my home telephone was (and may still be) tapped and that, in March 1985, 7 assorted police officers and BT investigators would arrive at my house and take me away.

Elsewhere in this issue of the Magazine, I've printed the full text of the Lords' judgement. This document is the end result of 38 months of argument, a jury, 3 judges, the Lord Chief Justice, 7 Law Lords, 4 solicitors, 3 barristers, 3 QCs and around 3m of public money. It fully describes the current state of English law as it applies to computer access, and should make interesting reading for anyone who is charged with keeping a network secure.

Meanwhile, I should state now that I certainly don't regret what I did. Few people realised, back in 1983, exactly how powerful the computer password was. Since my case hit the headlines, peoples attitudes have changed slightly. Hacking into computers may currently be perfectly legal in England and Wales, but the reasons why this state of affairs exists is at last very clear. A new law is needed, and it is needed very soon. Otherwise, precedents more important than mine will be set. For example, if using someone else's password is legal, what's to stop me borrowing someone else's cashpoint card, finding their PIN number and looking at their bank balance? Of course, I couldn't actually withdraw money, as this would be fraud or theft, but simply using their card to access information is not theft and, once you've read the text of the judgement, you'll realise that it can't be forgery either.

The floodgates may now be wide open, but at least we are finally in a position which will see the drafting of a Computer Crime Act. Let's hope that those on the drafting committee understand their subject.

.EXE

DISK COPYING SERVICE

We can copy files to and from 400 disk formats including CP/M, CP/M-86, MS- DOS, PC-DOS, ISIS, APPLE, SIRIUS, BBC, TORCH, APRICOT, HP150, TRSDOS, DEC RT- 11, IBM BEF, ATARI520, AMSTRAD.

Our charge is £10.00 + disk + VAT with discounts on small quantities and disks are normally despatched within 24hrs of receipt.

For more information call us.

PROGRAMMING TOOLS

Ada Compilers	Algol Compilers
Assemblers & Libs	AWK
Basic Compilers	Basic Interpreters
Basic Utilities	Basic Libraries
BCPL Compilers	C Compilers
C Interpreters	C Libraries
C Utilities	Cobol Compilers
Comms.Libraries	Cross Assemblers
Database Libs.	Debuggers
Dis-assemblers	Editors
Engineers Libs.	Expert Systems
Forth	Fortran Compilers
Fortran Libraries	Graphics Libraries
Icon	Linkers
Lisp	Modula-2
Nial Interpreters	OPS 5
Pascal Compilers	Pascal Libraries
Prolog	Rexx
Risc	Screen Libraries
Smalltalk	Snobol

We stock many items for which there is no space in these advertisements.

C LIBRARIES

Microsoft OS/2 programmers toolkits are now in stock.

DATABASE

Btrieve v4.11	MS-DOS	£160
Btrieve/Multi user	MS-DOS	£380
Btrieve/Network `	MS-DOS	£380
XQL	PC-DOS	£510
CBTREE	(source any C)	£100
C-Index/Plus	(source any C)	£175
C-ISAM (L,MS)	MS-DOS	£210
C-to-dBase (source CI)	MS-DOS	£120
C-tree	(source any C)	£210
R-tree	MS-DOS	£155
db-VISTA v2.x (most C's)	MS-DOS	£145
db-VISTA v2.x (with sce)	MS-DOS	£350
db-VISTA v2.x multi-us.	MS-DOS	£350
db-QUERY v1.x single-us.	MS-DOS	£145
db-QUERY v1.x sngl.(sce)	MS-DOS	£350
Lattice dBCII (L)	MS-DOS	£160
Lattice dBCIII (L,MS)	MS-DOS	£160
SftFcus Btree&Isam	(sce any C)	£ 80
Virtual Memory File Man.	PC-DOS	£150

GRAPHICS

Enhanced Graphics Tlkt	PC-DOS	£145
Essential Graphics	PC-DOS	£175
GFX Graphics	PC-DOS	£ 50
GraphiC v4.1(CI,L,DS,MS)	PC-DOS	£270
GSS Kernel	PC-DOS	£305
GSS Lattice Binding	PC-DOS	£110
GSS Graph Dev Toolkit	PC-DOS	£305
GEM Prog.Toolkit (L)	PC-DOS	£405
GSX Prog.Toolkit (DR)	PC-DOS	£270
HALO v2.26(AZ,CI,L,MS,T)	PC-DOS	£165
MetaWINDOWS (CI,L,MS,T)	PC-DOS	£105
MetaWINDOWS Plus	PC-DOS	£150
Turbo WINDOWS/C (Turbo)	PC-DOS	£ 55

SCREEN & WINDOWS

Blaise View Mngr.(s'ce)	PC-DOS	£245
Curses Screen Mngr. (L)	PC-DOS	£120
Entelekon Windows(s'ce)	PC-DOS	£ 99
Multi-windows (MS,L)	PC-DOS	£220
Panel Plus (source)	PC-DOS	£270
Vitamin C (source)	PC-DOS	£130
Windows for Data (most)	PC-DOS	£200
Windows for C (most C)	PC-DOS	£135
Greenleaf Datawindows	MS-DOS	£135
C-Scape (MS,L,TC)	MS-DOS	£

GENERAL FUNCTIONS

C-Worthy Int.Lib (L,MS)	MS-DOS	£180
Greenleaf (source)	PC-DOS	£ 85
G'leaf C Sampler(TC,QC)	PC-DOS	£ 65
Smorgasbord (source)	PC-DOS	£130
Blaise Tools Plus (sce)	PC-DOS	£ 80
Blaise Tools Plus/5.0	PC-DOS	£ 80
Blaise Turbo C Tools	PC-DOS	£ 80
Blaise Light Tools (ZOR)	PC-DOS	£ 65
ESI Utility Lib(source)	PC-DOS	£105
Entelekon Funct(source)	PC-DOS	£ 95
Phoenix Pforce (source)	PC-DOS	£205
Phoenix Pforce C++ (sce)	PC-DOS	£220
Pro-C (most C)	PC-DOS	£720
Security Library object	MS-DOS	£ 95

COMMS LIBRARIES

Blaise Asynch (source)	PC-DOS	£115
Essential Comms(L,MS,T)	PC-DOS	£110
Greenleaf Comms(source)	PC-DOS	£ 85
Multi Comms (L,MS)	PC-DOS	£120

SCIENTIFIC LIBRARIES

Wiley Scientif.Lib.v2	ANY C	£145
Mathpak 87 (L,MS)	MS-DOS	£ 80

PROGRAMMERS UTILITIES

PC-Lint v2.15	MS-DOS	£ 80
Pre-C (Phoenix Lint)	MS-DOS	£175
C-Documentor	PC-DOS	£195
C-Scan	PC-DOS	£195
C Toolset	MS-DOS	£ 90
Lattice Cross Ref.	MS-DOS	£ 40
Lattice Text Utilities	MS-DOS	£ 85

REAL TIME & MULTI-TASKING TOOLS

Csharp (CI,L,MS) s'ce	MS-DOS	£445
Concurrent C (PC/MPX) sce	MS-DOS	£ 60
Multi C (L,MS,CI)	PC-DOS	£110
Multiforms (L,MS)	PC-DOS	£125
Op.Sys.T'box (MS) s'ce	PC-DOS	£ 65
Timeslicer v3.01 (L)	PC-DOS	£160
Timeslicer v4.01M (MS)	PC-DOS	£160
Over-C (L,MS)	PC-DOS	£225

THE C LANGUAGE

The new Watcom compiler produces very fast code with excellent debugging facilities.
MSC v5.1 for OS/2 should be here by now. OS/2 programmers toolkit is here.

C COMPILERS

Aztec C Personal 1.06D	CP/M-80	£115
Aztec Commercial 1.06E	CP/M-80	£190
BDS C 1.60	Z80 + CP/M-80	£ 65
Eco-C v3.48	Z80 + CP/M-80	£ 45
Hi-Tech C	Z80 + CP/M-80	£ 99
Mix C	Z80 + CP/M-80	£ 35
Toolworks C/80 v3.2	CP/M-80	£ 45
Advantage C++	MS-DOS	£405
Aztec C86 Professional	MS-DOS	£110
Aztec C86 Developer	MS-DOS	£165
Aztec C86 Commercial	MS-DOS	£250
CI C86 Plus	MS-DOS	£290
CI Optimizing C86	MS-DOS	£230
De Smet C Starter	MS-DOS	£ 85
De Smet C Programmer	MS-DOS	£125
De Smet C Enhanced	MS-DOS	£165
De Smet C Professional	MS-DOS	£200
ECO-C88 V4.0	MS-DOS	£ 70
HIGH C (Metaware) v1.4	MS-DOS	£380
HIGH C 386 v1.3	MS-DOS	£545
Hi-Tech C	MS-DOS	£115
Lattice C v3.22	MS-DOS	£240
Mark Williams LETS C v4	MS-DOS	£ 60
Microsoft C v5.0	MS-DOS	£260
Microsoft Quick C v1.0	PC-DOS	£ 60
MIX C	MS-DOS	£ 20
MIX Power C	PC-DOS	£ 30
Turbo C v1.5	PC-DOS	£ 50
Toolworks C v3.2	MS-DOS	£ 45
Watcom Express C	PC-DOS	£ 95
Watcom Exp C v6 Opt Comp	PC-DOS	£220
Zortech C v2	PC-DOS	£ 29
Zortech C Power Pack v2	PC-DOS	£ 59
Zortech C Optim.(for v2)	PC-DOS	£ 29
Aztec C86 Developer	CP/M-86	£240
Aztec C86 Personal	CP/M-86	£140
CI Optimizing C86	CP/M-86	£270
De Smet C Starter	CP/M-86	£ 85
De Smet C Programmer	CP/M-86	£125
De Smet C Enhanced	CP/M-86	£165
Hi-Tech C	CP/M-86	£115
Lattice C v1.04	CP/M-86	£260
HIGH C (Metaware)	Flex OS 286	£380
Aztec C65 V1.05	Apple DOS	£180
Aztec C65 V3.20	Apple PRO-DOS	£230
Aztec C68 Commercial	MACINTOSH	£250
Aztec C68 Developer	MACINTOSH	£165
Aztec C68 Commercial	AMIGA	£250
Aztec C68 Developer	AMIGA	£165
Hi-Tech C	ATARI	£ 99
Lattice C	ATARI	£ 85
Mark Williams C v2	ATARI	£110
Laser C	ATARI	£150

C CROSS COMPILERS

We supply 2500AD, Aztec, Lattice, IAR and Hi-Tech Cross Compilers hosted on MS-DOS and targeted on Z80, 8085, 6502, 6801, 68HC11, 6301, 6809, 8051, 68000 & 68020. Please call for information or advice.

C INTERPRETERS

Latest v3.00 of C-terp has improved debugging facilities.

C-terp V3.0X	PC-DOS	£165
Interactive C	PC-DOS	£195
Introducing C	PC-DOS	£ 85
Living C Plus	PC-DOS	£135
Instant-C v2.2	MS-DOS	£380
RUN/C	MS-DOS	£ 60
RUN/C Professional	MS-DOS	£110

PROLOG LANGUAGE

Arity Standard	PC-DOS	£ 79
Arity Prolog int. v5	PC-DOS	£225
Arity Int. & Comp. v5	PC-DOS	£495
LPA PROLOG Prof.Int.v1.5	MS-DOS	£265
LPA Micro-PROLOG v3.1	MS-DOS	£ 75
Prolog-86 plus v1.0	MS-DOS	£195
Prolog-2 Personal	PC-DOS	£145
Prolog-2 Programmer	PC-DOS	£495
Prolog-2 Professional	PC-DOS	£995
Turbo-Prolog v1.1	PC-DOS	£ 59
ADA Educ.Prolog	MS-DOS	£ 26
ADA FS Prolog	MS-DOS	£ 45
ADA VMI Prolog	MS-DOS	£ 70
ADA VML Prolog	MS-DOS	£160
Prolog-1 v2.2	CP/M-86	£299
LPA Micro-Prolog v3.1	CP/M-86	£ 75
LPA Micro-prolog v3.1	CP/M-80	£ 60
Prolog-1 v2.2	CP/M-80	£225

PRICES & DELIVERY

Prices do not include VAT or other local taxes but do include delivery in UK and Europe. Please check prices at time of order, ads are prepared some weeks before publication.

This page lists some of our products. Call us for a complete pricelist.

Order by phone with your credit card.

FORTRAN COMPILERS

Microsoft Fortran for OS/2 is in stock now and new Lahey for the 286.

Lahey F77L v2.21	MS-DOS	£330
Lahey Personal Fort-77	PC-DOS	£ 70
RM/FORTRAN 77 v2.4	MS-DOS	£450
MS-FORTRAN 77 v4	MS-DOS	£260
Pro Fortran 77 v1.22	MS-DOS	£240
Pro Fortran for GEM	MS-DOS	£ 95
Utah Fortran	MS-DOS	£ 35
FS-Fort.(CGA & Herc)	MS-DOS	£ 35
Pro Fortran v2.1	CP/M-86	£240
Pro Fortran v1.25	CP/M-80	£240
Nevada Fortran v3.3	CP/M-80	£ 35
Pro Fortran 77	ATARI 520ST	£ 95

We have Fortran Libraries in stock.

GREY MATTER
4 Prigg Meadow, Ashburton, Devon TQ13 7DF.
TEL. (0364) 53499

GREY MATTER
4 Prigg Meadow, Ashburton, Devon TQ13 7DF.
TEL. (0364) 53499

GREY MATTER
4 Prigg Meadow, Ashburton, Devon TQ13 7DF.
TEL. (0364) 53499

CIRCLE NO 2

Letters

The magazine welcomes your comments, suggestions and opinions. Write to 10 Barley Mow Passage, Chiswick, London W4 4PH or send Telecom Gold message to 72:MAG90141. We are also on CIX as RSCHIFREEN, on BIX as HEXMANIAC and on Microsoft DIAL (in the UK) as EXEMAG. You can dictate a letter or make a point (no answering machine, guaranteed) during office hours on 01 994 6477 x2212. Please mark your letter accordingly if you do not wish it to appear in print.

Dear sir,

Your reviewer, Roger Dalton, made some rather misguided comments and observations about the Complete C Video Course in the April issue and also failed to answer some key questions regarding the product.

Firstly, Carol Weaver is a charming young lady, a professional and proficient instructor with over 5 years experience in teaching C at the highest corporate level. To suggest that her voice would 'infuriate' the student sounds a bit queer to me – perhaps Roger would have preferred a man's voice?

Roger is only the second unhappy customer on our books out of all the many hundreds of course users. References are available from many of the large corporate buyers, universities and individuals, many of whom have adopted the course as their standard in-house C course. No mention was made of any other comparable products, which all incidentally retail at over £1200, and which would surely have given the video course much higher points from the "value for money" perspective.

The review said that the course "doesn't even attempt to exploit the amazing power of the medium". Perhaps the reviewer would have liked to have seen some flash 3-D computer graphics animation – or maybe a well-known presenter from Tomorrow's World? We purposely avoided using any such gimmicks. Rather, we decided to produce an "open university" style course – the technique and format being already well proven.

Roger Dalton complains that the PC next to Carol Weaver never gets turned on. Who cares? What's the big deal? It was only a prop. We obviously shot the most important thing, which is the computer screen. Did he actually want to see Carol's fingers 'faking it' on the keyboard? He also says that it's criminal that, having explained how each program would work, we didn't show the same programs actually running. This would have been an utter waste of teaching time....

John Haggins
Zortech Ltd

Robert Schifreen replies: *Perhaps someone else in the training business would care to comment?*

Dear Robert,

I suspect that Bill Hedley (Letters, .EXE , March 88) is less interested in the pros and cons of C than in advertising his own machismo as one who does "real programming". (Does this mean that what the rest of us have been doing all these years is somehow unreal? And isn't the inescapable conclusion of this that machine code is even more "real" than assembler?) Anyway, never mind. Rather than slag off Bill Hedley (who is probably happy in his blinkered world), I wish to point out that this discussion has been started off from the wrong end.

A programming language is simply a notation and, as such, can be used well or badly. Though this may be news to some people, the agent that produces well structured, elegant and readable programs is the programmer, not the language itself. As proof of this, you may look in the back pages of a well known computer magazine and see oodles of code in many languages which can best be described as something off the Gorgon's head. On the other hand, you may look in K&R's The C Programming Language and see some extremely elegant code. The fact that Kernighan, this time with Plauger, has produced the same uncluttered elegance in Software Tools, using both Ratfor and Pascal, lends credence to my thesis.

But let's face it. Reading code in any language is time consuming, tedious and, like reading the manual, we only do it when all else fails. This is why we make such a fuss about structured programming and decent documentation: with any luck we may only need peruse the blurb that comes with the program or, if pushed, the comments alongside the code. If it really comes to the crunch, good structure and modularisation will mean that we only have to read the bits of code we're interested in. (We won't enjoy even that, but the job will at least have been reduced to a minimum.)

So we should stop being silly about "good" and "bad" programming languages and put the emphasis where it really belongs – with the programmer. A "real" programmer is one who uses the language that is right for the job and designs and documents the program properly.

Perhaps there really are programmers who are scared of English, as Bill Hedley says. So what? You wouldn't employ a chauffeur who's scared of traffic, or a life guard who's scared of water. A programmer who can't write English (or whatever the vernacular of the country happens to be) is simply not a competent programmer and you shouldn't try to construct a programming philosophy out of such an aberrant species.

Bob Knight
London, NW4

Robert Schifreen replies: *What's a Gorgon anyway?*

Dear Sir,

I am writing to correct a misleading comment made about APL People in April's .EXE Magazine. It was stated that we used to find jobs for APL programmers, before (it was implied) seeing the light and deciding to specialise in C and Unix.

I would like to set the record straight for any of your readers that use APL. We still operate a service for them, and embracing C is part of our expansion plan, brought about as more and more of our APL clients started to ask us if we had anyone on our books with both APL and C experience, or just with C alone.

I trust that you will inform your readers of this fact. We have no intention of severing our APL connections and, needless to say, the number of people we place in APL jobs each year far exceeds the number that was mentioned.

Jill Moss
APL People Ltd

EXE

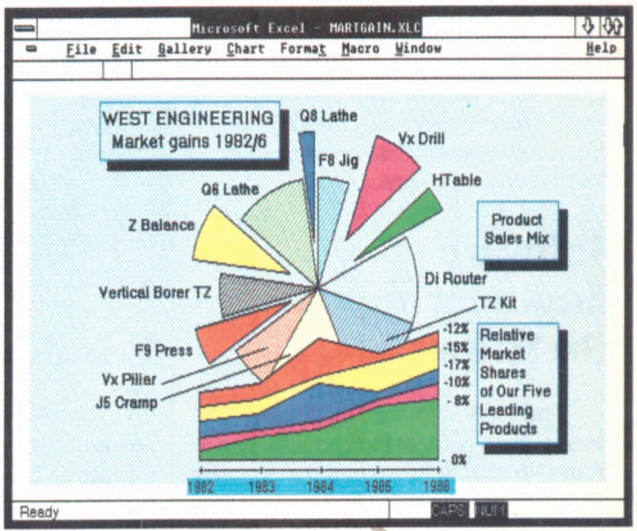
Have you ever wondered what your PC will be doing in 1991?

There's a new generation of software coming. It's so friendly, so efficient and so powerful that we'll all be using it soon. Will your existing personal computers cope?

Almost certainly not.

We asked Microsoft, the software pacesetters, to tell us about the products they are building for the 1990s. They used their Excel spreadsheet as an example of the new Windows-based superprograms.

Above all, Excel makes the most of high resolution colour graphics, preferably 640 x 480. That's OK, NEC desktops have multi-mode graphics, MultiSync monitor and mouse port too.

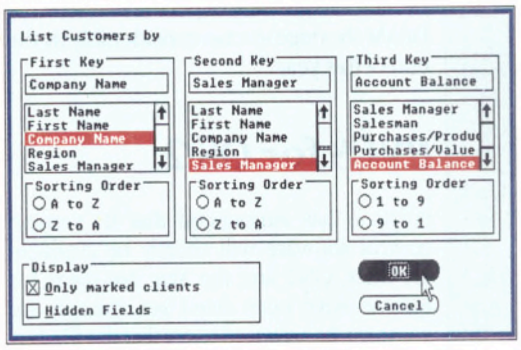


Multiple consolidations and hot links between programs need fast hard disks. Good: our entry level 40msec drives have speed to spare (useful, as Excel can actually use disk space as virtual memory). For power users, our 66MB, 23msec drive takes it to the limit.

Looking to the future, our PowerMate 386 with 130MB ESDI hard disk is eagerly awaiting SQL Server, a key element in Microsoft's LAN Manager strategy. The interleaved 32-bit memory in the PowerMate 386 will make quick work of multi-user updates, sorts, and complex enquiries.

NEC's goal is to bring higher performance PC systems within the reach of all departmental budgets. We do this by making standard many of the enhancements that others sell as low volume, high cost "options".

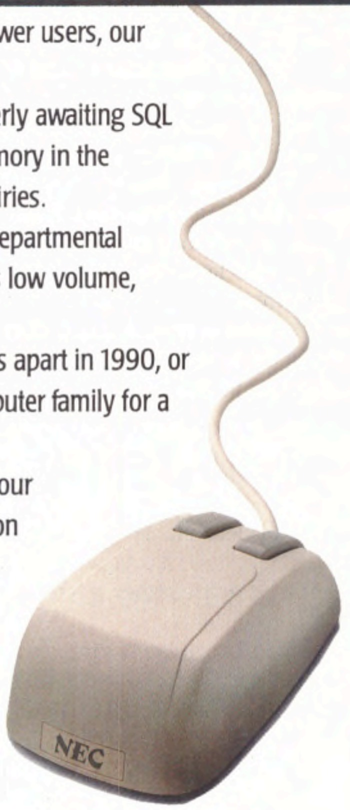
But there's no point in buying a computer to cope with 1991's software if it falls apart in 1990, or even 1989. So you'll be reassured to hear that NEC, and only NEC, guarantees its computer family for a full three years.



```
List Customers by

First Key              Second Key             Third Key
Company Name           Sales Manager          Account Balance

Last Name              Last Name              Sales Manager
First Name             First Name             Salesman
Company Name           Company Name           Purchases/Produc
Region                 Region                 Purchases/Value
Sales Manager          Sales Manager          Account Balance

Sorting Order          Sorting Order          Sorting Order
○ A to Z               ○ A to Z               ○ 1 to 9
○ Z to A               ○ Z to A               ○ 9 to 1

Display
☒ Only marked clients                          OK
☐ Hidden Fields                                Cancel
```

You've put the new software in your plans for the future, so put NEC on your shortlist today.

Call our hotline on 01-924 1244, or mail your business card to John Parkinson at NEC Computer Division, 1 Victoria Road, London W3 6UL.



Think Ahead. Shortlist NEC.

C&C
Computers and Communications

CIRCLE NO 3

NEC

More on OS/3

Microsoft has stated in the US that a version of OS/2 for the 80386 will be made available to developers towards the end of next year. Latest name for the system is OS/2-386 and will, as mentioned in .EXE last month, incorporate the new filing system currently being developed by Gordon Letwin. The filing system will be installable, and can be customized by OEMs. This will allow systems to be developed that use file names longer than the standard 11 characters (8 for the name and 3 for the extension). There's also talk of adding multiprocessor support to add performance to OS/2 based file servers.

DOS Bug

News has surfaced of a bug in all versions of MS-DOS 3.30. Apparently, the parser that interprets CONFIG.SYS gets confused over DRIVPARM, so you can't use this command at all. This means that if you change one of the standard floppy drives (A: or B:) to a type not normally recognised by the controller, you can't format it because you can't define its type. Once the disk is formatted, though (in aother drive), it can be read from and written to because the BIOS will work out its format automatically. The bug only affects built-in drives, as extra drives (beyond C:) are handled by DRIVER.SYS instead.

Microsoft has issued a fix to the problem, in the form of a complete new DOS. The disks went out to all OEMs in March. If your DOS came with your Taiwanese clone, therefore, the official way of upgrading is to contact the manufacturer. Some dealers may have the upgrade, though.

According to Microsoft's technical people, the bug was in the parser, which insisted on looking for backslashes between DRIVPARM parameters though the DOS manuals say you should use forward slashes. I have yet to locate the 1-byte change that fixed the bug, though I can report that changing a DRIVPARM command so that it uses forward slashes still results in an 'Unrecognised command in CONFIG.SYS' message.

Software Engineering On Show

The 1988 Software Engineering Conference takes place at Liverpool University from 11th to 15th July. A number of papers have already been submitted, covering users' experiences in the field, and there will also be debates on the role of formal methods for software maintenance, specification testing and so on.

There will be tutorial sessions on the first day, covering 'software prototyping techniques and management', quality assurance and the STARTS initiative. There will also be an opportunity on the last day to visit the NCC and its Software Tools Demonstration Centre.

More information from the IEE, who are organising the event together with the BCS. The IEE is on 01 240 1871.

C-Index+

The Software Construction Company are distributing the latest version (v3.2) of C-Index+. This package from Trio Systems is a single- and multi-user database backend, to which you add a front end written in Microsoft, Lattice or Turbo C. There's a complete sample application on the disks, to show you how things should be done, and a demonstration system (limited only by source file size) is also available. Prices are £25 for the demo or £175 for the real thing, including full source code. If you don't want the source code, it's only £69.95, which gets you object code for linking to Turbo C or QuickC files. The Software Construction Company is contactable on 0279 792566.

State of the art

Although quick to point out that the new devices won't be available for some time, IBM say they have developed silicon circuits capable of switching on or off 75 billion times a second (a 13-picosecond switching time).

4-Mbit DRAMS available

Japanese chip manufacturers have produced 4-megabit DRAMs in limited quantities. The samples have been sent to manufacturers, but there's no word yet on when the devices will be available in volume. Meanwhile, Dell computers, who switched from using SRAMS to DRAM in their new machines (see separate story) stated at its UK launch that they expect the worldwide DRAM shortage to ease considerably by the end of this year.

New 80286 Opcode

Intel has made available full documentation on the LOADALL instruction. This is an undocumented opcode that is available on the 80286 only, and was brought to public attention when a programmer was testing his disassembler on Microsoft's latest RAMDRIVE.SYS.

The instruction, which has the opcode 0Fh, 05h, loads the entire 286 register set from a block of memory at 0:0800h. All registers (including LDTR, TR, GDTR, IDTR and MSW) are loaded from memory by this instruction. The normally hidden descriptor cache registers for the ES, DS, SS, CS, TR and LDT registers are also loaded. LOADALL may be executed in either real address mode or protected mode (CPL must be 0). Any attempt to execute LOADALL at any privilege level but 0 in protected mode causes exception 13 with error code 0.

In protected mode, LOADALL can set the selector, base address, limit and access rights for a segment register. The normal protected mode protection rules can also be changed. In real address mode, the physical address, limit and access rights for a paragraph id can also be changed from the normal real mode definition.

The layout of the 105-byte data block plus other information is available from Intel in Swindon. There's also a transcript of the Intel documentation in the pc/hardware conference on Cix.

PVCS for OS/2

Polytron has announced that its version control software will shortly be available for Unix, OS/2 and the Mac. No dates for release have been fixed yet, though the company has assured users that the file formats of the new version will be identical to that used by the existing MS-DOS system. Also, the systems will be equally easy to use, as they will all incorporate the new menu-driven interface that's currently being shipped with the PC versions. Polytron are based in the US on 0101 503 645 1150, and Grey Matter are official UK distributor.

New Zortech Products

Zortech has introduced the first C++ compiler. Rather than translate C++ code into normal C source code for later compilation, the Zortech product compiles C++ source directly into . JOB files.

Zortech has also announced version 4 of its C compiler. Notable additions include a powerful graphics library and the claim that it is now fully compatible with Microsoft's Codeview debugger. It remains to be seen quite what Zortech means by using the word 'compatible', as the format of information that Microsoft's C compiler adds to an .EXE file to allow Codeview to access local variables is proprietary. Also new to Zortech's catalogue is an EMS 4.0 toolkit, that acts as a front end for the EMS driver and allows C programs to control and use LIM memory.

More from Zortech on 01 316 7777.

More PS/2 Cloning Kits

Western-Digital and Phoenix has joined forces to produce a chip set that will allow manufacturers to produce IBM PS/2 model 50 and 60 clones. The package consists of a peripheral controller, DMA controller, data/address buffers, memory controller, floppy-disk controller, async communications controller and a Paradise VGA chip set. System designers will be offered the kit together with a Phoenix BIOS and an Intel 286 for $2500. The kit is similar in function to the Chips & Technologies offering. Although benchmarks are not yet available, both systems claim to run faster than IBM's equivalent PS/2 models.

No Advantage to OS/2 OEMs

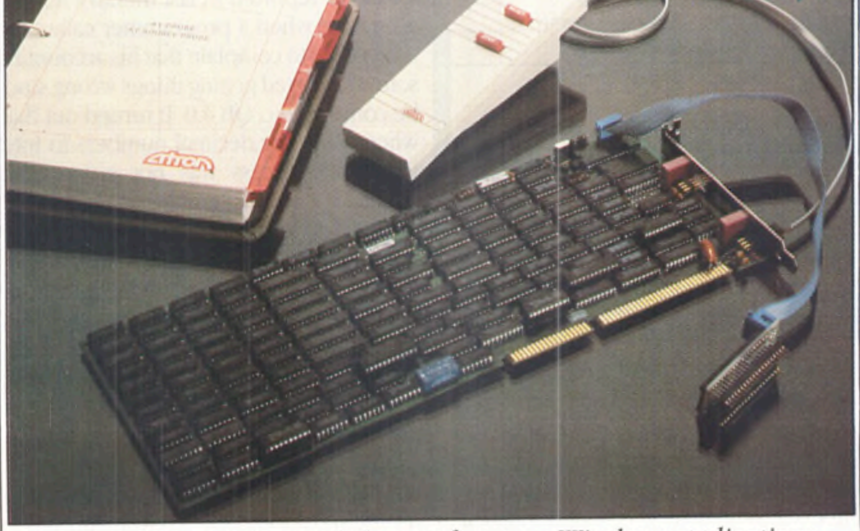
OEMs who improve the performance of OS/2 are compelled, under the terms of their license, to make the source code of their improvements available to Microsoft. This was revealed by Michael Dell, founder of Dell Computer Corp, when he attended a product launch in London last month. Launching the Dell version of OS/2 1.0 he stated that Dell engineers had rewritten the hard disk drivers for OS/2 and that this part of the code now runs at least 50% faster than IBM's version. At times, the increase is more like 300%, he said. Only a small proportion of the improvement is due to tailoring the code for Dell's drives. Most of the gains are achieved through the use of caching algorithms, and the new drivers will speed up any machine capable of running OS/2, including the PS/2 range.

Having explained to the audience just what the Dell engineers had managed to do with OS/2, it was revealed that Dell would not be allowed to keep their new code secret. Microsoft, it was confirmed, includes a clause in the OEM license that forces Dell to make the new drivers available, and Microsoft reserves the right to incorporate the amendments in future releases of the new operating system.

New Dell Hardware

Dell has announced 2 new machines in the UK. The first is a small footprint machine, which is benchmarked as being the second fastest IBM-compatible in the world, losing a little ground to the Compaq 386/20. However, Dell is quick to point out that their machine is actually based on a 286 CPU, running at 20MHz. Billed as the System 220, entry price is £2099 for a mono VGA system with 40MB hard drive. Add £300 for colour VGA. The small footprint means that there are only 3 free slots, and that the system uses 3.5 inch drives.

If you really want speed, though, there's the new System 310. This is a 386 system running at 20MHz and is noticeably faster than Compaq's. With 1MB of dynamic RAM, a mono system starts at £2799. The top of the range is a colour VGA system with 322 MB ESDI hard drive for £5099, which is considerably cheaper than a fully configured PS/2 model 80. Of course, you don't get Microchannel with the Dell machines, though this is promised before the end of the year. The System 400 and System 500, which were actually demonstrated by Dell in London last month (I know, 'cos I was there), are microchannel-based machines that are equivalent to the PS/2 models 50, 60 and 80. The machines use the C&T chipset and have a Phoenix BIOS. No prices are available, though Dell says that adding Microchannel will push up retail prices by around 30% because of the extra complexities involved in manufacture. No details of the levy demanded by IBM's licensing department was announced, though it's thought to be about 1% of the RRP.


Atron's latest debugger helps you fine-tune Windows applications.

Windows Debugger

Atron has upgraded its hardware assisted debugger to aid developers of Windows applications. The debugger contains 1MB of protected memory on board, so that the debugging software will not eat up RAM normally used by the application being debugged. The debugger, which runs from an external terminal to avoid the problems caused by Windows taking over screen and keyboard, costs £495. It's available in the UK from QA Training in Cirencester, who can be contacted on 0285 5888.

Motorola launches RISC CPU

Motorola has finally launched its long-awaited RISC CPU. The company also used the occasion of the launch to reveal more details about the still-unannounced 68040.

The RISC chip family is known as the 88000. This is actually composed of 2 separate 32-bit chips. The primary processor is designated the 88100, and a system would typically contain just one of these 20 MHz devices. The 88200 is the other part of the package, and acts as a 16 KB cache and memory management unit. At least 2 of these are normally used in a system, to provide separate instruction and data caches. The chips will not be in full production until February next year, at which point they will retail for around $400 each (that's $800 a pair).

Because all the 88100's instructions are implemented directly in hardware, it runs significantly faster than something like an 80386 that relies on a large amount of microcode. Two companies are known to be working on PC emulators for the 88000, and Motorola has stated that these emulators should produce a PC that runs faster than today's 386-based ones. If this turns out to be true, then Motorola may have finally found its place in the PC market.

The 88100 contains 32 registers, each 32 bits wide. The chip contains both integer and floating point units, and one of the reasons for its speed is that all units can operate concurrently. To keep the processor as busy as possible at all times, the units must always be kept supplied with instructions. Motorola has developed a software tool known as a code scheduler, which rearranges instructions in a program to do this.

A software development environment is now available, and includes an assembler, linker, code scheduler and C compiler. Fortran, Ada, Pascal, Cobol, Lisp and Prolog compilers are currently being developed by third parties. Also available will be a version of Unix V.3, and Motorola has signed an agreement with AT&T to ensure that it has early access to Unix V.4.

As for the 68040, this will be fully compatible with other members of the 68000 family. In a single chip, the 32-bit 68040 integrates an 030-compatible integer processing unit, an IEEE-compatible floating point unit, independent 8KB instruction and data caches and a paged MMU. With multiple internal buses and pipelines, a high degree of parallelism is achieved. More details are contained in document MC68040/D, available from Motorola's European Literature Centre in Milton Keynes.

Mnemonic	Description
Integer Arithmetic Instructions	
add	Add
addu	Add Unsigned
cmp	Compare
div	Divide
divu	Divide Unsigned
mul	Multiply
sub	Subtract
subu	Subtract Unsigned
Floating-Point Arithmetic Instructions	
fadd	Floating-Point Add
fcmp	Floating-Point Compare
fdiv	Floating-Point Divide
fldcr	Load from Floating-Point Control Register
flt	Convert Integer to Floating Point
fmul	Floating-Point Multiply
fstcr	Store to Floating-Point Control Register
fsub	Floating-Point Subtract
fxcr	Exchange Floating-Point Control Register
int	Round Floating Point to Integer
nint	Floating-Point Round to Nearest Integer
trnc	Truncate Floating Point to Integer
Logical Instructions	
and	AND
mask	Logical Mask Immediate
or	OR
xor	Exclusive OR

Mnemonic	Description
Bit-Field Instructions	
clr	Clear Bit Field
ext	Extract Signed Bit Field
extu	Extract Unsigned Bit Field
ff0	Find First Bit Clear
ff1	Find First Bit Set
mak	Make Bit Field
rot	Rotate Register
set	Set Bit Field
Load/Store/Exchange Instructions	
ld	Load Register from Memory
lda	Load Address
ldcr	Load from Control Register
st	Store Register to Memory
stcr	Store to Control Register
xcr	Exchange Control Register
xmem	Exchange Register with Memory
Flow Control Instructions	
bb0	Branch on Bit Clear
bb1	Branch on Bit Set
bcnd	Conditional Branch
br	Unconditional Branch
bsr	Branch to Subroutine
jmp	Unconditional Jump
jsr	Jump to Subroutine
rte	Return from Exception
tb0	Trap on Bit Clear
tb1	Trap on Bit Set
tbnd	Trap on Bounds Check
tcnd	Conditional Trap

Instruction set for the Motorola 88100 RISC CPU.

S&S Goes Half Inch

S&S Software has set up a new service for transferring 1600 bpi half-inch tapes (or rather the data from them) to standard MS-DOS disks. Formatted conversion between ASCII, EBCDIC and BCD is offered, as is pre-processing and compression if required. If you're about to throw out your mainframe and replace it with a PC running OS/2, send a large box of floppies to S&S but call 0494 728095 for a quote first.

Unix 1-2-3

Lotus has announced that it is developing a Unix version of 1-2-3. The package will be Lotus's first entry into the Unix applications market, and the company has already said that it will not be the last. The Unix version is being developed alongside Release 3.0 of the DOS product, the launch of which is still being delayed while the developers try to get the size of the code down to a more acceptable level.

The Unix version of 1-2-3 will be POSIX compliant and will adhere to the X/Open specification.

Also announced by Lotus recently is Blueprint, a powerful data access architecture to allow users to capture data from a number of sources into 1-2-3 worksheets. The Blueprint system will be incorporated into all future Lotus products and, with the aid of a special toolkit, developers will be able to develop drivers for their own software so that it can communicate bi-directionally with Lotus software. Lotus sees possible uses for the system as CD-ROM and live satellite data feeds, as well as more conventional mainframe links.

Lotus has also talked for the first time about its database system that it is developing for OS/2. The product will run under Presentation Manager (ie it's graphics based) and will support SQL. Lotus/DBMS, as it's called, will be closely integrated with 1-2-3/G, the graphics-based version of 1-2-3 being developed for Presentation Manager and Windows. More from Lotus on 0753 840281.

Bug or standard?

It's not a bug, it's a standard. That's the official word from Microsoft concerning rather strange happenings in QuickBASIC 4.0 as we reported in last month's .EXE. It all started when a programmer called the .EXE office to complain that his accounting software started getting things wrong since he converted to QB 4.0. It turned out that, when rounding decimal numbers to integers, the process was not going as it should. If a decimal number ends in .5 (say 4.5, or 7.5) and you round it to a whole number, even numbers like 4.5 get rounded down (to 4, in this case) while odd numbers (like 5.5) get rounded up (to 6, here). Microsoft checked this out for me, and I have been assured that the current IEEE standard says that numbers must be rounded to the nearest EVEN integer, and not simply the nearest integer.

Meanwhile, be careful if you're converting software to QB 4.0 from any other version of Microsoft Basic.

A Million Megabytes

Vancouver based Creo Technology has developed a recorder for use with reels of ICI's digital paper – a new form of optical data storage. The machine, shown for the first time at a recent Chicago event, can hold 5,000 mag tape cartridges on a single reel of the new tape. That's a billion sheets of typed paper, or a million megabytes of data. Storage costs work out at just over half a cent per megabyte. A reel of terabyte tape is 880 metres by 35mm and average seek time to any byte is 28 seconds. Data life is quoted at a minimum of 15 years. No price is yet available, but the company says the recorder, used in a 24-hour DP operation, will pay for itself in 8 days simply on the reduction in media costs! More from ICI on 0707 323400.

Presentation Manager Ships to developers

The latest update to the OS/2 SDK started shipping in the US early in April. The bug list stands at over 200, but all are documented and Microsoft considers the system stable enough for developers to work with. The system runs slightly slower than Windows 2.0, though the actual performance will be improved before the software is made available to end users.

Presentation Manager looks very similar to Windows 2.0 in operation. It adds windowing to OS/2 and replaces the Session Manager screen, which no longer exists. This means that you can now have multiple OS/2 applications executing on the same physical screen, instead of having to swap between them with Ctrl and Alt-Escape combinations. The biggest change to the user interface, however, is that files are now depicted by icons and not just by names.

According to reports from the US, the final release of OS/2 1.1 will fill five 1.2MB floppies and take 6MB of disk space. An installed version will require 2MB of RAM, though this will be reduced if the DOS compatibility box is disabled.

Cheaper Apollo

Apollo has chopped 35% off the price of its entry level workstations. The Series 4000, with 68020 CPU, 68881 CoPro, 19 inch mono monitor and 4MB of RAM now sells for £7495. Apollo quotes the machine's speed at 4 MIPS, compared with the 5 MIPS offered by the new (and much more expensive) Sun 386i systems.

Helios conference

A Helios developers' conference will be held in Bath on 7th and 8th of July. This is the first conference of its kind, I'm told, and speakers will include Tim King and Nick Garnett, two of the Helios developers. David May from Inmos and Ray Anderson from IXI (The X-Window people) will also be around.

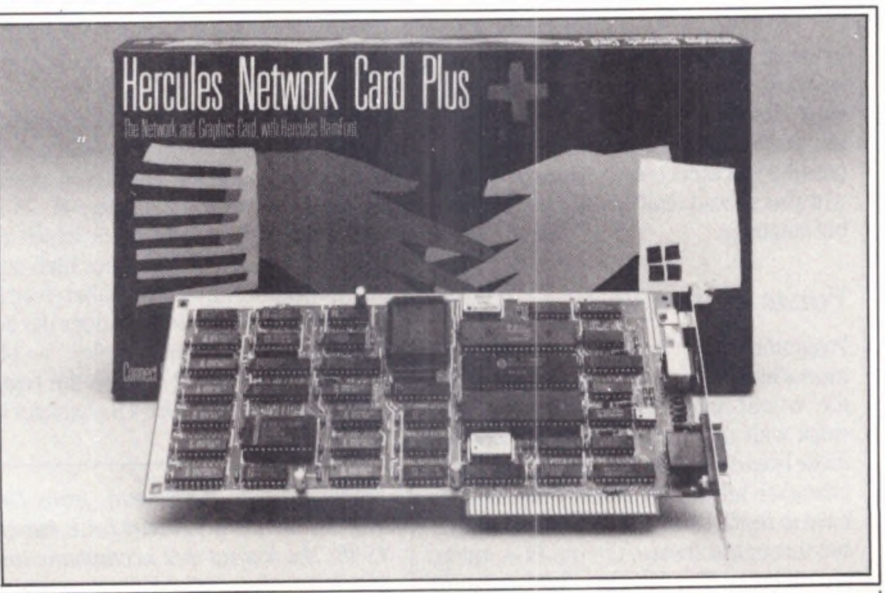
The 2-day event is aimed at programmers and system designers involved in developing Transputer-based applications and who are, or may be using the Helios operating system. As well as providing delegates with an in-depth look at the innards of Helios itself, much talk is also promised about X-Window and parallel programming in general.

Helios is an operating system for the Transputer, developed in the UK by Perihelion software. Helios is a fully distributed, multi-tasking OS with a high level of Unix compatibility (it says here), a degree of fault tolerance and includes X-Window support. It's currently being used in embedded systems as well as in add-in Transputer-based boards for the PC. Although interest in the Transputer among PC users is currently low, this may change if IBM launches its much-rumoured PS/2 Model 90 which, we're told, will include both an 80386 and a Transputer. Atari already has a Transputer-based system, namely the Abaq workstation, and this will be running Helios.

For more information about the conference, call 0749 4203.

Who Needs OS/2?

AI Architects Inc, of Cambridge Mass., has launched an MS-DOS extender that allows programs to take advantage of protected mode on an 80286 or 386 without the need for OS/2. Software tools are provided to convert C, Fortran, Pascal and assembly language code to run under the extension, and code can then use up to 16 MB of RAM on a 286, or 4GB on a 386. A sampler costs $50, and the full works retails for $500. Wait until it's cheap rate, then call 0101 617 577 8052 for more information, or fax 0101 617 577 9774.


Hercules Network Card Plus
The Network and Graphics Card, with Hercules RamFont
The latest offering from Hercules integrates networking and graphics on the same card.

Hercules Network Card

Hercules has just started shipping its Network Card Plus. The new board combines a Herc graphics facility, including RamFont capability, with a TOPS/FlashTalk port. Integrating graphics and video like this frees up a slot in a crowded PC. Price is £399, complete with TOPS software. Details from distributor First Software on 0256 463344.

Companies with products for inclusion on these pages should send relevant information to the News Editor. We would also like to receive information on any product known to work in protected mode under OS/2, for a list currently being compiled.

An Italic Font in C for the EGA and VGA

Andrew Chalk explains how to design and implement a custom font for the EGA in Turbo C. He also lists source code for a resident program that defines and installs the font.

Innovative screen displays sell software, which is one reason why so many developers provide demonstration disks, often free. One thing that makes a program distinctive is having a custom typeface for the text that it displays. Until the advent of the EGA, this was a luxury reserved for programs that operated in graphics mode.

Developing a non-graphics application to operate in graphics mode (for example, Framework in its EGA two-colour graphics mode) is more costly than using the strong text mode support in the PC, however. Furthermore, prior to the EGA, graphics (with the exception of Hercules) were too poor to be very attractive. The EGA was such a rarity a year after its introduction that Peter Norton could write in his authoritative 1985 'Programmer's Guide to the IBM PC' that "we won't be discussing the 64-colour palette of the EGA/ECD combo because it's quite rare and specialised and doesn't really fit into the mainstream PC family".

This situation changed with the advent of EGA clones from Chips and Technologies and others. At the present time, a no-frills EGA card sells for around £150 and will probably cost around £100 before the end of 1988. At least for a while, the EGA will be the de facto video standard, so it pays developers to take advantage of its special features.

In this article I show how to exploit one of the interesting features of the EGA – the ability to replace the standard character font in text mode with one of your own choosing. As an example, I use an italic font and provide source code for a TSR that loads the font into RAM so DOS and (most) application programs can use it. The TSR is necessary because the EGA reloads the ROM character set when the video mode is changed. By intercepting the BIOS for video mode changes, you can reload your custom font in its place. The TSR can be deinstalled, freeing up the 17K of memory it occupies for other programs and preventing incompatibilities.

The source code is written in C (Turbo C, to be precise), so this article also serves a second purpose – explaining some of the techniques for writing resident code in high-level languages.

In order to use the code included here, you first need to understand fonts on the EGA. Then I'll talk about the C implementation in the source code. After that I'll discuss the TSR aspects of the source code and why you should really write TSRs in assembly language.

Fonts on the EGA

Programmers writing for the monochrome display adapter (MDA) and the colour graphics adapter (CGA) are stuck with the character sets provided in those board's ROMs. If you want a different character set, say as users of APL do, you have to replace the ROM, as it's not accessible through software. On the EGA, things are different. The fonts are 'soft', meaning that although the ROM character generator is used by default, it can be replaced by a character set of your choosing. In fact, the EGA can support four character sets in what IBM calls four different blocks. Normally, you use block 0.

The EGA has BIOS support for the loading of an alternate character set through interrupt 10h, function 11h (AH = 11h), subfunction 0 (AL = 0). You make a call to this function with ES:BP pointing to a table containing your font (in a format I will explain). DX should be set to the ASCII code of the first character in your character set, CX to the number of characters in your character set (maximum 256), BH set to the number of bytes per character, and BL to the block to load (usually 0). These parameters provide the BIOS with sufficient information to load your font because of the way that fonts are stored.

Figure 1 shows a letter g as a magnified version of its screen image and as a stored character in memory. The figure assumes an enhanced colour display in 25-line mode (as opposed to 43), in which case each character is 14 scan lines high and 8 pixels wide. A 25-line screen therefore fills 14 x 25, or 350 scan lines, as does the EGA. On the monochrome display, a 14x9-character box is used and, on the regular colour display, the CGA 8 x 8-character box

Reprinted, with permission, from Dr. Dobb's Journal of Software Tools, March 1988. The listings that accompany this article are also available for downloading from Cix.

is used. The 43-line mode that is popular on the EGA driving an enhanced colour or a monochrome display is achieved by loading the 8 x 8 ROM character set (because 8 x 43 is 344, just less than 350 scan lines available). BIOS support exists for this, too, although two bugs in the original EGA BIOS make its implementation too big a subject to digress into here. It was mentioned in January's issue of .EXE, though, when Robert Schifreen discussed how to patch ANSI.SYS so that DOS uses the whole of a 43-line screen.

Let's assume that the EGA is driving an enhanced colour display. In this case, each character is 14 scan lines high and 8 pixels wide. Representing this in RAM is simplified by the fact that each pixel that forms part of a character can be considered to be either on or off when a given character is on the screen. This means that the state of each pixel can be represented in binary by a single bit. Furthermore, the designers of the EGA seem to have chosen a character width of 8 because this permits each character-scan line to be represented by exactly 1 byte.

In Figure 1, the hexadecimal values of each scan line are shown above the character. The arrows that lead in the direction of the memory show that the letter g is stored as 14 contiguous bytes of data. Because g has an ASCII value of 67h, the characters next to it are the ASCII values 66h and 68h. The latter of these is h.

When you load a custom font into the EGA you tell the BIOS, through ES:BP, the address of a buffer containing your chosen character set. This buffer is 3,584 bytes (14 bytes x 256 characters) long. The EGA lets you load fewer than 256 characters, and it lets you choose the starting position in the ASCII sequence. The sequence of characters for any one load operation must be consecutive members of the ASCII character set, however, or you will get garbage on the screen.

Although the ROM character generator is used by default, it can be replaced by a character set of your own choosing

Note two important limitations of the EGA font features. First, characters have a fixed width (but may vary from 1 to 32 scan lines high), so you do not have the same flexibility as in graphics modes. Second, although your 14 x 8 fonts work fine on a monochrome monitor in 25-line mode, a different character set is required if you support different numbers of screen lines. Suppose, for example, you were going to incorporate a feature into a database such that the user could press a hot key and immediately switch into 43-line mode and thereby see more records in 'table view'. If you had a custom typeface, you would need a 43-line-mode equivalent of it that would be loaded at the same time.

Notwithstanding these limitations, the custom font capability of the EGA is impressive. As stated earlier, a video mode reset restores the ROM character set, so your application should perform a reload operation every time it performs a video mode reset. Furthermore, if you want to load your custom fonts only in certain video modes, you should check the video mode before loading. The example program ITALIC.C in Figure 2 only loads the custom character set in modes 0-3 (text modes) and 7 (monochrome).

A Resident Italic Font

The example program consists of two parts: ITALIC.C (Figure 2) is the source code and ITALIC.ASC (Figure 3) contains the code for the font.

The program first checks the system configuration to see if custom fonts are supported. This is done by means of the function get_video_info(), which first checks for the presence of an EGA in the system through the recommended BIOS call

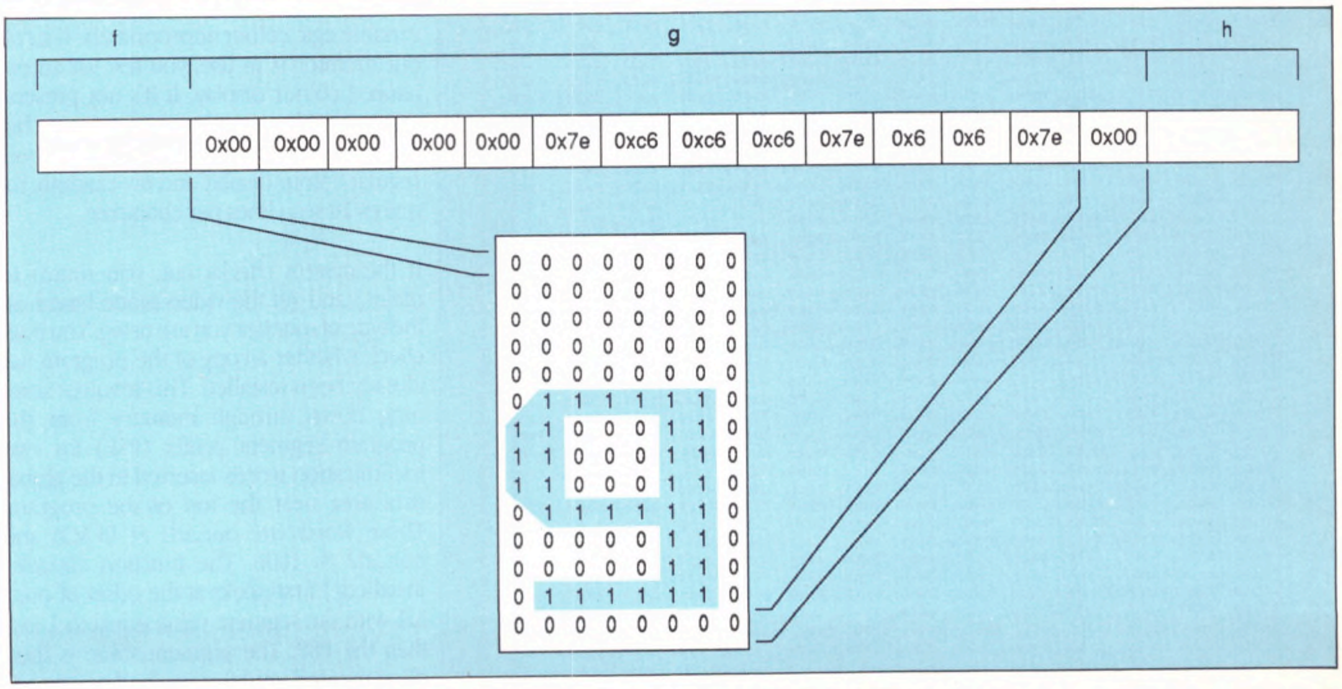


Figure 1 – EGA character representation on the screen and in memory.

```
/* ITALIC.ASC -- This is an ASCII representation of the italic font    */
/* characters used in ITALIC.C.        This file is #includeD.         */
/* In the table below, each row corresponds to a character.  The       */
/* 14 elements of each row correspond to the 14 scan lines of the      */
/* character.                                                          */
char italic_arr[128-32][14] = {
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x06, 0x0f, 0x1e, 0x1e, 0x18, 0x18, 0x00, 0x30, 0x60, 0x00, 0x00, 0x00},
{0xc0, 0x0c, 0x99, 0x19, 0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1b, 0x1b, 0x7f, 0x36, 0x6c, 0x6c, 0xfc, 0xd9, 0xb0, 0x01, 0x00, 0x00},
{0x03, 0x03, 0x9f, 0xf1, 0x61, 0xe0, 0x7c, 0x06, 0x0c, 0x8d, 0xf0, 0x61, 0xc0, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x61, 0xe3, 0x0c, 0x18, 0x60, 0xcc, 0x18, 0x03, 0x00, 0x00},
{0x00, 0x00, 0x0e, 0x1b, 0x36, 0x1c, 0x76, 0xdc, 0x98, 0x99, 0xd8, 0x01, 0x00, 0x00},
{0x00, 0x06, 0x0c, 0x0c, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x03, 0x06, 0x18, 0x18, 0x30, 0x30, 0x60, 0x30, 0x30, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0c, 0x06, 0x06, 0x06, 0x0c, 0x0c, 0x18, 0x30, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x33, 0x1e, 0xff, 0x3c, 0xcc, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x0c, 0x0c, 0x7e, 0x18, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x60, 0xc0, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x60, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x80, 0x01, 0x06, 0x0c, 0x30, 0x60, 0x80, 0x01, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x71, 0x63, 0xcf, 0xf6, 0xe6, 0x8c, 0x8c, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x06, 0x0e, 0x3c, 0x0c, 0x18, 0x18, 0x30, 0x30, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x31, 0x03, 0x06, 0x18, 0x30, 0xc0, 0x8c, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x11, 0x03, 0x03, 0x3c, 0x06, 0x0c, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x03, 0x07, 0x1e, 0x36, 0xcc, 0xfe, 0x18, 0x18, 0x78, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x30, 0x60, 0x60, 0x7c, 0x06, 0x0c, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0e, 0x18, 0x60, 0x60, 0xfc, 0xc6, 0x8c, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x71, 0x03, 0x06, 0x18, 0x30, 0x60, 0x60, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x71, 0x63, 0x63, 0x7c, 0xc6, 0x8c, 0xcc, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x71, 0x63, 0x63, 0x7e, 0x06, 0x0c, 0x18, 0xe0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x06, 0x0c, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x06, 0x0c, 0x00, 0x00, 0x00, 0x30, 0x30, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x01, 0x03, 0x0c, 0x18, 0x60, 0x30, 0x30, 0x18, 0x18, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x00, 0x00, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x18, 0x0c, 0x0c, 0x06, 0x06, 0x0c, 0x30, 0x60, 0x80, 0x01, 0x00, 0x00},
{0x00, 0x00, 0x9f, 0xf1, 0x63, 0x06, 0x18, 0x18, 0x00, 0x30, 0x60, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x9f, 0xf1, 0x63, 0xef, 0xde, 0xde, 0xb8, 0x81, 0xf0, 0x01, 0x00, 0x00},
{0x00, 0x00, 0x04, 0x0e, 0x36, 0x62, 0xc6, 0xfe, 0x8c, 0x8c, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x1b, 0x33, 0x33, 0x7c, 0x66, 0xcc, 0xcc, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0f, 0x19, 0x61, 0xe0, 0xc0, 0xc0, 0x84, 0xcc, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3e, 0x1b, 0x33, 0x33, 0x66, 0x66, 0xcc, 0xd8, 0xe0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x19, 0x31, 0x30, 0x78, 0x60, 0xc4, 0xcc, 0xfc, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x19, 0x31, 0x30, 0x7c, 0x60, 0xc0, 0xc0, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0f, 0x19, 0x61, 0xe0, 0xc0, 0xde, 0x8c, 0xcc, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x71, 0x63, 0xe3, 0xfe, 0xc6, 0xcc, 0x8c, 0x88, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0f, 0x06, 0x0c, 0x0c, 0x18, 0x18, 0x30, 0x30, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x07, 0x03, 0x06, 0x06, 0x0c, 0x0c, 0x98, 0x98, 0xe0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x39, 0x19, 0x36, 0x36, 0x78, 0x68, 0xc8, 0xc8, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3c, 0x18, 0x30, 0x30, 0x60, 0x60, 0xc4, 0xcc, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x7b, 0x7f, 0xfe, 0xd6, 0xc6, 0x8c, 0x8c, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x79, 0x7b, 0xff, 0xde, 0xce, 0x8c, 0x8c, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0e, 0x1b, 0x63, 0xe3, 0xc6, 0xc6, 0x8c, 0xd8, 0xe0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x19, 0x33, 0x33, 0x7c, 0x60, 0xc0, 0xc0, 0xc0, 0x80, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x71, 0x63, 0xe3, 0xc6, 0xd6, 0xbc, 0xf8, 0x30, 0x38, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x13, 0x33, 0x33, 0x7c, 0x6c, 0xcc, 0xcc, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1f, 0x71, 0x63, 0x30, 0x38, 0x0c, 0xcc, 0xcc, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x3f, 0x2d, 0x0c, 0x18, 0x18, 0x30, 0x60, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x71, 0x63, 0xe3, 0xc6, 0xc6, 0xcc, 0xcc, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x71, 0x63, 0xe3, 0xc6, 0xc6, 0xd8, 0xf0, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x71, 0x63, 0xc3, 0xd6, 0xd6, 0xdc, 0xf8, 0xb0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x31, 0x71, 0x36, 0x1c, 0x38, 0x78, 0xd8, 0x8c, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1b, 0x1b, 0x32, 0x36, 0x3c, 0x18, 0x30, 0x30, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x3f, 0x71, 0x46, 0x0c, 0x30, 0x60, 0x84, 0x8c, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0f, 0x0c, 0x18, 0x18, 0x30, 0x30, 0x60, 0x60, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x20, 0xf0, 0x70, 0x38, 0x38, 0x1c, 0x1c, 0x0c, 0x08, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0f, 0x03, 0x06, 0x06, 0x0c, 0x0c, 0x18, 0x18, 0xf0, 0x00, 0x00, 0x00},
{0x02, 0x07, 0x9b, 0xf1, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x00},
{0x06, 0x06, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3c, 0x0c, 0x7c, 0x98, 0x98, 0xd8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x38, 0x18, 0x30, 0x3c, 0x6c, 0x66, 0xcc, 0xcc, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0xc6, 0xc0, 0x80, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x07, 0x03, 0x06, 0x1e, 0x6c, 0xcc, 0x98, 0x98, 0xd8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0xc6, 0xfe, 0x80, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0e, 0x1b, 0x32, 0x30, 0xf8, 0x60, 0xc0, 0xc0, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3b, 0xcc, 0xcc, 0x98, 0xf8, 0x30, 0x30, 0xc0, 0x00},
{0x00, 0x00, 0x38, 0x18, 0x30, 0x36, 0x76, 0x66, 0xcc, 0xcc, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x06, 0x06, 0x00, 0x1c, 0x18, 0x18, 0x30, 0x30, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x01, 0x01, 0x00, 0x07, 0x06, 0x06, 0x0c, 0x0c, 0x98, 0x98, 0xe0, 0x00},
{0x00, 0x00, 0x38, 0x18, 0x30, 0x33, 0x6c, 0x78, 0xd8, 0xcc, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x0e, 0x06, 0x0c, 0x0c, 0x18, 0x18, 0x30, 0x30, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0xf6, 0xfe, 0xd6, 0xac, 0xac, 0x88, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0xee, 0x66, 0x66, 0xcc, 0xcc, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0xc6, 0xc6, 0x8c, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x6e, 0x66, 0x66, 0xcc, 0xf8, 0x80, 0x80, 0x80, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3b, 0xcc, 0xcc, 0x98, 0xf8, 0x30, 0x30, 0xf0, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x6e, 0x76, 0x66, 0xc0, 0xc0, 0xc0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x46, 0x70, 0x38, 0x8c, 0xf0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x04, 0x0c, 0x18, 0x7e, 0x30, 0x30, 0x60, 0x6c, 0x70, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0xe6, 0xcc, 0xcc, 0x98, 0x98, 0xd8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x33, 0x66, 0x66, 0xcc, 0xf8, 0x60, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x63, 0x46, 0xd6, 0xac, 0xfc, 0xb0, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x63, 0x6c, 0x38, 0x70, 0xd8, 0x98, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x63, 0x66, 0xc6, 0x8c, 0xfc, 0x18, 0x30, 0xe0, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0x4c, 0x18, 0x60, 0xcc, 0xf8, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x03, 0x06, 0x0c, 0x0c, 0x70, 0x18, 0x30, 0x30, 0x38, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x06, 0x06, 0x0c, 0x0c, 0x00, 0x18, 0x30, 0x30, 0x60, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x1c, 0x06, 0x0c, 0x0c, 0x0e, 0x18, 0x30, 0x30, 0xc0, 0x01, 0x00, 0x00},
{0x00, 0x00, 0x1d, 0xf7, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
{0x00, 0x00, 0x00, 0x00, 0x10, 0x38, 0x6c, 0xc6, 0xc6, 0xfe, 0x00, 0x00, 0x00, 0x00}
```

Figure 2 – An ASCII representation of the italic font. Each row corresponds to a character.

(function 12h). If an EGA is not present, the value of BL is returned unchanged, in which case you exit with a message explaining to the user that an EGA is required. Some people check for an EGA card be searching for the IBM signature in the EGA's BIOS. Not only is this kludgy, but it is also specifically disapproved of in the EGA BIOS listing. If you have to resort to such means, it's probably easier to simply ask the user whether he has an active EGA, and to ask him to press Y or N.

It is not enough to know that an EGA is present – it must also be active. The user may, for example, have an EGA driving a colour monitor and an MDA driving a monochrome monitor, and be currently be using the MDA. To see if the EGA is active, you check that bit 4 of the EGA information byte in the ROM data area at 0040:0087h is 0. If not, you exit with a message to the user explaining that the EGA must be active.

If space had permitted, I would have included code to save the video configuration and switch adapters. The saved information could then be used to restore the state of the machine on exit. For the same reason, I have also omitted code to detect a switch of adapters while ITALIC is resident. Be warned: ITALIC illustrates a technique and is not a full-blown professional, user-proof program.

Having determined that an EGA is active, you then determine whether it drives a monochrome or a colour monitor. The value of BH is 1 if monochrome and 0 if colour. The result is used to set the global variable ega_colour appropriately. If a colour monitor is in use, you test for an enhanced colour display. If it's not present, you exit with an explanatory message because the EGA will use the 8 x 8 font on a regular colour display and my example requires 14 scan lines per character.

If the system checks out, you return to main() and set the video mode based on the type of monitor you are using. You then check whether a copy of the program has already been installed. This involves scanning down through memory from the program segment prefix (PSP) for two identification words inserted in the global data area near the top of the program. These words are our_id1 = 0FACh and our_id2 = 100h. The function already_installed() first peeks at the offset of our_id1 with the segment value equal to 1 less than the PSP. The segment value is then decremented until 0 is reached or a match is found. If a match is found, the next word is peeked at and compared with our_id2. If

Quick, fix the bug...

```
stripChar(s,c)
char *s;
char c;
{
char *r;

while(*s)
    {
    if(*s==c)
        {
        r=s;
        while(r)
            *r++=*(r+1);
        continue;
        }
    s++;
    }
}
```

This function looks OK. It compiles OK. And it runs OK.

Or so it seems.

Unfortunately, it contains an insidious bug that can bring the whole program (and indeed the computer) to a spectacular halt.

As a C programmer, this is a nightmare you'll have to face sooner or later.

And with most C compilers, you're on your own.

The only way you can find out what's going on is to add temporary debugging instructions to the code. Running the program back through compiler and linker at each step.

No C environment can turn code around fast enough to make this bearable on a real project.

But Microsoft QuickC will get you out of trouble as quickly as you can get into it.

Because only QuickC has an integrated debugger based on Microsoft CodeView, the revolutionary debugger relied on by the world's top software developers.

So you can look inside your program and see what's really happening as it happens.

The integrated debugger is just one of the features that makes QuickC the most friendly C development system you can buy.

QuickC is more than a lightning-fast compiler and linker. It's the complete C solution including editor and MAKE, with all the help you'll need, both on-screen and off.

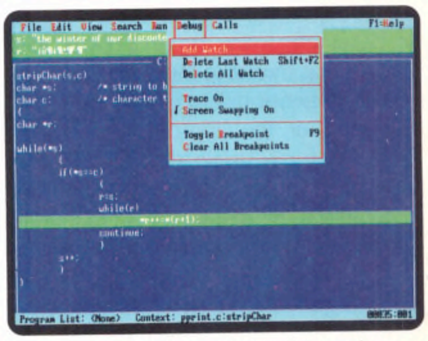
Microsoft QuickC won't do your job for you, but it'll help you do it better and faster. And it eliminates a lot of the guesswork.

Speaking of which, have you fixed that bug yet?

C power.

- Integrated menu-driven environment with editor, compiler, linker, debugger and automatic MAKE facility.
- Also includes command line compiler, LINK, MAKE and LIB.
- Full K&R C with ANSI extensions.
- Generates fast, compact, efficient code.
- In-memory compilation at 10,000 lines per minute.*
- Supplied with over 350 library routines, including full graphics library, floating-point maths, DOS and BIOS routines.
- Supports small, compact, medium and large memory models.†
- Fully source and object code compatible with the Microsoft C Optimizing Compiler, version 5.1.
- Supportive editor with parenthesis, brace and bracket matching; Cut and Paste editing; Search, Replace and Undo commands; support for 43-line EGA text mode; on-screen Help with QuickC, QuickC library functions and the C language.
- Source-level animating debugger with watchpoints, dynamic breakpoints and CodeView compatibility.

*IBM PS/2 Model 60.
†Medium model only within integrated environment.



```
s: "the winter of our disconte
r: "[illegible]"

stripChar(s,c)
char *s;          /* string to b
char c;           /* character t
{
char *r;

while(*s)
        {
        if(*s==c)
                {
                r=s;
                while(r)
                        *r++=*(r+1);
                continue;
                }
        s++;
        }
}

Program List: (None)   Context: pprint.c:stripChar                00035:001
```

Nothing gets you running in C faster than Microsoft QuickC, with integrated debugging.

YES! Please send me more information about Microsoft QuickC, together with a corrected source listing.

I'm also interested in:
☐ Microsoft C Optimizing Compiler version 5.1, with CodeView.
☐ Other Microsoft languages.

Name	....
Company	....
Address	....
	....
Postcode	....
Telephone	....

PLEASE USE BLOCK CAPITALS

Complete and return coupon to Microsoft QuickC, FREEPOST, London NW9 5AL, or call 01-200 0200. (No stamp necessary for coupon if posted in the UK.)

Microsoft QuickC.

EXE1

FOR MORE INFORMATION ABOUT MICROSOFT QUICKC, CALL 01-200 0200.

CIRCLE NO 4

you assume that all characters are equally likely, the chance of erroneously concluding that ITALIC is resident when it is not if you load halfway up memory on a 640K machine is 1 in 13,158. If you are uncomfortable with this, you can lengthen the odds by searching for more than two words. (There are other, better ways of checking for the presence of a TSR prior to installation, and we'll be looking at these in a future .EXE).

If you find a copy of ITALIC, you deinstall it and print a message telling the user. The mechanics of deinstalling a resident program are explained in the next section. If ITALIC is not present in memory, you can proceed with installation. There are three distinct phases of this process.

First, you only want to replace the alphanumeric characters in the ASCII set. Bow-drawing characters simply don't draw boxes if they are italicised. My strategy is to load a copy of the whole 14 x 8 ROM character set into the buffer called fontarray and then load only the alphanumeric characters from the data file ITALIC.ASC. The advantage of this is that your .EXE file need not be swollen with unnecessary data for characters that do not differ from the ROM 14 x 18-character set. Retrieving the ROM character set is easy thanks to BIOS support and is accomplished in get_egafont(). Next, a for() loop overlays the ASCII characters 32 through 127 with italic characters.

If you refer to the listing of ITALIC.ASC, you find that the italic font data is set up as a two-dimensional array with each row consisting of the appropriate hexadecimal values for a single character. The listing shown is actually the output of FONTEDIT, a full-screen, real-time EGA font editor included in the C Windows Toolkit. Before you take one look at the listing and give up in disgust, note that it will be available electronically. Details at the end.

The second stage is the loading of your font (telling the EGA to use it) using interrupt 10h, function 11h, as described earlier. This is accomplished by the function load_user_egafont(). At this point the screen display immediately changes to reflect the new font.

The third stage is the resident installation of a replacement interrupt handler for interrupt 10h so that you can detect video mode changes and reload your font if necessary. First, you save the PSP and environment pointer in the PSP at offset 2Ch to use for later deinstallation. Next, you save the old interrupt 10h address using getvect() and install your own handler with setvect(). Finally, you terminate and stay resident (more on this in the next section).

The new font affects every character on the screen. The italic font presented here is probably not distinct enough for serious text work, but it could be edited to be so. The italic font in Microsoft Word is created through exactly the same kind of techniques. Other fonts along the lines of the large selection supplied with the Hercules Graphics Card Plus are also possible.

Having determined that the EGA card is active, we must then determine whether it is driving a mono or colour monitor.

Resident Programs in High-Level Languages

As I stated in the introduction, the experience of writing this (simple) TSR in C has lead me to conclude that such programs are best written in assembly language. There are several reasons for this. Perhaps the most important one is the sheer size. ITALIC occupies 13K RAM, of which 3,585 bytes are the font buffer, 1,344 bytes are the replacement font data, less than 100 bytes are for other global data, and 512 bytes are for the .EXE header. The remaining 11,500 odd bytes are 'code'. It is this portion that assembly language could shrink down to perhaps 600 bytes, (I have not done it for this program). A second reason in favour of assembly language is the irrelevance of the portability issue.

An argument often legitimately raised in favour of high-level languages is their advantage in terms of development time. In fact, for resident code, even given that I was working without the benefit of large literature on the ins and outs of programming TSRs in high-level languages such as exists for assembly language, so many problems resulted from having a compiler between me and the machine that I spent a lot of time inside the debugger. Assembly-language TSRs are easier to debug.

It can hardly be argued that the problems arose from the choice of language. Wasn't it the low-level links that made C so popular as an application language? It is also difficult to argue that the choice of compiler was the problem. Turbo C has library support for all the function calls associated with resident code. Although it lacks a built-in debugger (at the moment) Periscope does an admirable job. In-line assembly language is also available, but the objective here was to not use a single line of in-line code (that isn't really considered as writing a TSR in a high-level language).

It might be argued that programming TSRs obviates the need to learn assembly language. The last person I would recommend to write a TSR in a high-level language is someone who did not know assembly language. Ironically, the alleged portability of C code to different compilers (because it is a high-level language) becomes the opposite with TSRs. Because the generated code differs across compilers, code that runs flawlessly compiled with one compiler can produce subtle and hard-to-track bugs on another. The truly 'safe' code becomes the one that is 'immune' from portability virtues – assembly language.

These objections aside, here is how you implement a simple resident program in Turbo C. Bear in mind that this program does not access the disk or the keyboard, so many TSR techniques are not discussed. For a fuller treatment I recommend 'Turbo C: The Art of Advanced Program Design, Optimisation and Debugging' by Stephen Randy Davis. I have no doubt that ITALIC could be implemented more efficiently in C than I have done here, but the relevant question is how these improvements compare with the real alternative – assembly language.

First, consider the way that you would implement this TSR in assembly language. Near the top of the source code would be the resident section, preceded by a jump to the installation section, which you would jettison when the resident part was installed. In C you perform the same installation steps of saving the old interrupt 10h vector and installing your own.

A problem arises when you wish to terminate and stay resident. Turbo C contains the keep() function, which implements interrupt 21h, function 31h, and requires the

For Sale:– Bug free real-time code

Unlikely? Not necessarily.

We at Ready Systems have spent literally thousands of man years in design, development, and QA of a real-time, deterministic multi-tasking operating system, available on a wide variety of processors.

Millions of copies are in use, and thousands of customers have used our kernel VRTX in a wide variety of applications from submarines to satellites.

We offer an integrated solution for real-time design with CARDtools, the worlds first real-time design tool along with real-time C, Ada and of course VRTX and associated tools.

To find out more about how we can help you develop your applications to deliver the right answer on time, visit our U.K. distributor IPL on Stand 26 at Software Tools '88 or return the coupon below.

Please send me further information on the subject matter ticked below.

- ☐ **CARDtools**
- ☐ **VRTX 32**
- ☐ **RTAda**
- ☐ **Real-time training courses**
- ☐ **My need is urgent, please contact me**

Name ______________________

Position ______________________

Company ______________________

Address ______________________

____

Post Code ____________ Tel: ____________

READY SYSTEMS

Ready Systems UK Ltd
33 Queen Street
Maidenhead
Berks, SL6 1NB
Tel: 0628 773100
Fax: 0628 773044

CIRCLE NO 5

```c
/* Copyright (C) Magna Carta Software, 1987.  All Rights Reserved.      */
/* MAKEFONT -- Makes an italic font for the EGA.                        */
/* Note: Do not run this program from within the IDE.                   */
/* First version 10/29/87.  Last update: 10/31/87.                      */

#ifndef __SMALL__
     #error Should use SMALL compilation model
#endif

#include <stdio.h>
#include <dos.h>
#include <process.h>
#include <mem.h>
#include <stdlib.h>

#define TRUE 1
#define FALSE 0


/* Functions related to video operations */
void load_user_egaxfont(char *fptr,int block,int bpc,int char_count,int spos);
void get_egafont(char *fptr, int font);
int get_video_info(void);

/* Global variables and #DEFINES related to video operations */
#include "mk_ital.asc"          /* ITALIC.ASC contains our italic font */
#define VIDEO 0x10                              /* BIOS video interrupt */
char fontarray[3585];                           /* buffer for font storage */
int our_id1 = 0xfac, our_id2 = 0x1000;          /* our ID words */
char ega_color;

/* Functions related to TSR operations */
int already_installed(void);
int deinstall(void);
void interrupt (*old_int10h)(void);
void interrupt int10h(unsigned bp, unsigned di, unsigned si,
                      unsigned ds, unsigned es, unsigned dx,
                      unsigned cx, unsigned bx, unsigned ax);

/* Global variables related to TSR operations */
unsigned save_bp1, save_bp2, old_ds, old_psp;
unsigned old_env;


/* Turbo C system variables (see text for explanation ) */
extern unsigned __brklvl;
extern unsigned _psp;


/* Other functions */
void error(int errnum);


main()
{
        int i, j, k;
        union REGS regs;

        get_video_info();
        if (!ega_color) regs.x.ax = 0x7;        /* set the video mode */
        else regs.x.ax = 0x3;
        int86(VIDEO,&regs,&regs);
        if (already_installed()) {
                deinstall();
                printf("\The Italic font is now no longer installed");
                exit(0);
        }

        /* system checks out -- go ahead and put italic chars. in font */
        get_egafont(fontarray,14);      /* store the ROM font in fontarray */
        for(i=14*32,j=0; i< 14*128;j++) {
                for(k=0;k<14;k++) fontarray[i++] = italic_arr[j][k];
        }
        load_user_egaxfont(fontarray,0,14,256,0);       /* load our font */

        old_psp = _psp;                 /* save the resident program's PSP */
        old_env = peek(_psp,0x2c);      /* save the resident program's ENV */
        old_int10h = getvect(VIDEO);    /* save the old vector */
        setvect(VIDEO,int10h);          /* install our INT10h handler */

        /* terminate and stay resident. Program length is determined by */
        /* subtracting the psp address (_psp) from __brkval which is    */
        /* dynamically set to the address of the end of DS.  This       */
        /* appears to be reliable in the TINY and SMALL models, but     */
        /* results are unknown for other models.                        */
        keep(FALSE, _DS + (__brklvl + 15)/16 - _psp);
}

/* ALREADY_INSTALLED: This routine scans through memory for our ID byte.*/
/* Returns: 0 if not found, 1 if found.                                 */
int already_installed()
{
        unsigned int next_seg;

        for(next_seg = _psp-1; next_seg > 0; next_seg--) {
                if (peek(next_seg,(unsigned) &our_id1) == our_id1) {
                        if (peek(next_seg,(unsigned) &our_id2) == our_id2) {
                                old_ds = next_seg;
                                return (1);
                        }
                }
        }
        return (0);
}


/* DEINSTALL: Remove our TSR by reseting interrupt 0x10 and video mode. */
int deinstall()
{
        union REGS regs;
        struct SREGS sregs;

        /* initialize old interrupt vector */
        old_int10h = MK_FP(peek(old_ds,(unsigned) &old_int10h+2),peek(old_ds,(unsigned)
                                                                  &old_int10h));
        setvect(VIDEO,old_int10h);      /* reset the interrupt vector */
        if (!ega_color) regs.x.ax = 0x7;        /* reset the video mode */
        else regs.x.ax = 0x3;
        int86(VIDEO,&regs,&regs);

        /* Deallocate the memory used by the resident program */
        old_psp = peek(old_ds, (unsigned) &old_psp);
        old_env = peek(old_ds, (unsigned) &old_env);

        regs.x.ax = 0x4900;     /* DOS function to free allocated memory */
        sregs.es = old_psp;
        intdosx(&regs,&regs,&sregs);
        if (regs.x.cflag) error(3);

        regs.x.ax = 0x4900;     /* DOS function to free allocated memory */
        sregs.es = old_env;
        intdosx(&regs,&regs,&sregs);
        if (regs.x.cflag) error(4);
        return (0);
}


/* INT10: This function is the BIOS video interrupt handler.            */
void interrupt int10h(unsigned bp, unsigned di, unsigned si,
                      unsigned ds, unsigned es, unsigned dx,
                      unsigned cx, unsigned bx, unsigned ax)
{
        /* execute the old video interrupt */
        if (ax >> 8) {                  /* it is not a video mode reset */
                _AX = ax;
                _BX = bx;
                _CX = cx;
                _DX = dx;
                save_bp1 = _BP;
                _BP = bp;
                (*old_int10h)();
                _BP = save_bp1;
                ax = _AX;
                bx = _BX;
                cx = _CX;
                dx = _DX;
        }
        else {                          /* AH == 0 for a video mode reset */
                _AX = ax;
                _BX = bx;
                _CX = cx;
                _DX = dx;
                (*old_int10h)();
                ax = _AX;
                bx = _BX;
                cx = _CX;
                dx = _DX;

                /* reload our font destroyed by the mode reset */
                load_user_egaxfont(fontarray,0,14,256,0);
        }
}


/* LOAD_USER_EGAXFONT -- Load a user-defined font and reset page length.*/
/* Parms: ptr. to user table, block to load, bytes-per-char,            */
/* number of chars to store, starting position in font table.           */
/* First version 7/13/87.  Last update 10/31/87.                        */
void load_user_egaxfont(char *fptr,int block,int bpc,int char_count,int spos)
{
        unsigned byte block;
        byte_block = (bpc << 8) | block;

        /* Can't use intr() due to Turbo C v1.0 compiler bug.           */
        /* Note: we must do any assignments to segments prior to doing  */
        /* assignments to AX since AX is destroyed.                     */
        _ES = _DS;
        _AX = 0x1100;                   /* call function 0x11 */
        _BX = byte_block;               /* block to load */
        _CX = char_count;               /* number of characters to load */
        _DX = spos;                     /* character offset into table */
        save_bp2  = _BP;                /* save BP for stack addressing */
        _BP = FP_OFF(fptr);             /* load address of user font */
        geninterrupt(VIDEO);
        _BP = save_bp2;                 /* restore BP -- or die... */
}


/* GET_EGAFONT: This routine grabs an EGA font from ROM and stores it   */
/* in the global variable fontarray */
void get_egafont(char *fptr, int font)
{
        struct REGPACK regs;

        regs.r_ax = 0x1130;             /* EGA BIOS call to return font */
        if (font == 8) regs.r_bx = 0x0300;
        else if (font == 14) regs.r_bx = 0x0200;
        intr(VIDEO,&regs);
        movedata(regs.r_es,regs.r_bp,_DS, (unsigned) fptr,14*256);
}

/* GET__VIDEO_INFO: A VGA or an EGA must be installed for this program  */
/* to work. The monitor must be an Enahanced Color or Monochrome        */
/* display and the correct adaptor must be active.                      */
int get_video_info()
{
        union REGS regs;
        unsigned char e_byte;

        /* First check for the presence of an EGA */
        regs.h.ah = 0x12;       /* EGA BIOS alternate select            */
        regs.h.bl = 0x10;       /* return EGA information.              */
        int86(VIDEO, &regs, &regs);
        if (regs.h.bl == 0x10) error(1);        /* EGA not found */

        /* EGA is present -- is it active? */
        e_byte = peekb(0,0x487);        /* EGA info. byte */
        if (e_byte & 8) error(2);       /* EGA not active */

        /* Does the present, active EGA drive a color or mono monitor? */
        if (regs.h.bh) ega_color = FALSE;  /* EGA drives a mono monitor */
        else ega_color = TRUE;          /* EGA drives a color monitor */

        /* See if EGA drives an Enhanced Color Display */
        if (ega_color) if (!(regs.h.cl == 3 || regs.h.cl == 9)) error(1);
        return (1);
}


/* ERROR: A simple error handler.                                       */
void error(int errnum)
{
        switch (errnum) {
                case 1: printf("\An EGA and Enhanced Color or Monochrome Display");
                        printf("\nmust be present to use this program.");
                        break;

                case 2: printf("\Please make the EGA the active adapter");
                        printf("in order to run this program.");
                        break;

                case 3: printf("\nError deallocating program memory.");
                        break;

                case 4: printf("\nError deallocating program PSP.");
                        break;

                default:break;
        }
        printf("\nProgram exiting.\n");
        exit(0xf);      /* Return code for DOS errorlevel */
}
```

Figure 3 – MAKEFONT makes an italic font for the EGA.

number of paragraphs of memory to reserve as one of its parameters. Whereas this is simple to compute in assembly language, it is not so in high-level languages generally or in Turbo C in particular (the Turbo C manual does not even mention this problem).

The method I used in ITALIC was discovered by Dean McCrory. Turbo C uses two internal variables: _psp (to contain the PSP address) and __brklvl (to contain the address of the end of the initialised and uninitialised data). As memory is dynamically obtained and released, _brklvl is adjusted accordingly. If you visualise Turbo C memory allocation as in the 'User's Guide' (that is, the data segment follows the code), then adding _brklvl to DS and subtracting the PSP address gives the size of the code and data. That is what I do in the keep() statement at the end of main.

Although this is ingenious, you should be aware of some potential problems and limitations. First, this will not work in the large data models (compact, large, and huge). Second, I do not know what happens, but you must presumably not farmalloc() any memory you wish the resident program to use. Third, this is undocumented and should be considered not fully tested.

The other part of the TSR code is the deinstallation routine, including the deallocation of the program's memory. This practice appears to be little known in both C and assembly language (it is certainly not officially documented). It probably enhances the value of a TSR to the user if it is removable, however.

The first thing you do is restore interrupt 10h to its original value. Next, you must deallocate memory. Deallocating the space occupied by a resident program is as easy in C as in assembly language, and the technique is the same. You must remove the program itself and its copy of the environment. In order to do so, when you first load the TSR, you must store the PSP and the contents of the environment pointer, which is located at offset 2Ch in the program's PSP.

When you try to install ITALIC, already_installed() finds a match and stores the data segment address of the installed copy in the global variable old_ds. When you deinstall the program, you peek at old_ds, offset by the address of old_psp to get the PSP, and then you repeat this using the offset of old_env to get the environment address. Next, you deallocate the program memory using interrupt 21h, function 49h. ES must hold the address of the installed program's PSP. Finally, you repeat this function call with ES pointing to the installed program's copy of the environment.

It is instructive to run CHKDSK before and after installation to confirm that this procedure works. I have found it to be reliable, but TSRs are a world without rules.

Summary

The techniques for loading custom fonts described here give a program an edge of distinctiveness in an ever more crowded software marketplace. Not only is this now worthwhile for developers because of the greater abundance of EGAs, but the code also works on the VGA. This means a fairly long time span for the investment in program development to pay off. Graphics environments offer users custom fonts, but program development is longer. The custom font facilities of the EGA offer a faster development path that does not require the wholesale recoding of applications.

.EXE

CROSS SUPPORT SOFTWARE

for EMBEDDED MICROPROCESSORS

Now wih X-Ray high-level debugger 68000/020

Features

- **SUPPORT:** we have been writing microprocessor software development tools since 1974. We offer the knowhow, training, product quality, and in-depth documentation needed for demanding development environments
- Advanced C and Pascal compiler design produces **ROM**mable, highly **OPTIMISED CODE**
- Hosts include **VAX, IBM PC, APOLLO, SUN, HP 9000**
- Compiler targets include **68000/08/10/20, 8086/186/286, 8080/85, Z80, 64180**
- Maths **COPROCESSOR** support
- Assemblers and simulation/debug tools for most **8, 16** and **32** bit microprocessors
- **BIT SLICE** meta assemblers
- Suitable for download to all major **ICE** hardware.

MICROTEC RESEARCH

Ringway House, Bell Road
Daneshill, Basingstoke
RG24 OFB England
Telephone: (0256) 57551
Telex: 858893
Fax: (0256) 57553

CIRCLE NO 6

A Toolkit for Turbo Pascal 4.0

Alan Solomon tries out a toolkit of routines for Borland's new Turbo Pascal version 4.

Kim Kokkonen is considered to know more about Turbo Pascal's insides that anyone. Except possibly the Dane that wrote the compiler. His company, TurboPower Software, was the first to bring out a decent debugger for Turbo 3, and also brought out many other tools for Turbo programmers. They also sold Turbo Extender, which let you use big arrays with TP3, getting around the 64K limit. Sunny Hill Software is another small US software house; they offered a set of tools for Turbo 3 that gave more advanced features (such as TSR support) than anyone else.

Now TurboPower has joined forces with Sunny Hill, and they have brought out Turbo Professional 4 (T_Prof_4), one of the first toolboxes available for Turbo Pascal 4. At £60, it offers excellent value for money, and should be purchased (along with many of the Borland Toolboxes) by any professional Turbo programmer. The time saved will recover the money spent within a few weeks.

Disk One

T_Prof_4 comes on three diskettes, although most people will only use one of them. The first disk is called the Demo disk, and for this reason I nearly ignored it. Don't – it has some excellent software on it, which stands up in its own right as programmer productivity software.

The first group of programs are pure demos, worth running to see the spectacular display of T_Prof_-4 windows. The second group of programs are for converting code written using the previous version of TurboPower's tools. It's good to see such attention being paid to the needs of existing customers, and augurs well for the future. The third group of programs are programmer's tools. These include:

SMACS, a keyboard macro processor. It doesn't have all the features of Superkey, but it has all the ones that most people use, and it's included in the price. I used SMACS while I was writing this review, to write the T_Prof_4 name with a single keystroke, and I can see how it could be very useful while programming. The source code is also a useful demonstration of how to use the TPMACRO and TPMACED units.

PCALC is a programmer's calculator; this is slightly better than the Sidekick calculator, and better behaved than Sidekick (but then what isn't?). It does binary, decimal and hex, and has And, Or, Xor, Mod, Shl and Shr operations.

PTIME is a time manager, for people charging an hourly rate. It also has an alarm, and lets you run an SMACS macro at a preset time. There is also a utility for generating reports from the file that PTIME creates.

PREF is a programmer's quick reference chart. For each integer 0 to 255, it shows you the screen character, binary, octal and hex, the ASCII code, the key that generates the character, the auxiliary key, and the video attribute associated with it. I always feel embarrassed when I have to search the BASICA manual for this kind of information. All the resident programs above were written with the T_Prof_4 modules, and I used then while writing this review. They didn't crash once.

MAKEMENU lets you design menus interactively, for later incorporation in your code via the TPMENU module. With this, you can make Lotus-style or pull-down menus that look very professional and slick. Anyone wanting to put together a menu-driven system should look closely at MAKEMENU in conjunction with TPMENU.

BSORT is a general purpose sorter – it is faster than the SORT utility that accompanies DOS and can handle bigger files. It also acts as a demonstration program showing how to use the TPSORT unit.

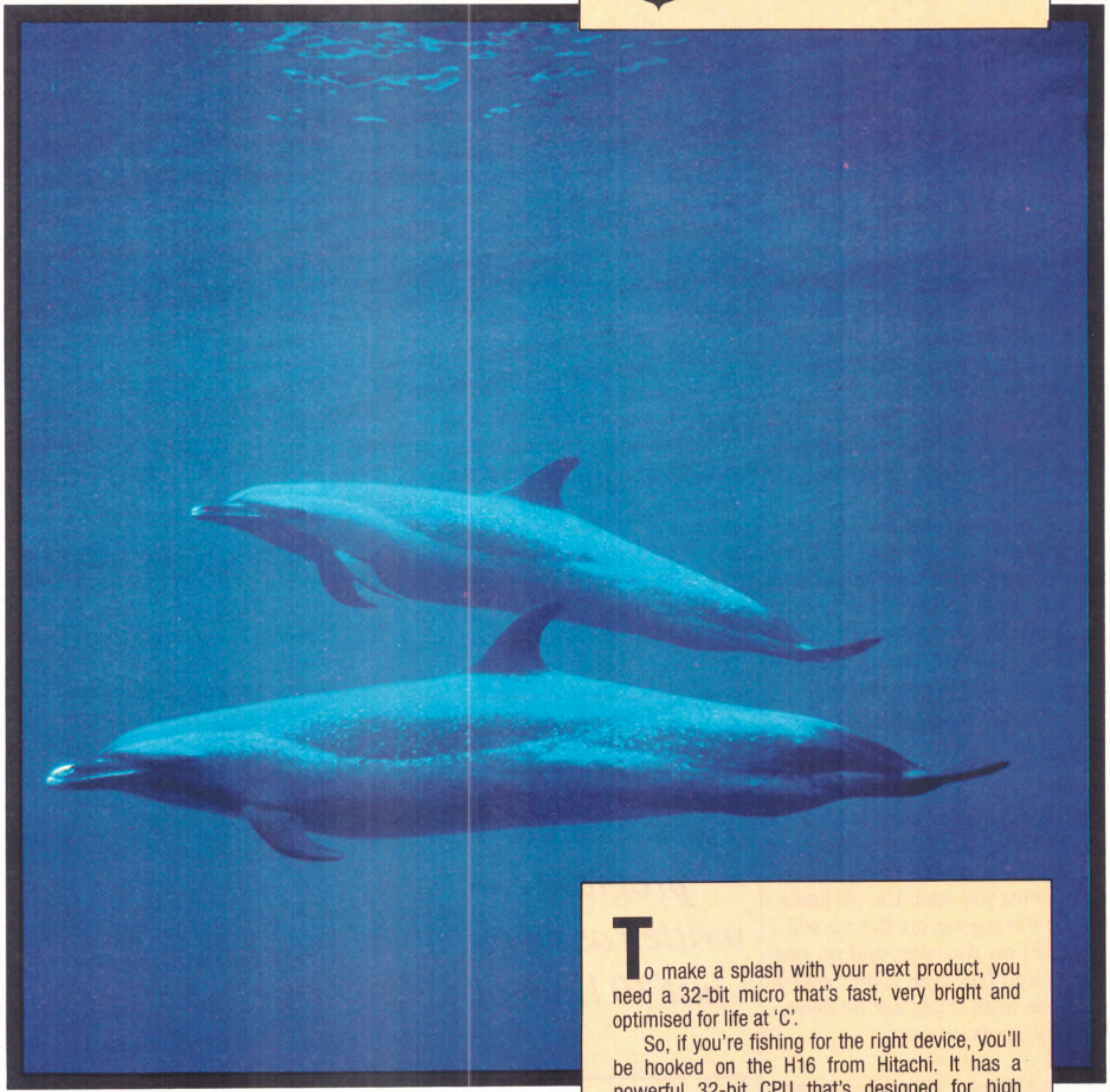
TFIND is a simple program for searching files for a string of text, while DIFF compares two files for differences – it is so simple as to be useless, but does demonstrate some advanced memory management techniques.

REP is a command repeater – it will do a command a number of times, and can pass information to the command at each invocation. This could take the form of fake

Alan Solomon is a Turbo Pascal addict, though he also confesses to using Fortran at times. He also runs S&S Enterprises, which sells other people's software and claims to recover data from mangled hard disks. Turbo Professional 4 is available from Grey Matter (amongst others) and retails for £60 + VAT. TurboPower Software can be contacted on 0101 408 438 8608.

The T-Debug Plus debugger is available from In Touch, who are on 0222882334.

HITACHI



H16 The intelligent solution for those at 'C'.

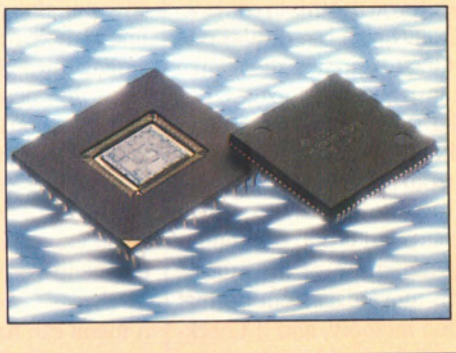
To make a splash with your next product, you need a 32-bit micro that's fast, very bright and optimised for life at 'C'.

So, if you're fishing for the right device, you'll be hooked on the H16 from Hitachi. It has a powerful 32-bit CPU that's designed for high speed bit handling. The unique 1Kbyte ring bank register structure contributes to its advanced language skills by minimising the drawbacks of high level languages.

H16 shows all-round performance with a high level of peripheral intelligence. A 4-channel DMA controller, serial communications controller and glue logic all reduce the CPU overhead and guarantee universal appeal.

We'll make sure your development goes swimmingly with our range of hardware and software support options.

At last, it really is safe to go back in the water, with H16 from Hitachi.



Hitachi Europe Ltd., Electronic Components Division (Northern Europe),
21 Upton Road, Watford, Herts WD1 7TB
Tel: (0923) 246488. Telex: 936293. Fax: (0923) 224422.

Anzac Components Limited. Telephone: (06286) 4411. Telex: 847949

Dialogue Distribution Limited. Telephone: (0276) 682001. Telex: 858944

Farnell Electronic Components. Telephone: (0532) 636311. Telex: 55147

Impulse Electronics. Telephone: (0883) 46433. Telex: 291496

Micro Call. Telephone: (0925) 825065. Telex: 628783

MMD Limited. Telephone: (0734) 481666. Telex: 846669

STC MULTIcomponent. Telephone: (0279) 442971. Telex: 818763

CIRCLE NO 7

keystrokes, or something to customise each repetition of the command.

Disks Two and Three

All of the above were simply demo programs supplied on disk one. See, I told you it was worth looking at!

The second disk contains the TPU files which are all that most people will use, and the source code for the demo programs. This source code is well worth printing out. If you examine some of the techniques used by TurboPower, your own code will be improved, and you will undoubtedly learn from their treatment of expanded and extended memory. Disk three contains the source code for these TPU files; the Pascal source of the interface modules and the assembler source of the external modules. Having full source code means that if you don't want to do things exactly in TurboPower's way, you can change the source and recompile or reassemble it.

The Units

There are about 400 routines here altogether, although a mere count of routines doesn't tell you if there is anything worth having. Turbo Pascal 4 already does almost everything you want. The old limitation on code size is gone, but there is still a 64K restriction on data size, and no one array can exceed 64K even if the heap is used. Also, the heap might not be convenient for some applications.

The main limitations of TP4 are in extreme cases. So, for example, a string is limited by the way it is stored to 255 characters, and there is no support for expanded or extended memory. It is now possible to write ISRs and TSRs using the INTERRUPT directive, but it isn't totally easy.

Screen Handling

The first group of units in T_Prof_4 are concerned with screen handling. TPCRT is a greatly improved version of Borland's CRT unit. It allows you to switch the different video pages; if you use this in conjunction with TPSCREEN, you can display one page while writing to another, then when you are ready, instantly switch pages. It is also written with TSRs in mind, as the normal CRT's keyboard routines can prevent some pop-ups from popping. Fastwrite is a very fast way to write directly to screen memory, and there is a group of routines for controlling the shape of the cursor. Movescreen will move data from normal memory to video memory, and if CheckSnow is true, it will do so without any snow or flicker on the screen (useful for those with standard CGA cards).

TPSCREEN lets you treat any chunk of memory as a virtual screen and write to it. Then you can move it to the actual screen. This chunk of memory could be a video page other than page zero (the default page), except on a mono card, which has only one page. TPSCREEN also has routines for scrolling a window horizontally or vertically.

The third screen unit is TPWINDOW. This gives you pop-up windows, which can be stacked on top of each other. You use Write and Writeln, and your output will automatically wrap and scroll within the window. There is also a FastWrite routine that confines itself to a given window. When a window is removed, the text in the window below becomes visible in the way you'd expect. Each window requires 40 bytes of overhead plus a buffer that's the same size as the window's capacity (2 bytes for each character, as there's the ASCII code and also the attribute). So a 10 by 10 window would need 240 bytes, all of which is kept on the heap. The demo program WINWOW shows what can be done with these windows, and I was duly impressed.

With this package the classic "Hello World" program can be written as a pop- up in just 30 lines of code.

TPMENU is the fourth screen unit. This lets you create Lotus-style horizontally scrolling menus, or else Borland/GEM/Windows style pull-down menus. You could also define a whole new style of menu if you were feeling inventive. From a user point of view, the menus are excellent and intuitive; selections are made using the cursor and Enter keys to point and shoot, or by single letter keystrokes if you want to set that up too. Pressing ESCape has the effect you'd expect; it backs you up one level in the menu tree.

A menu can have several items, and has a position on the screen, an orientation (horizontal or vertical), a style of frame, a set of colours, a title and a help row. Menus are kept on the heap, which leaves the limited data space available for the rest of the program. TPMENU is best used in conjunction with MAKEMENU, which means you don't even have to do any programming to set up your menus – just design them on-screen.

The last useful item in the screen unit is TPEDIT. This gives you a way to get a line of input from the user, and offer him a fairly sophisticated WordStar-type editor while entering the line, so that a mistype doesn't force him to retype the whole line.

String Handling

The TPSTRING unit includes routines that let you store strings on the heap, which is useful if you are doing a lot of string processing. There is also a hexadecimal output routine – mine is about five lines long, but I seem to rewrite it every time I write a program. T_Prof_4 adds Octal and Binary routines, and a nice little Soundex converter. Another unit, TPASCIIZ, gives long strings (up to 65520 characters). But the strings must be ASCIIZ (null-terminated) so you can't store a zero byte in them. ASCIIZ strings can be stored and retrieved from the heap, as otherwise they would soon eat up the available 64K of data space. All the usual string functions are available, including copy, insert, delete, read and write. TPCMDLIN gives you an easy way to parse command tails, so that tails like /R 3 /X /Size $12 can be acted on without a lot of tedious coding.

DOS and BIOS calls

Turbo Pascal already gives you easy access to DOS and Bios routines. T_Prof_4 makes it even easier. It gives you easy access to the DOS print queue; in particular it lets you submit a file for background printing. One of the best routines here is ExecDOS, which improves Turbo's Exec procedure, by acting as a front end to it. It also gives you routines to read and write absolute disk sectors; it took me hours to write this myself, because of the bug (Microsoft calls it an unusual interface) in DOS's Int 25h and 26h.

Interrupts and TSRs

It was always very flaky writing interrupt service routines (ISRs) and terminate and stay resident (TSR) programs in Turbo 3. Turbo 4 offers the programmer the ability to do it properly with the INTERRUPT directive. T_Prof_4 makes it easy, by wrapping everything up for you. The main problem with writing TSRs and ISRs is that DOS is not re-entrant because it maintains three stacks, and switches between them depending on what it happens to be doing at the time. With T_Prof_4, the classic Hello World program written as a pop-up is just 30 lines of code, and the manual gives this as an example.

You can choose any hot-key, choose whether you can pop up in the middle of DOS doing something or not, and your TSR can even offer the option to unload itself from memory when it is safe to do so. It also replaces the DOS critical error handler (the routine that gives the Abort, Retry, Ignore message) with an improved version. You can also make sure that your pop-up doesn't load itself twice, and you can cope with the 8087 by saving its registers before popping.

Once your routine is installed on an interrupt vector, you can call the old interrupt vector from your routine, or you can chain to it, which is useful if you want to pass something on to the old code.

When the hot-key is pressed, the pop-up tries to pop. If it can't (perhaps because DOS is busy), the timer tick starts counting down from the value you requested and, at each tick, it tries to pop. When it reaches zero, it gives up. Anyone writing TSRs should get T_Prof_4 for the sake of this module alone.

Sorting

TPSORT is very like the Borland Database Toolbox sort. You define three routines which input an element into the sorter, compare two elements and output elements.

Memory Allocation

T_Prof_4 has some very sophisticated memory management routines. The first of these is TPEMS, for managing expanded (LIM, or EMS) memory. You can allocate 16k EMS pages and the routine gives you back handles to refer to them with. Then, an EMS page can be mapped onto physical memory by using this handle, and you can use the data stored in that memory. If the concept of EMS memory still confuses you, see the article in March's issue of the magazine.

The extended memory routines work differently; they work in words of memory rather than pages, and you can allocate and deallocate extended memory, and move blocks of memory between extended memory and normal memory. The demo program DIFF shows the use of expanded and extended memory.

Macros

When you write a program, it is very handy if you can give it an autopilot facility, so that you can define macros, give them names, record them, play them back and edit them. The TPMACRO unit lets you have the macro facility in your programs, and the TPEDIT unit adds the editing capability. There is a macro engine running underneath which is started off with a procedure call, and then macro recording can be switched on or off with procedure calls. Macros can be written to a file or read from one, triggered off, or cleared. TPMACED gives the user a full-screen macro editor. I once wrote a massive Fortran program, and added a macro facility afterwards because I realised too late that driving an interactive program can be boring when you have repetitive tasks to do. A macro facility like this means that you can just set it running and go out for lunch.

Large Arrays

Turbo Pascal 4 still has the annoying limitation of 64K of data space. T_Prof_4 uses the heap in normal memory to emulate large arrays, or it can use EMS memory for arrays of up to 8 MB (EMS version 3) or 32 MB (EMS 4). The slowest way to have big arrays is to use virtual memory, paging the array between disk and memory as required. The three methods use almost exactly the same calling conventions, so it will be easy to change a program to use whatever it finds available on the host computer.

These large arrays can be allocated, loaded from a file or stored to a file, and initialised to a value. Two routines let you put data into an element of the array and get data from an element. Arrays can only be one or two dimensional.

BCD Arithmetics

Turbo Pascal version 4 does not support Binary Coded Decimal (BCD) arithmetic. With version 3, you could buy a version of TP that used BCD for reals (giving 18 digit accuracy), but very few people used it. If you use TP4 with an 8087, you get 19 digits of accuracy, but you still get rounding errors, which can be unacceptable for some financial applications. Anyway, you can't assume that all computers have an 8087. TPBCD restores this lost BCD capability, except that everything is done using arrays of bytes to store the BCD number, and all arithmetic is done by procedures and functions. It is very clumsy to use compared with the old Turbo Pascal 3 BCD compiler, and if you are using that, I'd recommend that you don't upgrade to Turbo Pascal 4.

Others

TPERRHAN installs a run-time error handler which allows you to trap the Turbo error exit, and decide for yourself whether to exit to DOS, or try to recover. There are some errors that can't be recovered; stack overflow, heap overflow and anything to do with the 8087. TPINLINE is an assortment of inline macros, which is really just a piece of machine code inserted directly into the executable code by the Turbo compiler.

Conclusion

Is there anything missing? Yes, of course there is; whatever one programmer provides, another programmer will feel is insufficient. I wanted support for IEEE reals on machines without the 8087, and some support for communications would be useful. The PC's BIOS support for the asynchronous port is pathetic, and everyone writes their own buffered ISR. I've heard a rumour that this will be coming out of Borland soon, so perhaps this is why it is absent.

The other big thing missing from TP4 is a source level debugger, and given TurboPower's past ability in this area (TDebug plus) we can hope that something comes out Real Soon Now, although Borland are about to launch their definitive answer to Microsoft's Codeview.

Turbo Professional version 4 is a must for every Turbo programmer whose time is worth money. The source code is provided, partly because you can examine and learn from it, partly because every programmer can always improve every other programmer's code, and partly because if you upgrade your TP4 compiler, you'll need to recompile the units. But TurboPower didn't have to provide the source code of the utilities, and it's a token of their excellent attitude that they did.

The menu maker and windower will add polish to any program, the big array and big string handler will be useful to anyone with that kind of problem, and I confidently expect to see a horde of TSRs hitting the market, based on the T_Prof_4 TSR modules. Now even the Sunday programmers can write rock-solid TSRs that don't crash the system.

.EXE

Porting a multi-tasking system from MS-DOS to OS/2

It was early 1986 and, although he knew that a multi-tasking operating system from Microsoft was on the cards, Peter Coates used MS-DOS as the basis for his multi-tasking rendezvous system. When OS/2 finally arrived, it was time to port the system across and it was decided to use a DLL. What lessons were learned in the process?

Monday night at about 7.30 there was a phone call. It was David. "I've arranged a really important demo of SNADS for Thursday week". I was going to be down in London anyway that week. "Good", I say. "I was just thinking how good it would look if we could show it working under OS/2", says he. This is a system which I had been working on for over a year. "Do you think you could get something working by then if you use the PS/2 we've got down here: you'll have Monday, Tuesday and Wednesday?".

"You are out of your tiny mind", says I. So I drive down to London the following day, and spend seven days converting a convincing subset of the system to run under OS/2. Several very valuable lessons were learned, as SNADS (System Network Architecture Distribution Services) is not a trivial system. It involves several independent tasks running concurrently, and the first job was to look at our interprocess communication mechanism under DOS and BackTrap (see separate box), and the IPC facilities under OS/2 to determine what direction to go.

When we first started on SNADS-PC, we knew that IBM and Microsoft were eventually going to announce a new operating system for 286 based machines, and we also knew that DOS was an unnatural host for a system like SNADS, but we wanted to get on with it. We (like, no doubt, many others) tried to guess what sort of facilities the new operating system would have. Unfortunately, we guessed wrong. I thought that all the Ada hype of recent years would affect their thinking and that a rendezvous mechanism would be provided, and so that's what we put into our little multitasking system (BackTrap) under DOS. I also thought that rescheduling would be cooperative rather than preemptive. How wrong can you be?

The rendezvous mechanism actually has certain attractions. Pipes and their big brother, queues, are all very nice but they are very one-directional. Often when you have a server process you not only want to give it information, but also to get information out: in SNADS-PC there are several named servers with anonymous clients. This is quite easy to do with a rendezvous, as shown in Figure 1. This is the basic mechanism behind the entire system. The question was whether to restructure the whole system to use queues (named or otherwise), or pipes, or semaphores, or to implement the rendezvous mechanism under OS/2; and if so how?

The approach we chose was to implement the BackTrap rendezvous mechanism, and to do this we used a Dynamic Link Library (DLL) and the signalling was done using memory semaphores, with the information being passed in shared memory segments. The basic form of this proved to be relatively straightforward, the problem remaining being that other assumption about cooperative rescheduling, which had, I'm embarrassed to admit, permeated the SNADS code. Under DOS, BackTrap only reschedules when the currently active process issues an INT 21H, INT 20H or INT 16H. OS/2 can reschedule at any time.

Creating a DLL

One of the most irritating and unnecessary difficulties I encountered was the misleading nature of the example DLL in the OS/2 Developer's Toolkit. I was following the C example as most of BackTrap and SNADS is in C. I don't like writing in assembler: I lack the necessary self discipline. There is enough rope in C to do practically anything, but in assembler I'm all too likely to get in an awful tangle. However every now and then a little assembler glue is needed, but it is powerful stuff. DLLs need an assembler main entry point. This entry point doesn't need to do much, and the sample one looks pretty generic, and even works for the supplied example. It is, however, wrong. Messing around with linkers and segments is as much fun as clearing a blocked drain. This is exactly the

Peter Coates is a software consultant and author of BackTrap. He can be contacted at Whincroft, Upper Colquhoun St, Helensburgh, G84 9AQ.

Figure 1 – Implementation of a rendezvous

```
Server                  client

suspend .
        .
        .               id = who_is(by name)
                        resume(id, info)

(suspend returns id of client and info)
                        suspend
resume(id, info)
                        (suspend returns info)
```

part that the sample gets wrong. As written, the sample causes the DGROUP segment to be concatenated with the code segment and therefore to be read-only. This is not what is generally wanted. The solution here was to combine the logic of the sample with the segment directives of the normal C startup code. A little more assembler glue is needed to make DS point to the DLL's data segment on entry to each DLL routine, and to restore the user's DS on exit. This is a very simple bit of code and is shown in Figure 2.

Using this code it is possible to code DLLs in C without too many special considerations. However, it must be noted that the compiler's assumption that DS=SS is not valid, so bad code will be generated if you attempt to take the address of any automatic variable. Also, if you want to return a value from a DLL routine, the expression in the return statement must not refer to any static or external variable or to any constant, as you have to go through the exit routine to restore the user's DS before the return statement. It wasn't until somewhat later I found out about the /Au switch on the compiler which must be designed for this purpose: it assumes that DS is not equal to SS, so it loads DS on entry to each subroutine and restores it on exit. However if you use the /Au switch, all pointers are 32 bits wide and, besides producing less efficient code, it makes it very hard to use standard library routines.

While on the subject of coding caveats, here is another one: it is not uncommon in C to have a procedure return a value in an address passed as a parameter. Often the address may be NULL indicating that the user is not interested in that value. Obviously address parameters passed to DLLs have to be far pointers. The IBM (Microsoft) C compiler correctly coerces the constant NULL into the far pointer NULL, but not an expression evaluating to NULL (such as a variable whose value is NULL). This gets converted to DS:0000, which looks like a perfectly reasonable, non NULL, pointer. It is up to you to work round this one, but you have been warned!

Implementation of the Rendezvous mechanism

The data structure behind BackTrap is called a context, mainly because that is what it is in the DOS version, where it contains things like registers and interrupt addresses. The OS/2 context structure is a much more modest object containing little more

One of the most irritating difficulties I encountered was the misleading nature of the example DLL in the OS/2 developer's toolkit

than a buffer pointer and length, a name, and a few RAM semaphores. Named tasks (servers) give themselves a name by introducing themselves, and keep their contexts until they are freed when the tasks exit (using an "exit list" routine). Unnamed tasks (clients) are given contexts as needed, and these contexts are freed typically on return from BackTrap. Active contexts are kept in a double linked circular list. It's double linked for ease of deleting contexts, and circular because of a hangover from the round-robin scheduling in the DOS version.

How a rendezvous works

The rendezvous works like this: a client task issues a resume giving the id of the correspondent task, a message buffer, the length of the message, and a time out value. Resume puts the calling task's context in the rendezvous queue of the specified task. A shared memory segment is allocated to the target task (to be freed, a shared memory segment must be freed by every task which can address it). The target task's suspended semaphore is then cleared, the calling task's waiting semaphore is set, and then resume waits for the waiting semaphore to be cleared, indicating that the rendezvous has completed. The wait for semaphore specifies a time out as given by the user, and so may fail, in which case resume has to abnormally terminate the rendezvous by removing itself from the target task's queue. Resume then returns either -1 in the case of a time out, or the length of message actually accepted by the target task. Meanwhile, before this, or while the resuming task is waiting on its waiting semaphore, the server task calls suspend specifying a message buffer, the length of the buffer, a time out, the address of an integer to receive the length of the message received (<= the length of the buffer), and the process id of the task we want to hear from. Suspend first checks to see if its rendezvous queue is empty, and if it is it sets its suspended semaphore and waits for it to be cleared. Then, knowing that there is at least one task on the rendezvous queue, they are each looked at to see if they meet the rendezvous criterion.

Here, BackTrap is more limited than Ada: it allows a server task to specify an open rendezvous, ie accepting any resuming task, or a specified task. Ada allows a list of acceptable tasks, but we did not see any need for that. If none of the tasks in the queue is the

```
on entry:
        pop     bx          ; return address
        push    ds          ; save users DS
        mov     ds,DGROUP   ; DS now points to DLL DGROUP
        jmp     bx          ; return

on exit:
        pop     bx          ; return address
        pop     ds          ; users DS
        jmp     bx          ; return
```

Figure 2 – Ensuring that DS points to the DLLs data segment. multi

one we want, then again the task gets blocked on the suspended semaphore. If a wait-for-semaphore times out, then suspend returns -1 to the calling task. When a rendezvous is accepted, the content of the shared segment specified in the context of the resuming task is copied into the used buffer and freed, and the id of that task is returned.

Obviously suspend and resume were implemented entirely differently under DOS, but the important thing is that the interface is practically identical. Thus once I had a stable multitasking environment which behaved the same on both systems, the port of SNADS began to look possible. Indeed other than that part of the system dependent on Extended Edition (Get used to that refrain, you're going to hear it a lot), the system went straight across with no difficulty other than some embarrassing difficulties with an EBCDIC to ASCII / ASCII to EBCDIC conversion routine. In fact I had a demonstrable and apparently stable system up and running by Tuesday evening. Of course it fell over when we showed it at the demo, but at least the DOS version behaved itself...

Final thoughts

So what do I think of OS/2 after the event? Well for one thing it is a whole lot easier to debug multi-tasking systems under OS/2. This must come as a surprise to someone. The second thing is how remarkably stable OS/2 was while I was using in earnest features which probably have not been used much yet.

I certainly have some gripes, like not being able to attach CodeView to more than one task at a time, and not being able to access symbolic information in a DLL. It would be churlish to complain about the documentation: it is after all so much better than DOS, but the area of DLLs is liable to become a bit like that of Terminate-and-Stay resident routines – an area of unwritten expertise – unless they upgrade the documentation there. It seems likely that a lot of software for OS/2 will become available very quickly if my experience of porting is anything like typical.

.EXE

What is BackTrap

BackTrap is a Multitasking add-on to DOS and a Dynamic Link Library on OS/2 offering a set of portable rendezvous primitives. Under DOS it does its own cooperative rescheduling, but under OS/2 it uses the interprocess facilities available. Under DOS, more or less as a side effect, it also provides command line editing and a History function. The primitives provided are as follows:

```
task_handle   BTP_who_is(char *name);
```
Get process id of a named task.

```
task_handle BTP_who_am_I(void);
```
Get current process id.

```
int   BTP_introduce(char far *name);
```
Give a task a name.

```
int   BTP_resume_to(task_handle task,
                    char *msg,
                    int len,
                    int time_out);
```

Resume execution of a suspended task. The data passed to resume is copied to the address which the target task passes to suspend. The amount of data copied is the smaller of the length given to resume, and the length given to suspend. The number of bytes actually copied is returned. If the target task has not issued a suspend, the calling task is stopped until it does. If the target task doesn't exist, or exits before issuing a suspend then resume will return with a return code of -6, invalid handle. If other tasks are already waiting to resume a suspended task, this task is added to the back of the queue. If time_out is not zero then the rendezvous window is only open for time_out ticks, after which resume fails with a return code of -1, not ready.

```
task_handle   BTP_suspend1(char *buffer, int len,
                    task_handle task,
                    int time_out,
                    int *rlen);
```

Suspend execution of a task pending a resume message. If there is a message pending, then the rendezvous will complete immediately, otherwise the task will be suspended. Suspend returns the task handle of the task passing the message. If task is non zero then suspend will only accept resumes from the task specified. If time_out is non zero, then the task only remains suspended for the number of ticks specified, and if no resume has been received in that time then suspend fails and returns -1. If len is not NULL, then the integer pointed to is set to the number of bytes actually copied.

```
void   BTP_delay(int period);
```

Prevent the task from being scheduled for the time specified. The time is specified in ticks, or 65536/1193180ths of a second – about 18.2 ticks per second under DOS and a very similar unit under OS/2. The actual delay will be longer than this as the task only becomes available to be scheduled after that time. There could be an additional delay of arbitrary length depending on what other tasks are running.

```
void   BTP_pause(void);
```
Stop the current process from being scheduled until either an event flag is set, or a resume is pending

```
void   BTP_setflag(task_handle task, int mask);
```
Set an event flag.

```
void   BTP_clrflag(task_handle task, int mask);
```
Clear an event flag.

```
int   BTP_tstflag(task_handle task, int mask);
```
Test an event flag.

```
int   BTP_probe(void);
```
Returns TRUE if a resume is pending.

```
void   BTP_lockin(void);
```
Stop all BackTrap events (stop all rescheduling under DOS). This call is necessary in OS/2 before tstflag and probe to prevent there being any critical exposure.

```
void   BTP_reschedule(void);
```
Signal a reschedule point under DOS, and re-enable rescheduling, ie the opposite of lockin for both DOS and OS/2.

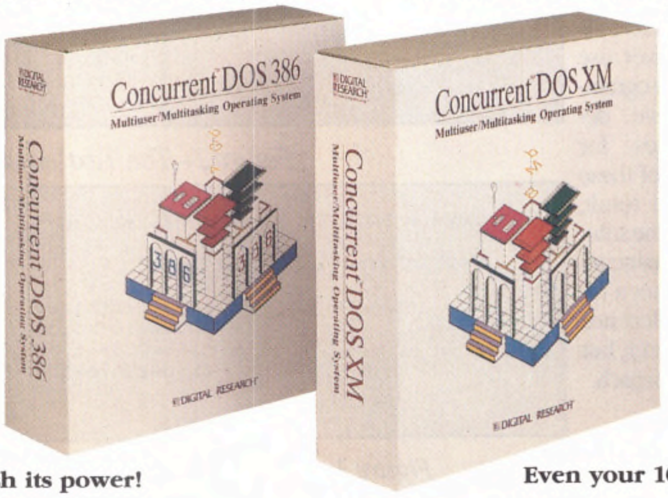
Turn your PC into a Powerstation

With the multiuser multitasking Concurrent™ DOS family

Concurrent DOS 386

Allows you to harness the full potential of the Intel 80386 microprocessor. As a 386 PC user. As an OEM. As a system builder. Concurrent DOS 386 uses the 80386 architecture to allow up to four gigabytes of address space. Provides compatibility with popular PC DOS single user and Concurrent multiuser applications. The standard retail pack turns your 386 machine into a central powerstation distributing high performance support to 3 users while multiuser packs let you configure for up to 10 users.

Choose the 386 and unleash its power!


Concurrent DOS 386
Multiuser/Multitasking Operating System
DIGITAL RESEARCH
Concurrent DOS XM
Multiuser/Multitasking Operating System
DIGITAL RESEARCH

Concurrent DOS XM

Gives extended capability to users working with Intel 8086/80286 based machines. It allows you to run popular PC DOS programs simultaneously as well as a wide choice of multiuser Concurrent software including accounting, database, word processing or project management. You can take full advantage of expanded memory cards to work in up to 8 Megabytes of address space. Standard retail packs provide 3 user support while 6 user configurations are available via multiuser packs, from our Authorised Concurrent Dealers.

Even your 16-bit PC can be a Powerstation.

The Concurrent DOS Family

- Provides full multiuser multitasking.
- Runs existing PC DOS applications.
- Easy to manage system operation. No system manager needed.
- Toolkits available for ISVs, OEMs and VARs for system building and programming.
- Runs the same single or multiuser software on 8086, 80286 or 80386 PCs without change.

Concurrent DOS derives from Digital Research's decade of experience developing multitasking, multiuser operating systems for microprocessor-based designs. Telephone (0635) 35304 for your information pack.

DIGITAL RESEARCH CIRCLE NO 8

String Handling in Modula-2

Steve Collins and colleagues at RTA conclude their explanation of how they implemented a word processor for the PC using Modula-2. This month, they outline the processes that need to be written, and give some starting points for a centre-justification routine. There is also a discussion of how best to represent a page of text in memory.

In last month's article we designed and implemented the string type and basic string handling functions. This month the intention is to implement a very basic word processor using the principles of Data Abstraction and to show the use of the string handler as a building block in the final system.

For many, the mere mention of the terms Data Orientated Design or Data Abstraction can induce an automatic shutdown and an early lunchbreak. This is fair comment as, on the whole, these buzzwords have been cheapened through over use and incorrect definition. Most programmers, through experience, have developed their own techniques for problem-solving. Although many of these methods do achieve the required result, the end rarely justifies the means. The solution is often overdue, the code unmaintainable and even inaccurate. This is not a reflection of the programmer's intellect nor their capacity to program a computer, but has more to do with the design approach.

In the days of unstructured languages, a programmer's merits lay in his or her knowledge of the underlying machine architecture – how, if you used such and such command, this would use register A instead of the big register B. This was fine as far as the machine was concerned, but made a real pig's ear out of the source code

RTA is a software house specialising in Modula-2. They can be contacted at Canning House, 59 Canning Road, Croydon, CRO 6QF. RTA will supply full source and executable code for the word processor in return for an SAE and a blank, formatted disk.

Figure 1 – The centreJustify function.

```
PROCEDURE centreJustify ( size       : CARDINAL;
                          pageOfText : Text       ) : Text;

(* this function pads each line in the text with spaces until the
   line matches the value specified by size :

  Thus -

  GIVEN    text = ' The Sisters
                    Of Mercy       '

       size = 14

  RETURNS the text ' The    Sisters
                     Of      Mercy '
 *)
```

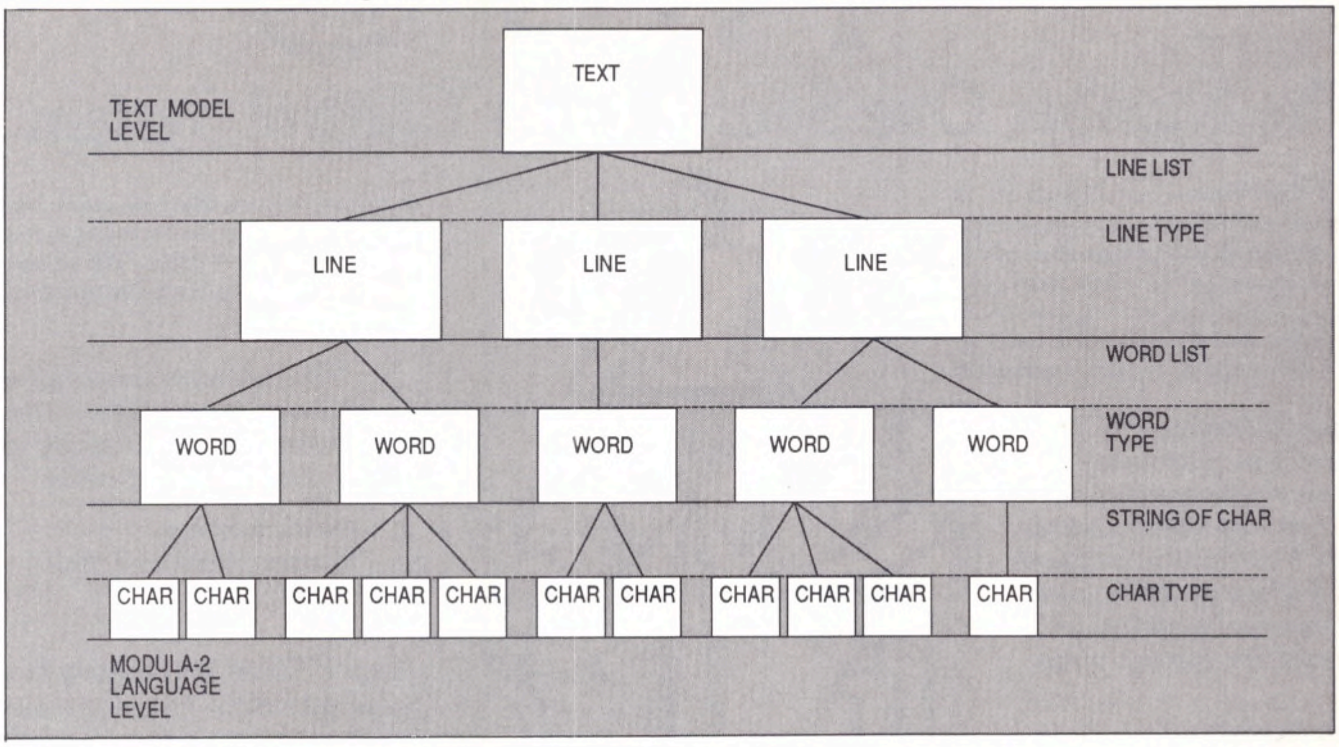
Figure 2 – The 'Text' data type

```
Text = POINTER TO TextInformation;

TextInformation = RECORD

                      NumberOfLines,
                      NumberOfWords,
                      SizeOfText       : CARDINAL;
                      LinesInText      : LineList;

                   END;
```

Figure 3 – LineList data type containing Line element.

```
 LineList = POINTER TO ListOfLine;

ListOfLine = RECORD
                 line        : Line;
                 RestOfList  : LineList;

              END;

Line = POINTER TO LineInformation;

LineInformation = RECORD

                      LineNumber,
                      NumberOfWords,
                      SizeOfLine     : CARDINAL;
                      WordsInLine    : WordList;

              END;
```

Diagram 1 – Levels of abstraction in a wordprocessor model



and was a nightmare to debug and maintain.

It is now the late 80's – machines are faster and more powerful than ever, but there are still many programmers who stick to the old school of thought. The software world can no longer afford this type of programming approach – it must keep up with the more scientific hardware world if we are to see more productive and creative computing in the 90's.

Data Orientated Design is part of a far larger design philosophy – Software Engineering. The aim of Software Engineering is to take programming out of the world of Confusion and into the world of Science.

The Data Orientated Approach

The problem we wish to solve is the implementation of a basic word processor. Owing to limited space we shall concern ourselves with the centre justification function (see Figure 1) and the actual modelling of a page of text.

It is tempting to functionally decompose the centre justify function. True, it is possible to write this using the string handler functions already defined (see last month's article), but the problem arises from the fact that these functions operate on a model of a string – StringOfChar – rather than on a page of text.

Figure 4a – List of words field.

```
    WordList    = POINTER TO ListOfWords;
    ListOfWords = RECORD
                        word        : Word;
                        restOfList  : WordList;
                END;
```

Figure 4b – WordInformation field

```
Word  = POINTER TO WordInformation;
 WordInformation = RECORD

                      WordNumber,
                      WordPosition,
                      WordSize,
                      WordSpace      : CARDINAL;
                      WordString     : StringOfChar;
          END;
```

We need to construct a model of the text before we think about the functions that act upon it. This is not to say we should not be aware of the functions when designing the model – data and functions go hand in hand. Rather it is saying that the data structure must be established before you attempt to perform any operations upon it – introducing the actors before the plot, if you like.

At the most basic level a piece of text is a hierarchy of data in a determined order. Text is a sequence of lines, a line is a sequence of words and a word is a sequence of characters. In design terms we may use each stage in the hierarchy as a new level of data abstraction. Thus the text data object is built from the line data object, which in turn is built from the word data object, and a word may in part be represented by a string – StringOfChar.

This is not the whole story. In our original definition, text is a sequence of lines. Our design only allows for one line, so what is needed is a method of representing a sequence of lines. This can be done using a dynamic data structure such as a list, the pros and cons of which were discussed in last month's article.

Diagram 1 shows the levels of abstraction of the Modula-2 data structures we are defining. Text may be represented as a list of lines. Before taking this any further, it may be worth our while to consider some other more 'functional' attributes of text, for example size, which can be represented us-

PVCS

The Number One Source Code Control System.

The POLYTRON Version Control System (PVCS) simplifies and automates Configuration Management so programmers and managers can effectively control the revisions and versions of source code. PVCS is the most widely used change control product and is used by the leading software, aerospace, manufacturing and service companies.

"In terms of features, PVCS provides everything necessary to a large multi-programmer project — more than any other package reviewed. No restrictions are placed in the development environment and all aspects of operation can be customized for specific project needs."

PC Tech Journal
September 1987

Unmatched Flexibility

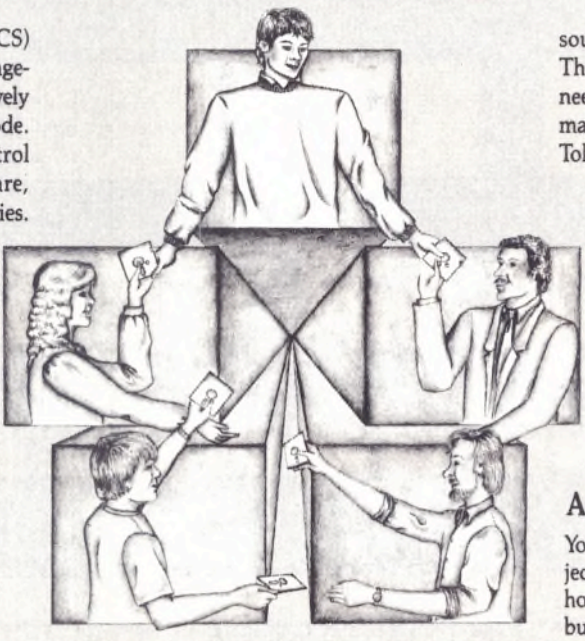
- Storage & Retrieval of Multiple Revisions of Source Code
- Maintenance of a Complete History of Changes
- Control of Separate Lines of Development (Branching)
- Resolution of Access Conflicts
- Optional Merging of Simultaneous Changes
- Release and Configuration Control
- Project Activity Reports
- Management Reports
- Command or Menu Interface

Project Control

PVCS maintains individual archieves of all project components in your system — source code modules, data files, documentation and even object code libraries. These "source documents" can be written in any language or multiple languages.

Fast Retrieval of Revisions

PVCS uses "reverse delta storage" which saves disk space and speeds retrieval of versions of any file in the project database. A delta is the set of differences between any revision and the previous revision. PVCS can rapidly recreate complete versions of any file whether it is the most recent revision of a module or the original version of the entire project. Differences are automatically detected and stored.



A Practical Necessity for LANs

While important for single-programmer projects, PVCS is absolutely essential for multiple-programmer projects and LAN-based development efforts. In a LAN environment, source code files are simply too easy to change. Because any change to any file can have major ramifications, coordinating and keeping a record of changes is critical. Project leaders can determine, on a module-by-module basis, which programmers can access or modify source files, libraries, object code and other files. The levels of security can be tailored to meet the needs of nearly every project. PVCS works on all major LANs including 3Com, Novell and the IBM Token Ring Network.

"PVCS has helped us maintain nearly 90 programs and utilities. Without it we would not have the quality of our upcoming release of NetWare."

Jonathan Richey
Manager, NetWare Utilities
Novell

Adopt PVCS on Your Existing Projects

You can obtain the benefits for your current project without disrupting development, regardless of how long your project has been under way. You can build PVCS archives from revisions stored in your present files or simply adopt PVCS from the current date.

PolyMake Reads PVCS Logfile Format

PolyMake, the leading Make utility, understands the structure of PVCS logfiles and is able to correctly determine the date and time of any revision. This prevents unnecessary operations that occur when the date and time of the complete project archive itself is used as with other make utilities.

Once you standardize on PVCS, the archives used to track and monitor changes are interchangeable between any PVCS product.

Personal PVCS — Offers most of the power and flexibility of Corporate PVCS, but excludes the features necessary for multiple-programmer projects.

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes multi-level branching to effectively maintain code when programs evolve on multiple paths.

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project.

PVCS for VAX systems — Requires VMS. Uses the same interface and archive format as MS-DOS version. Supports branching and offers file locking and other security features for multiple-programmer projects.

$1.75=£1.00	MS-DOS*	VMS		
	PC/XT/AT	Micro VAX II	VAX 7xx	VAX 8xxx
Personal PVCS	$149			
Corporate PVCS	$395			
Network PVCS	$995**	$4,950	$9,500	$10,500+
PolyMake	$149			
Network PolyMake	$447**	$1,250	$2,375	$2,500+

**5 Station LAN License. Call for pricing on larger Networks.

TO ORDER:
(027 979) 2566
The Software Construction Company
28 Churchmead, Roydon,
Harlow, Essex CM19 5EA

ALL PRODUCTS EX-STOCK

POLYTRON®

High Quality Software Since 1982

TURBO PASCAL

CODE GENERATORS

IMPEX/Query	£79.95
REPORT BUILDER	£79.95
System Builder	£99.00

COMMUNICATION TOOLS

Turbo ASYNCH PLUS	£69.95
COMPILERS	
Turbo Pascal v.4.0	£55.00

DATA ACQUISITION & CONTROL

MetraByte Dash-8 and Dash-16	£99.00

DATABASE TOOLS

Btrieve	£180.00
Turbo Database Toolbox	£49.00
TURBOSmith	£69.95

DEBUGGERS

TDebugPLUS	£35.00
Tmark	£69.95

DEVELOPMENT ENVIROMENTS

Alice: The Personal Pascal	£89.95
DOS TOOLS	
DOS/BIOS/Mouse Tools	£49.95

EDITORS

Turbo Pascal Editor Toolbox	£49.00
XTC with Pascal source	£75.00

GAMES TOOLS

Turbo Pascal Gameworks	£49.95

GENERAL TOOLS

KeyPlayer	£49.95
Mach 2	£69.95
Turbo POWER TOOLS PLUS	£69.95
TURBO Advantage	£49.95
Turbo-Plus	£59.95

GRAPHICS TOOLS

HALO	£175.00
TurboHALO Universal Graphics	£89.95
TurboHALO Graphics	£89.95
Turbo Pascal Graphix Toolbox	£59.95

ON-LINE GUIDES

The Norton Turbo Pascal Guide	£99.00

SCIENCE & ENGINEERING TOOLS

Science & Engineering Tools	£69.95
Statistical Toolbox	£69.95
Turbo Pascal Numerical Methods	£69.95
TURBO Advantage Complex	£59.95

SCREEN TOOLS

TURBO Advantage Display	£69.95
Screen Sculptor	£89.95
Speed Screen	£35.00

TEXT PROCESSING TOOLBOX

Lightening Word Wizard	£49.95

TUTORS

Turbo Pascal Tutor	£29.00

UTILITIES

Turbo EXTENDER	£55.00
Turbo Optimizer	£45.00
Turbo Optimizer with source code	£89.95
Turbo Spawn	£39.95
TURBOPOWER UTILITIES	£49.95
TURBOPOWER with source code	£69.95
TURBOREF	£49.95

WINDOW TOOLS

Flash-Up Windows	£59.95
Flash-Up Developer's Toolbox	£39.95
FontWINDOW	£69.95
FontWINDOW/PLUS	£185.00
MetaWINDOW	£110.00
MetaWINDOW/PLUS	£185.00
TurboWINDOW/Pascal	£55.00

BASIC LANGUAGE

Microsoft QuickBASIC	£60.00
Turbo Basic	£60.00

C COMPILERS

Borland Turbo C v.1.5	£60.00
Lattice C v.3.2	£199.00
Microsoft C v.5.0	£250.00

COMMUNICATION LIBRARIES

Greenleaf Communications v.2.04	£90.00
Multi-Comm	£120.00

DATABASE LIBRARIES

Btrieve	£180.00
db_Vista	£160.00
dBC III v.2.0	£160.00

DEBUGGERS

Advanced Trace 86 v.2.3	£115.00
C-SPRITE v.2.02B	£105.00
Pfix-86 Plus	£220.00

EDITORS

BRIEF v.2.0	£140.00
dBRIEF v.1.3 (Requires BRIEF)	£70.00
Pmate	£115.00

C FUNCTION LIBRARIES

C TOOLS PLUS v.3.0	£110.00
C Utility Library	£105.00
Greenleaf Functions	£90.00
PforCe v.1.03	£230.00

GRAPHICS LIBRARY

ADVANTAGE Graphics v.1.0	£250.00
Essential Graphics v.1.1	£155.00
GraphicC v.3.02	£255.00
HALO v.2.26	£175.00

LINKERS

Plink86+ v.2.22	£255.00

MAKE UTILITIES

PolyMake v.2.1	£85.00
Pmaker v.1.1	£80.00

RPG II

Lattice RPG II	£750.00

SYNTAX CHECKERS

PC-Lint v.2.00G	£80.00
Pre-C v.1.36	£175.00

TEXT MANIPULATION

C Beautifer	£29.95
PolyXREF	£124.00

TRANSLATORS

Bastoc (Basic to Lattice C)	£395.00
dBx (dBASE III+/Clipper to C)	£250.00
FOR_C (Fortran to C)	£495.00

UTILITIES

C SCAN	£195.00
C DOCUMENTOR	£195.00
MKS Toolkit v.2.2	£99.00
Norton Utilities Advanced	£95.00
PolyLibrarian II v.2.0D	£149.00

PLEASE NOTE

This is only a small selection of our extensive range of Software Development Tools please call for complete list.
All prices exclude V.A.T., but include first class delivery by recorded post. Always call for latest price on any product.

FACT SHEETS

Detailed Fact Sheets available on all products listed.

CALL FOR 8 PAGE TURBO PASCAL CATALOGUE

UNIX Tools on DOS MKS Toolkit v.2.2

Over 110 programs that perform tasks on machines like the IBM PC with the ease that one would expect while working under UNIX. Designed especially for those developing software in a DOS enviroment.

Only £99.00 excluding V.A.T.
Call for four page fact sheet

PVCS

The Most Powerful & Flexible Source Code Revision & Version Control System
Personal PVCS £99.00
For single-programmer projects
Corporate PVCS £225.00
For larger, multiple-programmer projects
Network PVCS from £570.00
CALL FOR CATALOGUE

A COMPLETE DATA ACCESS SYSTEM C-INDEX+ WITH SOURCE CODE

- Variable length keys and data
- True multi-user
- Keys and data in one files
- Compile and run-time definition
- Automated multi-key storage
- Written in K&R C
- Complete with full source code
- No royalties

ONLY £175.00
CALL FOR FACT SHEET

THE SOFTWARE CONSTRUCTION COMPANY LTD.

28 CHURCHMEAD, ROYDON, HARLOW, ESSEX CM19 5EA

TELEPHONE (027 979) 2566 FAX (027 979) 2609

CIRCLE NO 9

Figure 5 – The data constructor functions

```
For the main levels:

 Constructor functions

  - where the functions take the component fields and construct
    the relevant types.

    PROCEDURE mkText ( numberOfLines,
                       numberOfWords,
                       sizeOfText      : CARDINAL;
                       linesInText     : LineList ) : Text;

    PROCEDURE mkLine ( lineNumber,
                       numberOfWords,
                       sizeOfLine      : CARDINAL;
                       wordList        : WordList   ) : Line;

    PROCEDURE mkWord ( wordNumber,
                       wordPosition,
                       wordSize,
                       wordSpace       : CARDINAL;
                       WordString      : StringOfChar ) : Word;

For  each  of the fields,  accessor functions must be written,

for example :

  PROCEDURE obtainWordString ( word : Word ) : StringOfChar;

  (* this obtains the word string field in any given word *)

  PROCEDURE obtainWordSize ( word : Word) : CARDINAL;

  (* returns the value stored in the word size field *)

For the mezzanine levels:

To add a single 'element' to the relevant list:

   PROCEDURE prefixLineToLList ( lineToBePrefixed : Line;
                                 lineList   : LineList ) : LineList

    PROCEDURE prefixWordToWList ( wordToBePrefixed : Word;
                                  wordList : WordList) : WordList;

And to access the data in the lists:

 PROCEDURE firstWordInWList ( wordList : WordList ) : Word;

 (* head of list *)

 PROCEDURE butFirstWordInWList ( wordList : WordList) : WordList;

 (* tail of list *)

Likewise for LineList.

Thus to access the actual word string from the second word  in
the list :-

 obtainWordString (firstWordInWList (butFirstWordInWList (word)));
```

ing various units. In this design we use size in lines, words and characters (see figure 2). It is worth noting that the pointer type is used in this case not to build a list structure but to reduce data redundancy when moving pages of text around (see last month).

The Text data type is the top level of our abstraction; along with the various size fields it contains a list of lines. The line list is a list containing any number of 'Line' data types, where a Line is a list of words along with additional information concerning size, (see figure 3). Also included in the Line data type is a Line number field. This comes in useful when a line is extracted from a large page of text and the position of the line is required. The WordsInLine field is constructed in much the same manner as the LinesInText field in the Text data type, (see figure 4a).

The Word data type contains a string, StringOfChar, that consists of the actual stream of characters that represents a word. Also included is the word number, useful for finding a word in a line. The position of the word in a line is useful when adding or removing words. Word size is in characters. An interesting field is the WordSpace field: this plays an important part in justifying and padding out words, (see figure 4b.)

We have now designed our data types according to the data abstraction diagram (diagram 1). As in last month's article, each of these data types requires fundamental accessor and constructor functions, (see figure 5.) To summarise, we have three levels of abstraction: Word, Line and Text. Between each level there is a 'Mezzanine Level' – a dynamic data structure which is used by the next level. For example, LineList is used by Text. Each level is implemented in terms of its data.

Modula-2 offers the facility to place each data type and level in a separate module. The underlying structure of these modules is hidden using an opaque data type, thereby omitting the data structure from the definition module.

A Page of Text

At some point or other, when dealing with your own custom-designed Types in Modula-2, conversion functions are required. For example, it is not possible to pass the centreustify function (see figure 1) a string. The two conversion functions we require are stringToText and textToString (see Figure 6).

These functions are quite complex, so for the sake of simplicity we will spare the reader a fully coded explanation. In truth the actual conversion is straightforward. The difficult part arises in capturing the attributes of each datum. For example, when building a WordList the words must be in order and have the correct numbers etc. Figure 7 shows the pseudo code for constructWListFromString.

Designing the centreJustify function

We are now at the stage where we may design and implement our wordprocessor functions. We have a suitable model and some useful conversion functions, along with our string handler (see figure 8) for a definition module of some basic wordprocessor functions. These are very elementary; there are many more that could be implemented using our text model.

We now return to our original aim – to design centreJustify. The most basic definition of this function is – 'pad out the spaces between the words in a line, until the line is of the length specified by the size parameter'. CentreJustify works by going through the text justifying line by line until

THE TRAINING LEADERS

OS/2 PRESENTATION MANAGER

ONE DAY TECHNICAL OVERVIEW

- INTRODUCTION TO PRESENTATION MANAGER
- APPLICATION STYLE AND STRUCTURE
- PROCESSING OF DIFFERENT INPUT EVENTS
- TEXT, GRAPHICS AND PRESENTATION SPACES
- INTER-TASK COMMUNICATIONS
- OS/2 KERNEL
- IMPLICATIONS FOR EXISTING APPLICATIONS

1 DAY

20 JUNE-LONDON
25 JULY-LONDON
15 AUGUST
19 SEPTEMBER-LONDON

LEARN FROM THE EXPERIENCE OF THE LEADING AUTHORITY IN OS/2 TRAINING

OR, 5 DAYS HANDS-ON

13 JUNE 4 JULY
1 AUGUST
22 AUGUST
12 SEPTEMBER

OTHER COURSES
OS/2 · WINDOWS · NETWORKS · PASCAL · C
MASM · PC SUPPORT · XENIX · 286/386

QA TRAINING

CALL ALICIA SMITH ON 0285 5888

Help

Window

Microsoft Authorised

Figure 6 – The stringToText function

```
PROCEDURE stringToText ( stringToBeConverted : StringOfChar) : Text;

PROCEDURE textToString ( textToBeConverted : Text ) : StringOfChar;
```

Figure 7 – Pseudo Code for constructWListFromString

```
PROCEDURE constructWListFromString (string:StringOfChar):WordList;

    IF
         moreSpaces in string
    THEN
         prefixWordToList ( createWord ( calcWordNumber,
                                         calcWordPosition,
                                         calcWordLength,
                                         calcWordSpaces,
                                         getWordFromString),

           constructWListFromString(restOfStringWithoutWord )
    END;
(*positionOfSubString is used to locate 'words' in the string, by
  finding the position of the spaces between each word. Then
  passing the value to middleChars which abstracts the string thus

   middleChars (positionOfSpace,lengthOfWord,string)

  Where:
  positionOfSpace = postion of the space just before the word +1
                    i.e. the actual start position of the word.
   lengthOfWord = positionOfSpace minus the position of the space
                  just after the word . *)
```

Figure 8 – DEFINITION MODULE WPOperations;

```
(* IMPORT list .... *)

PROCEDURE centreJustify ( size    : CARDINAL;
                          pageOfText : Text ) Text;

PROCEDURE getLine ( lineNumber : CARDINAL;
                    pageOfText : Text ) : Line

PROCEDURE getWord ( lineNumber,
                    wordNumber : CARDINAL;
                    pageOfText   ) : Word;

PROCEDURE sizeOfText ( units : SizeUnits;
                       pageOfText : Text ) : CARDINAL;

(* SizeUnits = (chars,word,line) this is an enumeration type
   used for selecting which units the size is required in *)
```

Figure 9 – Padding a word with spaces

```
PROCEDURE addSpacesToWord ( numberOfSpaces : CARDINAL;
                            word           : Word      ) : Word;
BEGIN

  WITH word^ DO (* the record fields are dereferenced *)

     RETURN mkWord ( WordNumber,
                     WordPosition,
                     WordSize,
                     WordSpaces + numberOfSpaces,
                     WordString );

     (* mkWord reconstructs the word with alterations *)

     END; (* WITH *)

END addSpacesToWord;
```

there are no more lines left. To achieve this the LinesInText field must be accessed from the Text. It is then possible, using firstLineInLList and butFirstLineInLList (head and tailing), to move through the list removing a single line at a time.

Given that we now have our single line, we must access the WordsInLine field. This field is a list of the words and it is at this level that we can do the processing. The firstWordInWList function is used to access the first word in the list which is to be padded out with spaces. The rest of the list may be accessed recursively using butFirstWordInWList and firstWordInWList. Adding spaces to a word is a matter of incrementing the WordSpace field (see fig 4b). This may be conveniently done by using the addSpacesToWord function, (see fig 9).

One problem lies in calculating how many spaces are to be added to each word to pad the line out evenly. That is to say that, if a line is 23 characters in length and we wish to pad it out to 27 characters, it would be untidy to add 4 spaces to the first word only.

To solve this, we must calculate the number of spaces needed in total to pad the line out. This is obtained by (LineSize – size) where the 'LineSize' (see figure 3) is from the line header and 'size' is the first parameter in the centreustify function. The next step is to calculate the number of gaps in a line thus: (numberOfWords – 1). Note that the number of gaps is not the number of spaces, that is to say that if a line contains four words there are going to be three gaps – one between each word.

These values can now be passed to the work-horse function justifyWordList, (see fig 10). ustifyWordList evenly distributes the spaces by recursively adding a constant number of spaces, (totalSpaces DIV numberOfGaps), to each word. The remainder, if any, from the division is added to the line one space per word until there are no more spaces left to add.

Pros and Cons of iteration and recursion

Our original decision to use recursion was to demonstrate the elegant underlying structure of the data types. Recursive algorithms in themselves are very powerful and play an important part in functional specifications. It is unfortunate that many programmers tend to be biased against recursion for reasons best known to themselves. In saying this, we must also acknowledge that recursion may prove inefficient in some applications. Every function in the text suite has been defined recursively and most have been implemented in this manner. The main overhead when using recursive functions is that every time a function calls itself (recurses) the state of the function and its values must be recorded. So if there is a deep recursion then it may well be possible to run out of stack space.

This limitation is proving less and less of a problem in recent years with the advent of faster machines and more optimised compilers with better stack management. Nonetheless, a great deal of programming is still done on relatively small machines and this must be taken into consideration. Iteration offers a good and easy alternative to recursion. Iteration does not offer such an elegant solution to problems but the overheads are considerably less.

The difference between recursion and iteration is demonstrated by the length function (see Figure 11). The recursive solution better represents the data structure and – it may be argued – is clearer to understand. The problem arises from the fact that the function calls itself n times, where n is the length of the string and, therefore, requires n number of stack frames.

Conversely, the iterative solution requires less stack – the procedure is called once and only needs one stack frame. The iterative solution's Achilles' heel is that values must be mutable ie variables. This may lead to unwanted side effects.

As always in computing, there is a trade-off between good design and practical application. Again I return the reader to Algorithms+Data Structures = Programs by N. Wirth for some sound advice regarding this dilemma.

Figure 10 – Justifying an entire line

```
PROCEDURE justifyWordList ( numberOfGaps,
                            totalSpaces  : CARDINAL;
                            wordList     : WordList  ) : WordList;
BEGIN
   IF
       totalSpaces = 0
   THEN
       RETURN wordList
   ELSE
       IF
           (totalSpaces MOD numberOfGaps > 0)
       THEN
          RETURN
          prefixWordToWList (

               (addSpaces ( (totalSpaces DIV numberOfGaps)+1),
                             firstWordInWList(wordList)      ),

              justifyWordList (numberOfGaps-1,
                               (totalSpaces-
                               (totalSpaces DIV numberOfGaps)+1),
                               butFirstWordInWList(wordList)))
       ELSE
          RETURN
           prefixWordToWList (
             (addSpaces (totalSpaces DIV numberOfGaps),
                             firstWordInWList(wordList)),

              justifyWordList (numberOfGaps-1,
                               (totalSpaces-(totalSpaces DIV
                                                  numberOfGaps)),
                               butFirstWordInWList(wordList)));

       END; (* IF *)
   END; (* IF *)
END justifyWordList;
```

Figure 11 – The recursive solution:

```
  PROCEDURE length ( string : StringOfChar) : CARDINAL;

  BEGIN
    IF
        isEmpty(string)
    THEN
        RETURN 0
    ELSE
        RETURN length(tail)+1
    END;

  END length;

The iterative solution:

  PROCEDURE length ( string : StringOfChar) : CARDINAL;

  VAR numberOfChars : CARDINAL;
      refString     : StringOfChar;

  BEGIN

      numberOfChars:=0;
      refString:=string;

    WHILE moreChars(refString) DO
       INC(numberOfChars);
      refString:=tail(refString);
    END;

  END length;

  (* moreChars tests for the existence of any more characters *)
```

Overview

Having now designed and constructed our model of a page of text, the next stage is to prototype and test the application module in accordance with the original specification, with each function being tested separately. Given that the modules pass the testing stages, it is now possible to move to designing the user interface – screen layout, windows, cursor control etc. The user interface is a little more straightforward than the data model, and is designed interactively with the end user.

Most of the code already exists in various commercial Modula-2 libraries. The procedures and functions in these libraries are used initially to piece together a basic but effective interactive interface. The user is allowed to use the interface and pass comment regarding any alterations. The amendments are made and the new version is returned to the end user. Modula-2's concept of separate (reusable) modules is well suited to this type of 'iterative' prototyping, as it enables the programmer to cut and change the code without having devastating effects elsewhere in the code.

Final Comments

The overall design method has only been hinted at, yet we feel that we have made a point in showing that Modula-2 may be used effectively to implement well designed systems. Of course, an important point in an engineered approach is that the design ought to be language independent, but Modula-2 does prove, in our opinion, to be a more powerful tool than, say, Fortran. If you have any comments or arguments, please do drop us a line. EXE

Windows Programming

In this sixth part of his series, Jay Chapman continues to enhance his sample Windows application. This month, he explains how to recover text that has scrolled out of a window and starts to delve into memory allocation.

Where were we?

We left WINAPP.EXE last month with a new Scroll Window up and running but only partially, and with not too much finesse. We could type data into the window, delete the last character typed (up to the beginning of the current line) and move on to enter the next line by pressing the RETURN key, at which point the window contents scrolled up to leave space for the next line to be typed in. We also saw how the new window could be given a default size and position, and code was introduced to deal with the user moving or resizing the window. Finally, it was possible to scroll the window contents explicitly using the window's 'built-in' vertical scroll bar: the user could scroll a line or a half-page at a time either up or down.

Where does all the data go?

There remains a major problem for us to solve. Whenever any of the displayed text is scrolled off the top or bottom of the window's client area it is lost forever. Even if the window is immediately scrolled in the opposite direction all that gets scrolled back is an empty background. In considering why this is so we will go on to refine the design of the scrolling window. In particular we will have to arrange storage for the data that relates to any part of the screen display that will require painting again sometime in the future. This will lead us into a discussion of Window's memory allocation functions.

At the moment all scrolling is being handled by the Windows SDK ScrollWindow() routine. This routine is responsible for doing two things. Firstly, it copies the area of screen memory being scrolled into its newly mapped position (logically on the screen, physically in screen RAM). Secondly it causes a WM_PAINT message to be sent to the window procedure advising that the newly uncovered area of the client area needs repainting – this was described in detail last month.

The Model

Before presenting the solution to the problem let me generalise: we have some data that needs presenting on the screen. If it will only be displayed once and can then be forgotten life is very easy – this is what we did with the characters being typed in last month. Life is not usually so easy, however. It is fairly obvious that if we scroll up a line or page and then scroll down again we expect the data to reappear (unless our mental model of viewer and paper expects someone to go to work with an eraser every time the data/paper goes out of view). Also, if we close the window and then open it again we don't expect the data to have evaporated.

In a sense, therefore, we have to make the 'limbo' regions, ie those regions beyond the visible client area, into something more substantial. They must be capable of keeping a record of what is on the 'paper' even when we are not looking at it – if we are to implement our model of the 'viewer onto a large sheet of paper'.

Practical Implementation Restrictions

Making the implementation more realistic will also force us to make some decision about the maximum size of the 'paper' since if we have to store data we will be restricted by the finite amount of RAM available in the machine.

We will also have to place some restrictions on the scrolling permitted. We will only be able to view either data that exists in the present (the current input line) or the past (those lines that we have stored data for). That is to say we cannot view the future, so we must not allow the contents of the window to be scrolled any further up once the current input line is displayed on the bottom line of the window. Similarly we cannot view further into the past than that defined by the contents of the storage allocated, ie if we have storage space for 100 lines available we can only scroll the window contents down until the 100th line is shown in the top line of the display. Any more, and we have no data available to tell us what to paint.

Note that it is preferable to stop the scrolling as soon as the extent of possible data/paper is reached on the display. In other words we don't let the user scroll the window past such extents (perhaps displaying blank lines where there is no 'paper' to map from) since the 'metaphor' he is working with would then suggest that 'paper' exists where in fact it does not. Users are quite good at reshaping the metaphors being used a little, though. In our case not only does the paper we are viewing have a finite size but it is created almost 'under the pen' which is writing the characters

Jay Chapman is a lecturer and writer who specialises in the Human-Computer Interface in general, and Windows in particular.

Computer Aided Software Engineering

10,000 Sales = 10,000 Decisions

See our video and decide for yourself.

You now have the opportunity to see for yourself that Excelerator is a comprehensive, totally integrated CASE tool created specifically to help systems development professionals in the analysis, design and documentation of virtually any software system.

With over 10,000 copies sold, and 20,000 users worldwide Excelerator is now probably the most extensively used CASE workbench in today's marketplace.

Begin your evaluation today by calling for a copy of the Excelerator Video. The Proven CASE Workbench. It's available now.

For further details call Marion on 04427 76424 or fill in the coupon below.

Excelerator Software Products

a company of ***Index Technology Corporation***

Prince Edward Street Berkhamsted, Herts HP4 3AY. Telephone: 04427 76424.

Please send me a copy of the Excelerator Video. __________

Name: __________

Company Name: __________

Address: __________

__________ Post Code: __________

Position: __________ Tel. No: __________

.EXE 6·88

CIRCLE NO 11

```
 42  /* State of the scrolling window. */
 43  short nScrollClientDx, nScrollClientDy, nNumLinesInWindow, nCharDx, nLineDy;
 48
 49  /* Scrolling Buffer variables.    */
 50  HANDLE hMemScrollLine [ SCROLLBUFFERSIZE ];
 51  short  nScrollLineSize[ SCROLLBUFFERSIZE ];
 52  char   cinputline[ INPUTLINESIZE ];
 53  short  nBottomLineIndex = 0, nScrollOffset = 0, nCurrentColumn = 0;
 56
256  /*===============================*/
257    PaintScrollWindow(hDC, rcPaint)
258  /*===============================*/
259  HDC  hDC; RECT rcPaint;
260  /* Paint (at least) the rcPaint area of the Scroll Window. */
261  {
262  short nLineIndex, nLineY;
263
264  SetTextColor(hDC, RGB(0xFF, 0x00/*DEBUG*/, 0x00) /* Yellow */);
265  SetBkMode(hDC, TRANSPARENT);
266
267  for (nLineIndex = nNumLinesInWindow +
268                      nScrollOffset + nBottomLineIndex,
269         nLineY   = nScrollClientDy
                      - nNumLinesInWindow * nLineDy - nLineDy;
270       nLineIndex >=  nScrollOffset + nBottomLineIndex;
271       nLineIndex--, nLineY += nLineDy)
272    {
273    short  nScrollBufferIndex =
274              (nLineIndex >= SCROLLBUFFERSIZE
275              ? nLineIndex - SCROLLBUFFERSIZE
276              : (nLineIndex < 0
277                ? SCROLLBUFFERSIZE + nLineIndex
278                : nLineIndex
279                )
280              );
281
282    if (nScrollBufferIndex != nBottomLineIndex)
283      /* It's a line stored in a global memory block. */
284      {
285      HANDLE hMemCurrentLine = hMemScrollLine[nScrollBufferIndex];
286
287      if (hMemCurrentLine != NULL)
288        {
289        LPSTR  lpMem = GlobalLock(hMemCurrentLine);

292        TextOut(hDC, 0, nLineY,
293                      lpMem, nScrollLineSize[nScrollBufferIndex]
294              );
295        GlobalUnlock(hMemCurrentLine);
296
297        /* Code to demonstrate usefulness of painting rectangle. */
298        if (bGraphicsOnScreen ||
299              (nLineY + nLineDy > rcPaint.top     &&
300               nLineY            < rcPaint.bottom)
301          )
302          {
303          short nLineDx;
304
305          for (nLineDx = 20; nLineDx < 1000; nLineDx += 1)
306            {
307            MoveTo(hDC, nLineDx     , nLineY + nLineDy * 1 / 4);
308            LineTo(hDC, nLineDx + 10, nLineY + nLineDy * 3 / 4);
309            }
310          }
311        }
312      }
313    else
314      /* It is the line being built in the input buffer.       */
315      if (nScrollBufferIndex == nLineIndex)
316        /* If nLineIndex isn't the same as nScrollBufferIndex  */
317        /* then we have the 'worst case' situation where the   */
318        /* current 'last' line is the topmost full height line */
319        /* being displayed and the current input line might be */
320        /* 'vertically wrapped' (due to the nature of the cir- */
321        /* cular buffer) and be displayed in any partial line  */
322        /* above the topmost full line already mentioned.      */
323        /* If the partially displayed line is not the current  */
324        /* input line we can proceed to output the latter:     */
325        TextOut(hDC, 0, nLineY, cinputline, nCurrentColumn);
326    }
327  }
328
329  /*============================================================*/
330    long FAR PASCAL ScrollWndProc(hWindow, message, wParam, lParam)
331  /*============================================================*/
338  {
339  switch (message)
340    {
347
352    case WM_CHAR:
353      /* Before doing anything with the arriving character we must ensure */
354      /* that the window is not in a scrolled state.                      */
355      /* If the character is a carriage return then scroll the window up  */
356      /* one line and reset the current column offset to the left-hand    */
357      /* edge of the window. Copy the input line into a global block.     */
358      /* If the character is a back space then move back to the previous  */
359      /* column (unless in column zero) and overwrite the character there */
360      /* with an OPAQUE space to erase it.                                */
361      /* Otherwise display the character at the current column offset in  */
362      /* the bottom line of the scroll window.                            */
363      if (nScrollOffset != 0)
364        {
```

Figure 1 – additions to WINAPP.C.

into the bottom line of the window and it disintegrates just past the current 100th (or whichever) line. In fact it's sort of hyper-dimensional.

A final comment on the scrolling concerns the elevator on the scroll bar. When the window is fully scrolled in either direction, as discussed above, the elevator will be hard up against either the top or the bottom of the scroll bar. This will reinforce the user's awareness of the fact that he can scroll no further.

Full or Partial (Re)Painting

If another window covers our window and is then removed, what happens? Well, Windows can redraw all of the non-client area of the window (caption bar including title, border, mouse sensitive control points). It can even fill in the background colour of the client area. There is no way that it can recreate what you had caused to be drawn in the client area, however, because it has no knowledge of your data, its format, or its mapping onto the screen.

One last little problem: imagine that the window covering up our own window only covered up (say) a 4 cm square in the centre of the client area. When this area is uncovered (by the user moving/closing the offending window) what has to happen?

Well, if we can solve the problem for the whole client area we can certainly solve it for part of the client area – just redraw the whole damn thing. But what if it takes several minutes to recreate some fancy graphics that were being proudly displayed? Maybe it would be better to arrange to repaint the absolute minimum necessary to correct the screen image, so as to save processor and user time.

WM_PAINTing

When any of the nasties discussed above occurs, Windows will send to ScrollWindowProc( ) a WM_PAINT message which is dealt with in lines 437 to 446. Two things happen. The BeginPaint( )/EndPaint( ) pair of calls are required by Windows and among other things deal with the PaintStruct variable. The important thing from our point of view is that one of the fields of this structure is filled with the details of the rectangular area of the client window that has been uncovered and, therefore, needs repainting at this point in time. This area could have been uncovered because of scrolling, the removal of a (partially) covering window, the opening of the window or whatever. In fact it may well be the case that more than one uncovering has

TRAINING FOR THE FUTURE

THE MICROCOMPUTER UNIT

ANNOUNCES

OS/2, WINDOWS & COMMUNICATIONS COURSES

OS/2 — A Technical Overview — *1 Days*

This overview will consider the main concepts behind OS/2, the relationship between OS/2, Presentation Manager and Windows and the question of compatibility between MS-DOS, OS/2 and Windows. During the course you will have the opportunity to use OS/2 running on an AT.

OS/2 for Software Developers — *4 Days*

This course covers the major features of the OS/2 kernel including multitasking, memory management and inter-process communications. Example programs will be developed to highlight the main OS/2 features. Each delegate will be provided with a PC throughout the course to use for hands-on practice.

Windows Presentation Manager for Software Developers — *4 Days*

Aimed at software developers using either OS/2 Presentation Manager or Windows 2. The concepts and requirements for operating with the graphics display interface (GDI) and the multitasking environment will be covered in detail. Each delegate will be provided with a PC throughout the course to use for hands-on practice.

Data Communications and Networks — *3 Days*

This course presents a practical approach to data communications, covering both capabilities and limitations. State-of-the-art transmission media and standards are explored and the physical and logical layers explained.

Local Area Networks — *3 Days*

This course is aimed at providing participants with the basic knowledge for evaluating, designing and implementing LANs. The course will cover hardware, software and protocols as well as the various standards. Particular attention will be paid to networking personal computers with emphasis on the current state of the LAN market.

OS/2 Device Drivers
OS/2 LAN Manager Administration and Operation
OS/2 LAN Manager API Programming
OS/2 for the User

A number of new OS/2 courses will be announced later this year. To receive details as soon as they are available, ensure you are on our mailing list by phoning 01-353 9841. The Microcomputer Unit's close working relationship with Microsoft means that all the latest information will be covered on these courses.

TELEPHONE TODAY FOR A DETAILED SYLLABUS

01-353 9841 or 01-353 9595

All courses are held at our Central London premises. Other courses for which syllabuses are available include:

- Ada
- Pascal
- C
- Unix
- MS-DOS
- Real Time Software and Systems
- Using MS-DOS and PC Applications
- Digital Signal Processing
- dBASE III Plus
- Lotus 1-2-3
- 68000
- 8086
- Interfacing
- Troubleshooting 8 & 16 Bit Processors and the PC
- Digital & Analog Electronics

- Small class numbers ensure personal attention.
- A micro-computer per delegate is supplied for all hands-on courses.
- Established in 1979, The Microcomputer Unit has nine years experience of running successful hi-technology and management courses.
- Central London location. 45, Holborn Viaduct, London, EC1N 2PB. Convenient hotels nearby.

CIRCLE NO 12

```
    nScrollOffset = 0;
    SetScrollPos(hWindow, SB_VERT, 0, /*redraw =>*/ TRUE);
    InvalidateRect(hWndScroll, (LPRECT)NULL, /*erase =>*/ TRUE);
    UpdateWindow(hWndScroll);
    }
  switch (wParam)
    {
    case CR:
      ScrollWindow(hWindow, 0, -nLineDy, (LPRECT)NULL, (LPRECT)NULL);
      if (hMemScrollLine[nBottomLineIndex] != NULL)
        /* Already have a global block. */
        if (nCurrentColumn == 0)
          hMemScrollLine[nBottomLineIndex]);
        else
          GlobalReAlloc(hMemScrollLine[nBottomLineIndex],
                        nCurrentColumn, GMEM_MOVEABLE);
      else /* No current global block */
        if (nCurrentColumn > 0)
          hMemScrollLine[nBottomLineIndex] =
            GlobalAlloc(GMEM_MOVEABLE, nCurrentColumn);
      {
      short nColumnIndex;
      LPSTR lpMem = GlobalLock(hMemScrollLine[nBottomLineIndex]);

      for (nColumnIndex = 0;
           nColumnIndex < nCurrentColumn;
           nColumnIndex++)
        lpMem[nColumnIndex] = cinputline[nColumnIndex];
      GlobalUnlock(hMemScrollLine[nBottomLineIndex]);
      }
      nScrollLineSize[nBottomLineIndex] = nCurrentColumn;
      nCurrentColumn = 0;
      if (--nBottomLineIndex < 0) /* don't fall off bottom of buffer. */
        nBottomLineIndex =  SCROLLBUFFERSIZE - 1;
      break;

    default:
      /* Draw and store the new character, provided there */
      /* is space available in the cinputline buffer.     */
      if (nCurrentColumn < INPUTLINESIZE - 1)
        {

        cinputline[nCurrentColumn++] = wParam;
        }
    }
  break;

case WM_PAINT:
  {
  PAINTSTRUCT PaintStruct;

  BeginPaint(hWindow, (LPPAINTSTRUCT)&PaintStruct);
  PaintScrollWindow(PaintStruct.hdc, PaintStruct.rcPaint);
  ValidateRect(hWindow, (LPRECT)NULL);
  EndPaint(hWindow, (LPPAINTSTRUCT)&PaintStruct);
  }
  break;

case WM_SIZE:
  if (wParam == SIZENORMAL)
    {
    nScrollClientDx = LOWORD(lParam);
    nScrollClientDy = HIWORD(lParam);
    if (nLineDy)
      {
      nNumLinesInWindow = nScrollClientDy / nLineDy;
      SetScrollRange(hWindow, SB_VERT,
                     -(SCROLLBUFFERSIZE - nNumLinesInWindow), 0,
                     FALSE
                    );
      SetScrollPos(hWindow, SB_VERT, -nScrollOffset, FALSE);
      }
    }
  else
    return((long)DefWindowProc(hWindow, message, wParam, lParam));
  break;

case WM_VSCROLL:
  {
  short nLinesToScrollBy = 0;

  switch (wParam)
    {
    case SB_PAGEDOWN: /* Scroll paper half a page up. */
      nLinesToScrollBy = -nNumLinesInWindow / 2;
      if (nScrollOffset + nLinesToScrollBy < 0)
        nLinesToScrollBy = -nScrollOffset;
      break;

    case SB_LINEDOWN: /* Scroll paper a line up.      */
      nLinesToScrollBy = -1;
      if (nScrollOffset + nLinesToScrollBy < 0)
        nLinesToScrollBy = 0;
      break;

    case SB_PAGEUP: /* Scroll paper half a page down. */
      nLinesToScrollBy = nNumLinesInWindow / 2;
      if (nScrollOffset + nLinesToScrollBy >
            SCROLLBUFFERSIZE - nNumLinesInWindow)
        nLinesToScrollBy = -nScrollOffset +  SCROLLBUFFERSIZE -
                             nNumLinesInWindow;
      break;
```

Figure 1 continued- additions to WINAPP.C.

occurred (imagine that two pop-up windows are simultaneously removed) and Windows will combine the two areas into one by combining a second WM_PAINT message with an already outstanding WM_PAINT message. This is why all other messages overtake WM_PAINT messages in the application's message queue ... a call to UpdateWindow( ) can be used to force the an outstanding WM_PAINT to be actioned immediately if required.

The paint structure also contains the handle of the relevant display context which we need for the GDI calls we might make in repainting part or all of the Scroll Window's contents. In particular the routine TextOut( ) will need it. So, the hDC and the rcPaint rectangle are both passed in the call to PaintScrollWindow( ) which is responsible for the repainting work. Before we can look at this routine we have to decide on a method of storing the data that will keep it available whether on or off the screen, displayed completely or partially covered over.

Virtual Paper

Taking a simple view of things we see that our virtual sheet of paper would ideally be an infinite two dimensional array of characters – provided only characters are to be input and displayed (which is the case here). Windows may seem to have some infinite characteristics at times (requires infinite memory, infinite CPU speed, infinite patience to program, infinitely long learning curve!) but infinite memory we don't yet have so let's make some practical restrictions.

We'll limit the width of the paper to something reasonable like 80 or 120 characters (INPUTLINESIZE – defined in the .h file). The idea of being able to scroll the window (as in Windows Terminal) is so that we can review recent input both in terms of what we can see in the Scroll Window when it is on screen and in terms of a number of previously displayed lines which have already scrolled off the top of the window. How many lines? Whatever suits you (up to memory limits). Say 100, or 200 perhaps. SCROLLBUFFERSIZE, also defined in the .h file, states how many lines are held in memory. This includes storage for both the lines on screen and those off screen.

The Scroll Buffer

So, we end up with a two dimensional array of characters which consists of SCROLLBUFFERSIZE lines, each of a maximum length of INPUTLINESIZE characters. In fact a simple two dimensional array

Figure 1 continued- additions to WINAPP.C.

```
492
493         case SB_LINEUP: /* Scroll paper a line down.      */
494           nLinesToScrollBy = 1;
495           if (nScrollOffset + nLinesToScrollBy >
496                 SCROLLBUFFERSIZE - nNumLinesInWindow)
497             nLinesToScrollBy = 0;
498           break;
499
503         }
504
505       if (nLinesToScrollBy)
506         {
507         nScrollOffset += nLinesToScrollBy;
508         SetScrollPos(hWndScroll, SB_VERT, -nScrollOffset, /*redraw =>*/ TRUE);
509         ScrollWindow(hWindow, 0, nLinesToScrollBy * nLineDy,
510                     (LPRECT)NULL, (LPRECT)NULL
511                    );
512         }
513       }
514       break;
515
582  /*==========================================================*/
583    long FAR PASCAL OurWndProc(hWindow, message, wParam, lParam)
584  /*==========================================================*/
615
616    case WM_DESTROY:
617      /* Free and global memory still in use for lines in the Scroll */
618      /* Buffer. This must not be forgotten ...                      */
619      {
620      short nHandleIndex;
621
622      for (nHandleIndex = 0; nHandleIndex < SCROLLBUFFERSIZE; nHandleIndex++)
623        if (hMemScrollLine[nHandleIndex] != NULL)
624          GlobalFree(hMemScrollLine[nHandleIndex]);
625      }
626      PostQuitMessage(0);
627      break;
```

could end up being very wasteful of memory since not all lines input will be of the maximum permissible length. Instead we will have a single one dimensional array of characters to hold the characters being input on the current bottom line of the screen. As soon as the user hits the return key to move onto the next line we will copy the input line's characters into a dynamically allocated array of characters of (in theory) exactly the correct length to hold the characters. A second linear array, of SCROLLBUFFERSIZE elements, will be used to point to the storage holding the lines of characters. In other words we implement a sparse array that copes quite well with left justified text displayed in the Scroll Window.

Global Memory Allocation

The dynamically allocated memory chunks required will be provided from Windows' controlled global heap. To keep Window's options open we will allow these chunks to be moveable so that Windows can reorganise the global heap whenever it sees fit. This has as a consequence the fact that we cannot just store long pointers to the blocks since Windows may move the block and the pointer would then be out-of-date and downright dangerous. Instead we will store an unchanging handle to the block. All we are really doing is a form of indirect addressing where we always access via a secure value (ie via the handle which in effect is an index into a Windows data structure in which Windows keeps track of the current location and size of the memory block). That's the theory, what's the practice?

Windows provides several routines to deal with memory allocation. We will use the global heap routines. This makes sense, particularly where a lot of memory might have to be allocated since the local heap must satisfy all its allocation requests from just 64K. If we allow for a thousand stored lines of text, each 100 characters long – somewhat excessive I admit – we will run out of local memory. We will request global memory in exactly two sets of circumstances – let's deal with them in turn.

When we first start up an instance of WINAPP.EXE and start entering text we have no global memory in use and the array of handles will contain nothing but NULL handles. This array, hMemScrollLine[], is declared in line 50 and is filled with zeros – which correspond to the Window's NULL handle value. As the user types in the characters forming the lines of text they are displayed on the bottom line of the screen and are simultaneously stored in the cinputline[] character array (line 432). The 'if' statement (line 422) ensures that we do not overflow the input buffer.

The window's client area is scrolled as before (line 373). It is now time to transfer the contents of the cinputline[] array into a global memory buffer – and at the moment we have no such buffer. This takes us to line 384 where we eventually request a block using GlobalAlloc() in line 386. GMEM_MOVEABLE allows Windows to move the block should it need to do so. nCurrentColumn holds the number of bytes we require (since it is an index into the first free column on the screen and/or byte in the cinputline[] array). As is often the case with such memory allocation mechanisms Windows does not necessarily allocate exactly the number of bytes we ask for; instead we are guaranteed to receive at least as many bytes as we have asked for – there may well be more. I will come back to this point in a moment.

GlobalAlloc() returns a handle to the block of memory rather than a pointer; we need a pointer in order to copy the contents of cinputline[] so we have to lock the memory in place (so that any pointer to it will remain valid whilst we are using it). GlobalLock(), in line 389, both locks the memory and returns a pointer that will remain valid until GlobalUnlock() in line 396 is called on the handle. The for loop (lines 392 to 395) is responsible for the copy. In line 398 we carefully store the length of the line just stored into the array nScrollLineSize[]. This is done because, although there is a function GlobalSize() that will return the size of the block allocated by Windows, we must not assume that our line is exactly the same length as the block allocated.

I have tried without success to find exact details of the allocation mechanism in the Windows Manuals. Experiments have revealed that the smallest block that will be allocated is 16 bytes (1 paragraph). After that the block size will increment by 32 bytes giving allocations of 16, 48, 80, 112, 144 and so on. I assume that one must not rely on these sizes however.

If the program had been running for some time it might well be the case that we had already allocated a block of memory for the line which is now furthest 'in the past' in the cyclic buffer. This line's storage must now be used for the current input line. In other words we have reached the far side of our piece of virtual paper.

In this case we do not want to allocate a block but rather to reallocate one. Effectively, whether we end up with the same block that we had before or not, we will have a new block suitable for handling the current input line whether this is larger or smaller than the previous line that was stored. The reallocation is carried out in line 380. If the current line is empty (the user has just typed a carriage return and nothing else) and we previously had a non-empty line (so storage space has been allocated in the past but is no longer required) then GlobalFree() is called in line 378 to put the memory back in the global pool.

A very important point that must not be forgotten is implemented in the window procedure dealing with the WM_DESTROY message (lines 616 to 627). Quite simply we must return any global memory still allocated, before Posting ourselves a Quit Message (line 626). If we don't do this nobody else will do it for us, and your memory will appear to shrink.

Mapping

We now have everything nicely set up to enable us to repaint the Scroll Window's client area. Provided we know how the stored text lines map onto the screen it will be a simple matter of redrawing the relevant characters at the appropriate places in the window.

The mapping is quite straightforward when the window is not scrolled. The nBottomLineIndex variable is used to record the index into hMemScrollLine[] selecting the handle to the Global Memory block that will hold the line to be displayed at the bottom of the scroll window. Since we can calculate how many lines high the client area is (eg when the window is resized for any reason – line 455) we just output the correct number of lines above and including the current line (which is in the cinputline[] array rather than in a block of Global Memory).

Note that 'above' will need interpreting when the line 'above' the topmost line in the circular buffer is referred to since the buffer wraps back to index zero. This calculation is performed during the running of the paint procedure (line 257) when nScrollBufferIndex – which takes note of the wrapping effect associated with the circular buffer – is initialized in lines 273 to 280.

If the window is scrolled the variable nScrollOffset records by how many lines. This is used in the for loop (line 267) to offset the indexing into the hMemScrollLine[] array. With the scrolling thus accounted for in the characters held in the relevant (locked) global memory block are printed (lines 289 to 295) and the client area is built up line by line. The current line is a special case and is discussed and dealt with in the listing in lines 314 to 327.

In line 264 the text colour is selected. Normally the colour chosen would be yellow since that is the colour we have been using for displayed text in this window. To help with debugging the code associated with dealing with the WM_PAINT messages I have changed the green byte in the RGB macro call to zero which causes the text to be repainted in red. In this way I can easily tell whether text has been displayed by repainting (red) or upon its initial input (yellow).

rcPaint

Sometimes it will pay to only do the minimum amount of repainting required to complete the window's display, ie paint only the area designated by rcPaint. Having said that, both of my 80386 and 80186 machines output a whole screen sized window's worth of text in the blink of an eye so it is not really worth optimizing these routines. The output of time consuming graphics is another matter however so I have added the code in lines 302 to 310 to

Sometimes it will pay to only do the minimum amount of repainting required to complete the window's display.

slow down the whole process of window repainting. Repainting a large window will now take several seconds as will the repainting of even a couple of square centimetres uncovered in the lower part of such a window because the main for loop (line 267) repaints all the lines in the window whether they are in the area specified by rcPaint or not.

Note that the drawing of the graphics depends on the if statement in line 298. As a simple control I have used the Boolean variable bGraphicsOnScreen (which records whether the graphics in the main window are being displayed or not and is effectively set or unset from an existing main window menu item) to specify whether the tests for being in rcPaint's vertical boundaries are to be done (lines 299 and 300). In this way it is simple to observe the considerable speed differences with and without the rcPaint check.

In general, if this optimization in terms of 'do the minimum' is necessary it would be better to modify the calculation of the main for loop's end points rather than loop through all the lines deciding that many of them need not be displayed.

Scroll Bar messages

As mentioned earlier the extent of scrolling must be constrained to the size of paper that we have data storage space allocated to implement. If you compare this month's code for dealing with the WM_VSCROLL message and SB_ sub-messages (from line 467) with last month's you will see that an if statement has been added which checks for attempted out-of-bounds scrolling and corrects nLinesToScrollBy to something reasonable.

At various points (when the window has scrolled) the position of the elevator on the scroll bar needs to be set. This is done by using SetScrollPos() in lines 366, 460 and 598. Also, whenever the height of the window changes the amount the elevator will move during scrolling must change. For example, if the window is 20 lines tall and the ScrollBuffer[] holds 100 lines then scrolling at a page at a time will require four scrolls to view the whole of the paper so the elevator will cover the whole distance along the scroll bar in four steps. If the window is only 10 lines tall the paper will be scrolled over – and the elevator will move along the full length of the bar – in nine steps instead. This problem is neatly dealt with by setting the range of values that the scroll bar can cope with to correspond to the number of lines outside the window (lines 456 to 459).

Exercises for the reader

You may have noticed that the subject of sideways scrolling of the window has been omitted. How quickly can you add the code in? It should be almost as easy as turning your monitor on its side!

Next Month

Next month, I'll explain how to define your own controls. After that, we will draw this series to an end with an article about the differences in programming for Windows versions 1 and 2. Having reached this point, we will then concentrate for a few months on the OS/2 Presentation Manager, including articles on how to write PM code from scratch and how to convert existing Windows applications.

EXE

NEOW - The Windows Specialists

Windows Developer?

If you're not already considering MS Windows how much longer can you ignore the most significant advance in personal computing. NEOW are dedicated experts in Windows software and can offer new or experienced developers a unique service, helping you to write better software.

Actor

Actor and a personal computer are all you need to write powerful, interactive MS Windows programs in a shorter time than ever before possible. But Actor is more than a programming language. It's a sophisticated development environment suited to all types of programming, simplifying even the most complex task. Actor's origins lie in object-orientated languages, the natural choice for windowing environments, and utilises the latest incremental compilation and interactive debugging techniques to provide unrivalled productivity. Actor provides 1000's of lines of pre-written code, and has no runtime charge.

Windows 2 Dev Kit

The Microsoft Windows development kit provides the libraries for you to write Windows applications in 'C', Pascal or assembler. As a developer you benefit from device independence, graphics, menus, multitasking and data exchange. It includes editors for cursor, bitmap , icon and dialog box creation. You will also be developing far more powerful applications with features not possible in lesser environments. And Windows provides the natural route to OS/2 Presentation Manager applications, ensuring your investment and giving you a head start on the competition.

Quartz

Omnis Quartz is a professional database and applications development system utilising many of Windows features. Quartz supports a relational and hierarchical file structure offering up to 2560 MBytes of storage. Input forms, menus, buttons and overlapping windows can all be quickly created to produce prototypes for evaluation or approval. Its unique language provides a rich variety of functions and control structures and data can be imported from dBase, Lotus and most other packages. Multiuser and runtime systems available.

SQLBase

SQLBase is a distributed relational database management system for PC networks. It can be used as a DBMS engine for stand-alone applications development, as a database server for PC workstations on a network, or as a gateway to mainframe DBMS's. SQLBase features the complete SQL language syntax. It offers advanced concurrency, automatic consistency, and dynamic recovery - capabilities never before available on PC's. Single and Multiuser versions available. Fully compatible with Windows.

Other products include editors, code generators, debuggers and books. All for Windows. All from NEOW.

NEOW **The Windows Specialists** ™

NEOW. 470 London Road, Slough, Berkshire. SL3 8QY. (0753) 45115

Name__________ Position__________

Company__________ Address__________

__________ Postcode__________

Phone__________ Windows User (Yes or No)______ PC Type__________

Please send me more information on: Actor ☐ Quartz ☐ Dev Kit ☐ SQLBase ☐ and the Neow Windows Catalogue ☐

All trademarks acknowledged

CIRCLE NO 13

Liveware Matters

Part one of our series on the software development life cycle presented an overview of the entire process. In part 2, Philip Sully examines the subject of people themselves, and how best to organise them in order to get a job done on time, within specification and within budget.

The most important part of any software project is the team of people who actually produce it. Without people, no code will ever be generated. This article looks at how to manage people. To be more precise, it examines the best ways to organize them into teams in order to make the most efficient use of their programming skills. Of course, a project does not simply require programmers. It also needs quality controllers, managers, time schedulers and so on. All these areas will be discussed, as will the topic of how best to motivate a team. Hopefully, after reading this, you will understand the questions that should be asked before starting to assemble a group of people to complete a programming task.

The Team as the Vehicle

Software projects typically comprise a large number of tasks, each of which demands combinations of labour, skill and time to complete. The first problem, as we all know, is that not all combinations of these three factors will produce an effective team. It is the 'nine women can't do in one month what one woman can do in nine months' syndrome. In other words, the tasks are not perfectly partitionable. In software engineering terms, the reason for this, as Fred Brooks asserted in his book The Mythical Man Month, is that the increase in the communication burden within the project is not linear. This means that increases in the number of labour units employed does not necessarily result in a pro-rata increase in effectiveness.

The second problem is that not everyone has the same skill levels. Further, skill levels in individuals will change. They learn (hopefully). Suppose, in order to counter the skill level variation problem, we increase the number of labour units involved. How far should we go?

The size of a team

The most effective team size is very difficult to predict, but it is a function of at least two factors: individual judgmental discretion and activity cycle time. Where the discretion given to an individual is low and the time taken to execute an activity is short, large teams can be effectively controlled. Take for example the activity of laying pipes in a trench. Each individual has next to no discretion and the time to put a length of pipe down is, say, two to five minutes. In such situations a supervisor would be able to effectively manage a team of 20 or more people. But, as a contrast, a lead draughtsman can usually control only 4 to 5 people because each draughtsman has considerable discretion and each activity takes a long time and requires much communication. The size of a team is often determined by arithmetic means. But this approach can often be seen to ignore all the collective experience of mankind.

By definition, a team must consist of two or more participants. But what about the upper limit? Throughout his history, man has found a need to band together in order to achieve goals that either could not be reached on his own, or were more efficiently reached in concert with others. These goals have ranged from fighting for survival, through working to survive, to playing to enhance the quality of survival. In most cases, an upper limit to the number of people used to constitute the team can be traced.

The Romans used fighting units of ten men to form a unit, lead by a Decurion, and ten units were commanded by a Centurion. Modern armies have also formed themselves into similar structures. Few team sports exceed 11 players. Rugby is an exception, but even in that game, it is found efficient to divide the team into forwards and backs, each sub-unit having its own leader.

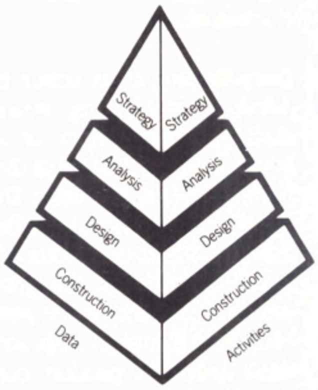
Project work versus operation work

The difference between project work and operation work is mainly centred upon the degree of routineness and innovation. In project work there is usually quite a degree of innovation and there is a definite starting point and a definite end point. For operational work, there is continuity in the form of the work being of an on-going nature.

This nature of project work does beg questions about how we should man the project. Should we have a central pool from which we draw out people with certain skills? Or should we, instead, construct a team specifically for the project in question?. Each approach has its advantages. If the situation is routine then the pool approach tends to have associated merits (eg economies of scale due to being able to make resources more fluid to match demand). In a project where there is innovation and possible risks, though, a team is often a more flexible unit related to the software development tasks.

Philip Sully is with Yourdon International, based in London. He can be contacted through the editorial office.

MANCHESTER & LONDON
SEMINARS



Isn't it about time you found out about Integrated Computer Aided Systems Engineering (I-CASE)?

All it takes is half a day of your time. Arthur Young (IES) Ltd are holding more ½ day seminars on Information Engineering and its Integrated Computer Aided Systems Engineering product set: IEW. You can see first hand the effect they could have on your systems planning, analysis and design. And how systems backlog really can become a problem of the past.

You're welcome to attend one of these free seminars, either on June 10th or 28th at our London offices, or on July 1st at our Manchester offices (Commercial Union House, Albert Square, M2 6LP.) To book a place, or simply for further information, contact Trudie Goodall at Arthur Young (IES) Ltd., Rolls House, 7 Rolls Buildings, Fetter Lane, London EC4A 1NH. Telephone: 01-831 2191.

Arthur Young (IES) Ltd

A MEMBER OF ARTHUR YOUNG INTERNATIONAL

CIRCLE NO 14

Figure 1 – The different personalities required to produce an effective team.

ROLE	DESCRIPTION
Chairperson	Clarifier of objectives and roles
Shaper	A foreman who calls attention to deadlines etc
Initiator	The idea proposer or generator
Evaluator	Evaluates / analyses Pros and cons of Ideas.
Company worker	Exhibits common sense and stability
Resource Investigator	The resources person for the Project set up
Team worker	Team member that is group oriented seeks compromises
Completer	This member has a marked compulsion for detail

Selection criteria for the members

The effective team rarely comes about through luck. The more effective teams I have seen in the past have had a balanced mix of personality type conducive to the team working as a cohesive unit. What are these useful personality types? A researcher named Belbin at the Administrative Staff college at Henley presented an idea that a team should be constructed from a group of people that made the team a cohesive and effective unit. The suggestion is that a team ought have a mix of personalities as illustrated in Figure 1. This provides a list of all the attributes that are needed in a team for its effective working. As long as the team has someone with each attribute, so much the better. Of course, it's likely that team members may individually have more than one role in the team.

This is very much like the situation that is mentioned in the book by Fred Brooks (The Mythical Man Month), in which the author likened the team types to a surgeon's team. Here, the roles were Chief programmer, Administrator, Librarian and Language Lawyer.

The key question is whether to recruit into the team these complimentary personality types? Personally, I'd say that you certainly should. Exactly how to go about recruiting people with these qualities is the subject of many books, and I don't intend to delve into the subject here. Suffice it to say that, in many people's opinion, a well-structured formal interview with a potential employee will tell you more about that person than a dozen aptitude tests and psychological exercises can. One method used by a certain software company may be of interest, however. The candidate is asked to bring along a printout of some code he has written recently. He is then asked to hold up the printout, and the interviewer studies it from the other side of the room. Although this method won't highlight good code, says the company concerned, the patterns in the printout almost always show up any badly written code.

At this point I think it is worth discussing the relationships between the end product and the nature and organisation of the work involved to produce it. From many workers in this field the relationship of effort and size of software product has distinct indications that it has diseconomies of scale.

Two theorems are shown in Figure 2. These indicate that effort and duration are, of course, related, but in a much more complex way than you might have imagined. An effort of 1 man month will translate to a duration of about 2.5 months.

What size team?

Earlier in this article we have alluded to an effective upper limit to the team size which was based upon the idea of sports teams. There are also practical limitations, from a project leader's point of view. He or she must be able to maintain a good working relationship with all the team members, and I believe that this criterion puts the limit at around 11. Some people's ideal is as low as 7.

How many teams?

If the project under development is of such a size that the work required is too much to be assigned to one group, the obvious solution is to put together a team of teams. This solution is no different to the similar situation with other work group situations such as the armed forces or in large construction projects. It's been found that the more you break down a project into small tasks that can be handled by small teams of around 3 people, the easier it will be for the person scheduling the work to ensure that the job is being done, and that it is being completed within the total time allotted for it.

Motivation

How predictable are people? Can anyone calculate the level of motivation necessary to extract a given level of extra effort? Is there a motivation versus effort transfer function? The Hawthorne experiments did show that all attempts at improving the work site had effects. But the effects were not immediately predictable. Social behaviour cannot be predicted with the same degree of certainty as factors in the physical sciences.

Barry Boehm, a key writer in the field of software engineering economics, does outline the importance of organisational issues in his assessment of the various sensitivities to productivities. The basic Constructive Cost Model of Boehm's does show the effect of various effort adjusters (cost drivers) and the group with the greatest sensitivity is the personnel capability (see Figure 3). This sensitivity overrides items like product complexity and timing constraints. The diagram shows just how much the capabilities of a team affects software development, especially when compared to such variables as the language experience of the team members or the overall complexity of the project.

According to Fred Herzberg, motivation may be explained in terms of two major components – the hygiene factors and the motivator factors. The hygiene factors ensured that basic working conditions were at least satisfactory (eg lighting, ventilation and comfort) whilst the real motivating factors were related with self actualisation (personal growth). Within the systems development industry a survey was made

Nanus and Farr:	*Effort = constant* Number of Instructions**1.5*
Boehm:	*Effort = constanta*Number of instructions**1.1*
	Duration=constantb Effort** 0.33*

Figure 2 – Two ways of relating required effort to the size of a programming task.

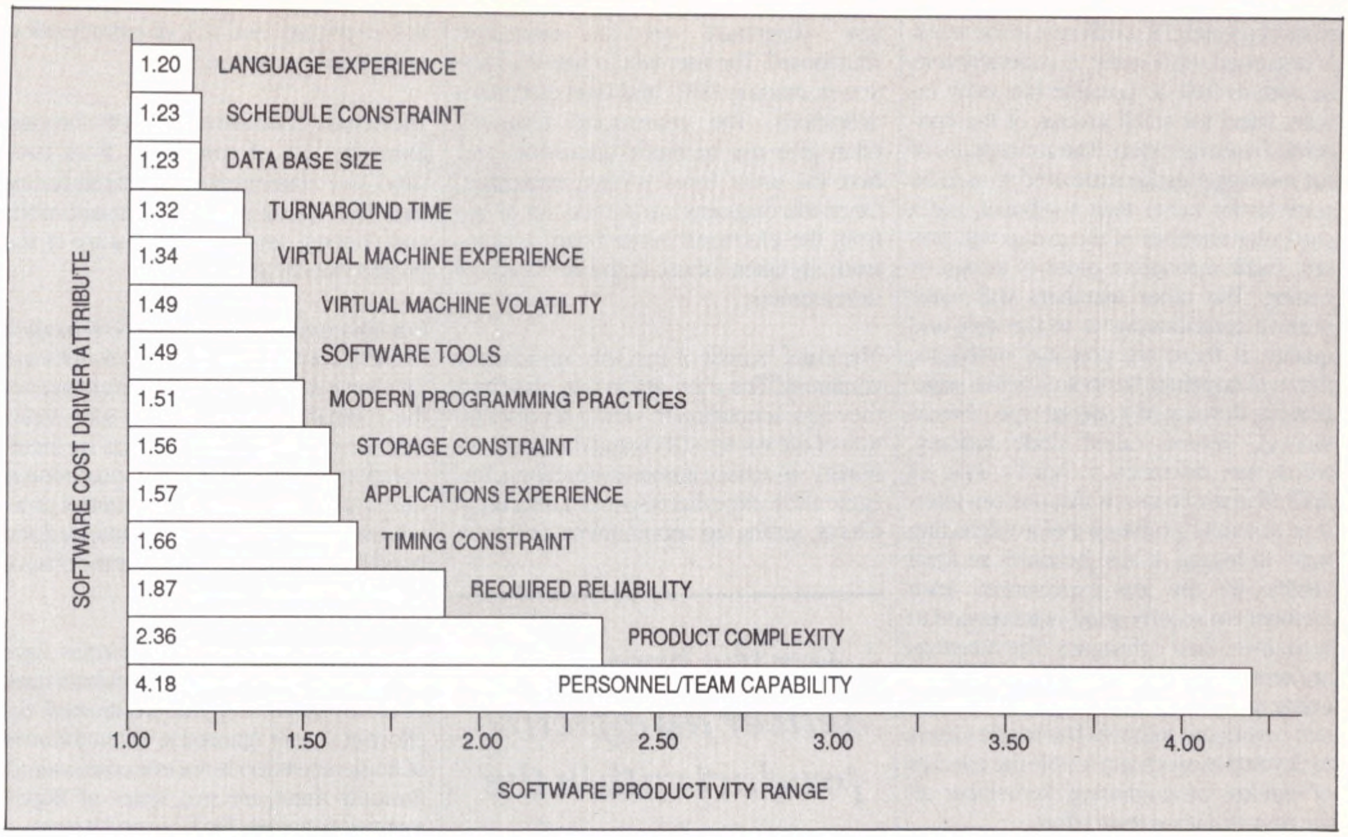


Figure 3 – One view of the factors that affect the final cost of software development.

(Fitz-Enz 1978) and the results did show the key preference towards personal growth. These were, in order, Achievement, Possibility for Growth, Work itself, Recognition and Advancement.

What are the consequences of this to the industry? It underlines that people have expectations towards the development activities that are positive contributions towards being productive.To quote Townsend, 'You can't motivate people if the door is locked on the inside'.

Remuneration

Before deciding how, and how much, to pay people, it is worth firstly seeing what does motivate people and to ask ourselves what we should be optimising? I suggest that the items we should be looking at are these. From the product point of view, we need high quality, low cost to build and good maintainability. For the project, it must be on time, within its specification and within budget.

A remuneration scheme ought ideally to be tied to each of these characteristics. However, we all know that compromises are made between cost, quality and functionality. Therefore, any formula we construct ought recognize this compromise. The following is a suggested scheme associated with the above categories:

To achieve a high quality product, pay people according to MTBB (mean time between bugs). To achieve a low building cost, fix deadlines for each stage of the work and pay according to whether (and how soon) the deadlines are reached. If you are working on a fixed-price contract, this is even more important. To achieve maintainability of the product, set out payment schedules related to the time taken to apply a change or an enhancement.

To keep the project itself on time, pay according to deadlines reached and, to keep within budget, make it clear to the team from the outset that satisfactory completion within budget will result in a bonus.

The walk through

The nature of the software project demands that there is much communication that is required through the project's duration. Also, some communication channels remain after the project's completion. The communications channels involved are team- member – team member, team – management and team – users. Note that all these are two-way links.

The nature of each type of communication channel is different. Communications can be split into product-related and project-related messages. Within the team, product-related messages include requests for technical help, discussion of structural and design details, testing strategy and the testing itself. Project-related communications include progress reports, requests for resources and general intra-team liaison.

Within the team-user communication channels, product-oriented messages include requirements specifications, systems analysis, testing and the request for changes. Project-related topics include the timetable, progress reports and, again, requests for changes and alterations.

Within the team-management communications, product-based communications include quality assurance and testing, while project-based ones cover timetables, progress, testing and anticipating potential problems:

For the project to deliver an effective product within the pre-planned criteria the communication must be frequent and effective. The two problem areas are intra-team and team-user communications.

The remedies that I suggest for improving intra-team communications is the walk-through. This is an active form of meeting for technical issues such as the structural, detailed specification, testing strategies and coding. How is it different from other types of meeting? It is different in the way it is organised. Principally, it is designed to be with as little as possible hierarchy involved and for small groups of the concerned team members. The atmosphere of the meeting must be structured so as to be constructive rather than a tribunal, and a particular member of the group will present (walk through) a piece of design or coding. The other members will make positive contributions as to the style and quality. If there are potential problems, they will hopefully be spotted at this stage. IBM has developed a special type of walk-through session called Code Reading, which was discussed in April's issue of .EXE. The whole idea is that one can intercept potential problems before detecting them in testing. It also becomes an ideal vehicle for the less experienced team members to acquire good practices and to learn from their colleagues. The meetings are kept to the relevant team members and a time limit of 2 hours is suggested to maintain participant interest. The whole idea is to develop as much as possible the concept of ego-less programming. Remember all the time that it is a team effort.

Brainstorming

The above style of meeting, arranged with minimum hierarchy, is a good format for problem solving and idea generation. But there are phases in a project's lifespan where active contributions for the generation of ideas are required. The session I am referring to is brainstorming, where the participants literally ,idea dump' as a response to stimuli provided by a moderator which may be cue from the problem directly or from some random source. This may be an ideal way to generate options in the design stages of a project, and such sessions should take place at all stages of the project.

Joint Development

Occasionally, it can be beneficial to allow end users of the software to attend walk-through sessions. One insurance company I know of actually uses an active participation method that I think is worth outlining. The method is known as JAD, or Joint Application Development. The JAD approach is optimised for the requirement capture phase of the project.

The procedure involves inviting users to attend these specific sessions where the moderator is usually a systems analyst with knowledge of structured methods and he/she runs a session of stepping through a particular subject area with the users and constructing a set of structured DFDs (data flow diagrams) on an electronic whiteboard. The user would have had tuition in reading DFDs and Data dictionary definitions. The constructed diagrams often give rise to much discussion, and here the users' input is often invaluable. Once the diagrams are agreed, an image from the electronic white board is automatically taken for use in the next stage of development.

The chief benefit of this JAD approach is continuity. The users are in one place and they have a notation to view a representation of the system (DFDs and the Data Dictionary) in a more immediate medium. Imagine all the time that is spent chasing user/clients, setting up appointments and then having to agree all the requirements. With JAD, users can interactively participate, discuss and document.

For the project to deliver an effective product within the pre-planned criteria the communication must be frequent and effective

Progress reporting

Who with? How often? This group of communications is related to team-management communication. The format of a progress report ought to be consistent and regularised but it does rely on some decision concerning the reporting interval. The project milestone is essentially too large a unit to be reported on. A suitable size of work unit to be included in a progress report would be one that takes between half a day and 5 days to complete.

The Penalties of Failure

It is interesting from an educational viewpoint to ascertain reasons for Project/Product failure. The most common reasons are communication failure (with the user or within the team), lack of management support and the consequential lack of priority, and lack of consideration to the implementation.

The penalty of failure can be in its cheapest form the lack of confidence from your client and consequential lack of return custom. The highest penalty is immeasurable, and depends totally on the nature of the project that failed.

Communication failures can be remedied by the incorporation of walk-throughs and JAD, but it does rely on the management of the company constructing an infra structure for running better projects by incorporation of proper project procedures, quality assurance and change control procedures. If a project is well documented and it still fails then at least you have the chance to learn from the mistakes.

Many technically excellent systems have come unstuck at the Implementation stage because either they underestimated the effort or simply ignored it. Often the style of implementation is not even considered. Basically there are two styles of implementation, namely big bang and stream.

The big bang approach is where the whole implementation is accomplished as one event happening, say, over a weekend. As you can imagine it is one that is fraught with a lot of associated risks. With preparation and heavy helpings of user involvement, though, it can succeed.

The stream approach is where the implementation is phased into a series of events where the implementation load is spread over a long time period. It is much the safer and easier route to take.

More information

IBM runs a course entitled Managing the Application Development Process, which goes into more detail about topics not covered here. This course, and the methods it teaches, was reviewed in April's issue of .EXE.

Bibliography

Software Engineering Economics by B. Boehm. Publisher – Prentice Hall. ISBN 0-13-822122-7. People & Project Management by R. Thomsett. Publisher – Yourdon Press. ISBN 0-917072-20-0. Teams, by P. Semprovivo, published by Yourdon Press. ISBN 0-917072-20-0. The Mythical Man Month by F. Brooks, published by Addison Wesley, ISBN 0-201-00650-2.

.EXE

OS/2 User Group

A forum for software developers and technical personnel

Next Conference

4 JULY – CENTRAL LONDON

MICROSOFT – Development Update & SQL

IBM – Development Update

NOVELL – Integrating OS/2 with Netware

3COM – Development Update, 3 + Open

ASHTON TATE – dBASE IV & SQL Server

LOTUS – OS/2 Strategy – A Technical Briefing

OPEN FORUM – Panel of the Day's Speakers

– BOOK EARLY – ENROLMENT LIMITED –

SEMINAR BOOKING FORM

☐ I am unable to attend – please keep me informed of future events

☐ Please book .... place(s) for the seminar on ....
at £60.00 per delegate + VAT @ 15% = £69.00

Name(s) ....

Address ....

....

Prepayment is essential.
Please make cheques payable to "OS/2 User Group"
Cheque enclosed for: £

☐ Access ☐ Visa ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiry date: .... Signature: ....

For Credit Card Bookings, please state the card holder's address if different from above.

....

All bookings must be received by 30 June – subject to availability

Please return to:
Karen Burns, OS/2 User Group, c/o QA Training Ltd,
Cecily Hill Castle, Cirencester, Glos. GL7 2EF Tel: (0285) 5888

CIRCLE NO 15

Why use a 4GL?

Jim Fisher offers his own opinions on why programmers should always use specialised 4GLs to write database applications, rather than coding from scratch in a favourite high level language.

Old habits die hard and this is nowhere more true than in computer programming. The fact that BASIC and COBOL, to name but two common programming languages, are designated as 'high-level' has more to do with the fact that the computer industry has, at least, moved away from programming in machine code or assembler than it does to represent a degree of complexity commensurate with the rapid explosion in use and the sophistication with which computers are applied to today's commercial problems.

That is not to denigrate the advance such languages represented in their day. It is merely a recognition that, with the inexorable march of technology, today's high-tech must inevitably, and by comparison, be tomorrow's low-tech. It is not a question of one man's meat being another man's poison. It is a mark of progress.

Today, the term 'Fourth Generation' is liberally applied to many new products (and in the context that machine code was the first generation; Assembler the second; with COBOL, BASIC and their counterparts the third, the term is not invalid). Yet there is insufficient evidence that these have been greeted with the popular enthusiasm that accompanied the announcement of their forebears.

Since the use of fourth-generation products has widespread advantages in the development of users' applications, this is hard to understand – but the observation is equally true, whether one looks at mainframe, mini or micro-based systems.

Perhaps the answer lies in the fact that third-generation, 'high-level', languages are regarded by computer programming professionals as sufficiently sophisticated for their own needs; whilst saving enough of the mystique and complexity of the task to ensure their continuing employment.

Job security

Another fact is, however, that there are not enough programmers to write all the applications industry and commerce desires now. Programmers are under no threat. It has been estimated that applications demand is growing at 45 percent per annum, and that there are more applications waiting to be developed than have ever been programmed in the history of computing and that, before long, the entire workforce will need to become programmers in order to satisfy this apparently insatiable demand.

Therefore, to overlook or ignore the use of modern applications developers is to wilfully neglect one's duty to the employer that pays each month's salary and the needs of the company to get or stay ahead of the competition. Along with this objective goes everybody else's salary too.

An example of what can be achieved using modern products is seen in the microcomputer end of the industry, although equivalent examples can be found in mini and mainframe environments. Traditionally, programming comprised preliminary and detailed design and flow-charting (collectively called systems analysis), the actual programming sessions, testing and debugging, then, finally, implementation.

With modern applications developers – such as dBase III, RBase V, Dataflex and TAS – the systems analysis stage can be eliminated altogether, since it is very much inter-woven with the programming phase. Using an editor, the programmer is able to embark upon an immediate interactive programming session, with the user at his side if need be. Since such advanced products inherently contain facilities such as a central data dictionary, a screen painter and a program generator, the programmer can feel comfortable developing applications with the user as he or she unfolds his or her requirements.

These facilities automate the simple, boring yet time-consuming parts of the applications development cycle. Those parts with which the user cannot identify and which, traditionally, have to be repeated time and time again using old methods. They do not detract from the programmer's skill in developing an application to meet a user's requirement – with which the latter can identify and for whom the very 'unfolding' of a system can be a wonder to behold!

Jim Fisher is with Walker Fisher PR, consultants to Megatech which distribute the TAS database/language/compiler. He can be contacted on 01-686 5602. Megatech is on 01-874 6511. The Magazine would be interested to receive comments on the points made here. See the letters page for details on how to contact us.

*E*asy to use, easy to remember, SAMNA has been written for people who need the widest range of word processing and desktop publishing facilities to produce professional documents.

Unlike other software products, SAMNA PLUS IV has the same user interface and functionality whether it runs under DOS, Networks, UNIX, CTIX, AIX or XENIX.

SAMNA works across the board and will provide consistency throughout your organization even allowing text files to be transferred between your PCs and UNIX systems!

SAMNA PLUS IV works

under PC DOS, MS DOS 3.0 or higher
- on IBM PS/2 models 30, 50, 60, 80 and on all true compatibles
- on IBM PC, XT, AT and all true compatibles
- on Networks of Novell, IBM PC NET, IBM Token Passing Ring, MS NET, 3COM, BICC Isolan, Ungermann-Bass and Banyan

Strategic players move with

under UNIX
- on AT&T models 3B2, 3B5 and 3B15
- on NCR Tower models 32/400, 32/600 and 32/800

under AIX
- on IBM 6150

under CTIX
- on Convergent Technologies S Series

and

on 80386 processors supporting UNIX SYSTEM V Version III or SCO XENIX

SAMNA

WORD PROCESSING PLUS

SOUTHBANK HOUSE, BLACK PRINCE ROAD, LONDON SE1 7SJ

01-587 1121

CIRCLE NO 16

Defining screen layouts using third-generation products meant specifying the exact screen contents at every cursor position; yet screen painters within modern applications developers allow programmers to work with almost a blank page and WYSIWYG commands to obtain colour, graphics, boxes and text exactly as the user wishes them to be. Similarly, report-generators enable screen and printed output to be defined quickly and simply; not slowly and pedantically.

Applications developed for one company can be quickly and easily tailored to fit the foibles of another set of users, using mere editors within the applications development product or word processing packages.

Third-generation languages are 'unstructured' and, as such, require programmers to have an intimate knowledge of the language's syntax; whereas fourth-generation developers release the programmer's creative abilities and free him or her from the constraints and tedium of even making such considerations. Most developers' editors automatically take care of syntax.

Right first time

Most importantly, modern applications development products provide a way through the 'now you've done it, you know how not to do it next time' syndrome.

Traditional methods of programming are too time-consuming, hence costly, to put right the shortcomings and defects of the developed application. Using new developers, however, means that it is simple, straightforward and relatively cheap to make the changes the user did not originally foresee or convey to the programmer. The end result is a more professional-looking system to which users can relate.

The restructuring of a database to add a new field is an inordinately complex activity with BASIC or COBOL, yet it can be performed simply with modern applications developers. Therefore, the use of such products is not only efficacious for today's new application; it is equally efficient for modifying that application to meet tomorrow's changed circumstances.

Time savings can be many orders of magnitude; yet the user can always get the application he or she envisaged – so there is kudos to be gained by the DP department within the organisation too.

According to Theo Van Dort, managing director of Megatech, which distributes TAS in the UK, "Modern applications development products are to programming what Word Processing was to typewriting. Fourth-generation languages have the power to combine in a single statement what was previously several commands. Old style programming required that the design of an application was translated to, and understood by, a variety of people – from systems analyst to programmer, to punch card operators – so, we had to get it right first time because of the cost implications. Until this year, die-hard programmers could still claim 3GLs had the advantage that they could still be used down to the level of bit manipulation. However, applications developers, like TAS, can call machine code subroutines.

"Because programmers had to do everything for themselves, they had complete control and, if anything went wrong, they knew the problem lay within their program. Because 4GLs contain the functions of program generators, screen painters, procedural languages and so on, programmers have been inclined to question the developer if a problem arises within a program but they should be assured that current products are at least as stable and mature as the 3GLs they currently use. The mere fact that products like Dataflex, dBase and TAS exist is a testament to the fact that some programmers, at least, recognise the need for them and that they are a better alternative to high level languages like BASIC."

To overlook the use of modern applications generators is to neglect one's duty to the employer that pays each month's salary

Quick browsing

Van Dort quotes one specific example as evidence of the benefits such 4GLs can bring. "Products, such as TAS," he says "have a facility known as a Database Browser which allows programmers to view a data file like a giant spreadsheet – where data fields are displayed across the top of a file, from left to right; with individual records listed against the data field from top to bottom. Screens can be considered as simply windows positioned anywhere on these spreadsheet-like matrices. Such a view is simply impossible with any other way of writing applications programs yet considerably simplifies the programmer's abilities to see in total and cross-check what he has prescribed.

"In the past, programmers had to write test data files with which to check their program files. It would be impossible to write a Browser in BASIC; much less use the facility."

That these latest applications developers force the programmer to follow a structure is itself a blessing rather than a criticism, since it does make debugging a simpler and similarly easily-structured routine. Because a programmer quickly learns where a sub-routine is within any application, it is a relatively simple matter to examine those other parts of the program which are isolated from these proven schedules.

Like the best of these latest applications developers, TAS uses a sophisticated binary tree (B-tree) file access mechanism. This means any file can have multiple keys (20 or more), held in sorted order, to considerably improve search and retrieval speeds. Experienced users will know that re-sorting amended records in BASIC can appear, literally, to take forever.

Lastly, the very best current-generation applications developers include a compiler, so that users programs can not only be written quickly, but also they can be compiled to the smallest possible executable code so they run quickly too. This also produces stand-alone executable files that prevent user access to the source code of the application.

What next?

All in all they do represent the best of the present moves forward. One wonders what their widespread adoption will lead to next. It will be fascinating to be at or near the forefront of future of developments. Especially to see whether the pace of acceptance quickens the pace of yet further programmer development tools and, especially, the use of artificial intelligence methods.

.EXE

WHO'S WHO IN SOFTWARE ENGINEERING?

Advanced Programming Techniques . Advanced Systems Architectures . Alsys . Analyst Workbench Products . Apollo Computers . Arthur Young . Axion British Telecom . British Aerospace . Computer Resources International . Cullinet . Delta Software International . Deductive Systems . Densitron . Department of Trade & Industry . ERROS . Excelerator Software Products . Ferranti . GEC Software . IPL . Imperial Software Technology . Information Engineering Products . Instrumatic . Interactive Development Environment . LBMS . Logica SDS . Marconi Defence Systems . McDonnell Douglas . Michael Jackson Systems . Micro Focus . Microtec Research . MF Systems . National Computing Centre . Oracle . PA Consulting Group . Pansophic Systems . Parsoft . Pioneer Computer Systems . Praxis Systems . Program Analysers . Program Validation . Real Time Products . Relational Technology . Rex Thompson . Scicon . Software Sciences . Strategic Systems Technology . Systematica . Systems Designers . Tektronix . Telecomputing . Thomson Computers . Vista Computer Systems . Warren Point International . Yard Software Systems . Yourdon .

THAT'S WHO

SOFTWARE TOOLS'88

exhibition and conference June 14 - 16 1988 Wembley Exhibition Centre London

And that's who can help you with your software development projects. But only if you come to Software Tools '88, the only exhibition dedicated specifically to software engineering.

For more information, kindly clip your business card to this ad and send it to the address below or phone Debbie Gales on 01-868 4466.

Blenheim Online Ltd, Pinner Green House, Pinner, Middlesex HA5 2AE, UK.
Telex: 923498 ONLINE G Fax: 018689933

E5.2

CIRCLE NO 24

COMPUTER AIDED SOFTWARE ENGINEERING = TOOLS & METHOD

Jackson gives you:-

the right mix of software tools plus well-based method to realize the productivity benefit of CASE.

- **PDF: an interactive diagram editor with code generation facilities**
- **JSP-COBOL: a cobol pre-processor with powerful implementation features**
- **SPEEDBUILDER: the analyst's workbench with text and graphics support**

These are tools to improve productivity by a factor of two or more when used with Jackson Technical Methods.

WE SUPPLY THE TOOLS, TRAINING AND CONSULTANCY TO HELP ACHIEVE QUALITY AND PRODUCTIVITY.

MICHAEL JACKSON UK
22 LITTLE PORTLAND STREET
LONDON W1N 5AF
TEL: 01-499 6655

MICHAEL Jackson UK

CIRCLE NO 17

Ray Duncan

This month, Ray looks at the available 80386-specific system software that allows applications to take full advantage of Intel's top-of-the-range CPU.

In the previous two installments of this column, I've tried to give you a glimpse of the reasons why some software developers – particularly those who must implement compilers or very large, complex applications – are so excited about the Intel 80386 microprocessor. The 80386 offers a true 32-bit architecture and address space, speed to burn at 20 MHz (with chips as fast as 35 MHz expected within two years), upward compatibility with current processors, and permanent relief from the claustrophobic 64K segments of the 8086, 186 and 286.

Well, reading about nifty 80386 features is all very fine, but what if you want to do something practical with them right now and not two years from now, when an 80386-specific, 32-bit version of OS/2 appears? Suppose your program brainchild is suffering from memory starvation, uses huge data structures, is already being loaded in five layers of nested overlays, and generally cries out for a bigger and better CPU. The 80386 seems ideal, and machines based on it are getting cheaper by the day, but how does a developer make sense of the current chaotic situation in 80386 system software and put the processor to work properly without burning any bridges?

80386-specific system software falls into 3 basic categories: control programs (sometimes called virtual machine monitors), alternative operating systems, and the so-called DOS Extenders. Let's take a closer look at each of these categories.

Control Programs

We can immediately dispose of the control programs, which are exemplified by Windows/386 and Quarterdeck's DesqView. These programs, although they themselves run in the 32-bit protected mode of the 80386, do not support 32-bit protected mode applications. They simply set up one or more 8086 virtual machines, each with a private 640K address space and run one 16-bit MS-DOS (real mode) application in each. The applications can't communicate with one another and can't address extended memory directly.

Alternative Operating Systems

As far as alternative operating systems go, there are currently four to choose from: Digital Research Concurrent-DOS 386 and FLEX-OS, Intelligent Graphics Corporation (IGC) VM-386 (purchased recently from Softguard), Software Link PC-MOS/386 and 80386 UNIX (several vendors, including Microport and Santa Cruz Operation).

80386 UNIX is definitely a viable candidate – if you can live with its huge memory and disk overhead, its baroque user interface, its oblique documentation (all 80 lbs of it), and all of its other little idiosyncrasies. You can buy 80386 UNIX off the shelf. It com with a complete set of programming to.. and it will let you write monstrously big multitasking virtual-memory applications. If you are content to switch to UNIX and stay there, you can start tomorrow. The UNIX market, although much smaller than the MS-DOS and OS/2 market (and likely to stay that way), is growing, and is much less price-sensitive: a typical UNIX programming tool or application costs from three to ten times as much as its MS-DOS counterpart.

However, if you think you might want to someday migrate back to the mainstream, you should adopt UNIX only after very careful consideration. The UNIX memory model, application program interface and (proposed) standard graphical user interface is vastly different than OS/2, so porting your application to 80386 UNIX now, and back to OS/2-386 later, might be a lot of extra work for relatively little return.

What about the other operating systems I mentioned? Personally speaking, I wouldn't touch a Digital Research operating system product if you paid me, no matter how fantastic it was, based on my experience with their lack of direction and their abysmal product support over the last few years. They still manage to eke out a living on their declining CP/M royalties and by selling real-time operating systems to a few high-priced vertical markets, but they've long since squandered any credibility they had in the real world where you and I must work.

I have looked carefully at PC-MOS/386 and VM/386, and both appear to be pretty solid products, but I have my doubts about their long-range prospects. IGC and Software Link are just taking advantage of a temporary window in the marketplace, and have no real depth or breadth in their product line (neither can offer you compilers, for example). Both of these operating systems can multitask MS-DOS real mode applications well, and can (theoretically) support 32-bit protected mode applications – but none of the dominant application companies is targeting either system, and you don't want to be a pioneer, do you? Sometimes pioneers are heroes. On the other hand, sometimes they are the guys with the arrows in their backs.

DOS Extenders

Now let us turn our attention to DOS Extenders. In the simplest terms, a DOS Extender is a term for a program which establishes a 32-bit protected mode environment on behalf of an application, but still lets that application use MS-DOS for file services and I/O. A DOS Extender works by loading an application into extended memory, then switching the CPU into protected mode, setting up the various descriptor tables and registers appropriately, and giving control to the application.

When the application makes a request for some operating system service such as reading a file, the DOS Extender intercepts the request, switches the machine back into real mode, performs any necessary address translation or copying of data from extended memory into conventional memory (below 640K), then passes the request on to MS-DOS. When MS-DOS returns, the DOS Extender switches the machine back to protected mode, again performs any address translations that might be needed and hands the results of the operation back to the application. Some DOS Extenders provide additional functionality too, such as interprocess communication or multitasking services.

There are currently three different sources for DOS Extenders: A.I. Architects, Intelligent Graphics Corporation (IGC), and Phar Lap software. All three products include a debugger and various utilities and allow the purchaser (by the purchase of a separate distribution license) to embed the DOS Extender into an application program for resale. Of course, to create such a program, you also need to buy an 80386-specific compiler or assembler and linker (we'll talk about these next month).

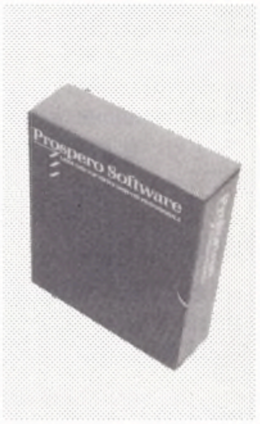
A. I. Architects sells Developers Kits for both an 80286-specific and an 80386-specific DOS Extender, called OS/286 and OS/386 respectively. The 80286 version can be used with most of the popular MS-DOS C and Fortran compilers; the 80386 version can be used only with Metaware's C and Pascal compilers. The A.I. debugger is actually an integrated command processor and machine language debugger, and can execute DOS-like commands and batch files as well as tracing, inspecting, and modifying machine language programs.

IGC's DOS Extender product is called X-AM and appears to be an incidental offshoot of IGC's VM/386 operating system project. Its documentation and debugger are the weakest of the three DOS Extenders discussed here. X-AM is currently supported by C, Fortran, and Pascal compilers from MetaWare, Language Processors Inc. (LPI), and Science Applications International Corp. (SVS).

Phar Lap's DOS Extender, called RUN386, is part of their 80386 Assembler/Linker package, which is used by almost every 80386 developer regardless of their selection of high level language. A basic debugger (similar to MS-DOS DEBUG) is included with the assembler/linker; a more sophisticated symbolic debugger (roughly equivalent to Microsoft SYMDEB) can be purchased separately. RUN386 can be used with the C, Pascal, and Fortran compilers available from MetaWare, LPI, and MicroWay. My own company has recently released an 80386 product that incorporates RUN386, and we have found Phar Lap's software products to be stable and reliable and their technical support is outstanding.

Why use a DOS Extender instead of switching to 80386 UNIX? One good reason is that you can continue to run all your favorite MS-DOS-based text processors, word processors, data bases, telecommunications programs, and the like during the development process. Although we keep hearing about 80386 UNIX's "DOS-Merge," it is not yet available in a form that is reliable or fast enough to be usable. Another good reason is that since the DOS Extenders all use MS-DOS for file system services, you don't have to reformat your fixed disk or have two incompatible partitions.

The last, and probably best, reason to use a DOS Extender instead of 80386 UNIX is: if you create an application for resale that incorporates a DOS Extender, you can simply market it on MS-DOS disks and through normal MS-DOS software channels, marking it "80386-based PC required." The user just installs and runs the program in his accustomed manner and will not usually be conscious of anything special going on, except that the application performs better than its competitors (those 32-bit registers, remember) and will be much more responsive since there is no need for overlays or other memory-conservation strategies. .EXE



Prospero Pascal for MS-DOS
Prospero Fortran-77 for MS-DOS

Professional quality for PC users

Special offer: Free £60 BIOS Library
Specially for PC users we have produced two BIOS graphics libraries, PAS-PC and F77-PC. These use the features of the PC BIOS for graphics, comms, keyboard and sound control. They are suitable for mono, CGA or EGA monitors and will set up text windows and have a full range of graphics commands. As a special offer to .EXE readers we will supply this library, normal price £60, free with Pro Pascal or Pro Fortran-77. Offer only applies to end users and is valid until 31/10/88.

Pro Pascal is a professional-quality Pascal compiler used by software houses, universities, government departments and industry throughout the world. Pro Fortran-77 is preferred for technical computing, and is widely used by engineers, scientists and research departments.

Both are available to PC users for £320 each. They are suitable for any PC using MS.DOS 2.1 or later. They are supplied on three 5¼" disks with a 250 page loose-leaf manual. With the compiler are a linker, librarian, run-time library and symbolic debugger as well as sample programs. Free hotline support is available to registered users.

If you already have a Pascal or Fortran compiler, why buy these ones? Simply, because they work. They are reliable and fast; the programs produced are tightly coded and very fast. Pro Pascal has been validated to ISO 7185, the international standard for Pascal, so you can be sure it works. Pro Fortran-77 has been validated to ANSI X3.9 1977. Both compilers have many useful extensions – Pro Fortran-77 will accept Fortran-66 code – Pro Pascal has a DOS extension library which provides advanced features such as wildcard directory search. You can transfer any program you have written using many other Pascal or Fortran compilers with a minimum of fuss. Once you have written your program you can transfer to other machines using another Prospero compiler. And, if you have access to a mainframe or minicomputer, you can restrict the compilers to using the standard language so that your program will run immediately. That's what we mean by Portability.

For more information telephone Prospero in London on 01-741 8531. Credit cards accepted for telephone orders. Or send £320 (+ 15% VAT in UK) to Prospero Software Ltd, Department 1006, 190 Castelnau, London SW13 9DH, England, saying which compiler you require. Postage is free in the UK, £5 in Europe and £10 elsewhere.

Prospero Software
LANGUAGES FOR MICROCOMPUTER PROFESSIONALS
190 CASTELNAU, LONDON SW13 9DH, ENGLAND TEL 01-741 8531 TELEX 8814396

Clarifying dBase Source Code

You're supposed to write flowcharts before you start coding. If you don't, and you're programming in dBase, there's always Clear. This is an application that will generate flowcharts, tree diagrams and structured listings from dBase source code. Colin Pratt-Hooson checks it out.

Clear is a set of tools for Program Developers which automatically generates complex flow charts and tree charts of your dBase applications. These powerful graphical tools are assisted by a formatted source printing capability to provide you with a complete set of program development and documentation tools, and the ability to see your code from many different angles.

Flow charts represent the interaction of the control statements in your programs by means of symbols and directed lines showing the program flow. Tree charts, however, present a more systematic view of your programs by depicting all the modules involved in an application and their respective places in the control hierarchy of the program. Source prints list the formatted code of your programs complete with line numbers, logical block indentation, titles, procedure names and critical dates. The printouts in this article show the different types of output that Clear can produce.

If, like me, you are one of those people who are used to bar menus, like those in dBase and Lotus 1-2-3, Clear will be very simple to operate as it too uses this menu method. When Clear is used for the first time, Global variables can be set up and saved to disk. These include the path name for source files and the type of printer you have. Installation was fairly painless, and was aided by the on-screen help which accompanies the printed manual.

Menu Structure

The main Clear menu has 4 options. These are Actions, File, Global and Exit/Run. I'll go through the File menu first. The options available on this menu are: Select a file to process, Change the directory to search, List the current directory, Rename a file and Delete a file.

When selecting a new directory, Clear allows pattern matching so you can display only .PRG files, for example. When producing a tree chart any files called by the file selected are automatically picked up and added to the tree, whereas a flow chart just charts the specified file.

Having selected a file to act on, you move to the Actions menu to actually produce the output. The menu has the following options: Diagram type, Output device, Flow chart options, Tree chart options, Source print options, Custom titles, Hard copy set-up, Page preview and GO.

Diagram Type

This option allows you to specify which of the three diagram options (flow chart, tree chart or source print) are to be produced and they are simple ON/OFF toggles. You can choose to output to screen or printer, and the printouts can be white on black if you prefer (and if your ribbon can stand the pace). All that bitmapped data takes a lot of disk space, so don't print to disk unless you really want to. My sample application (4.5K of source code) took 1.3MB of space when printed to disk. It does mean, of course, that you can then use the PRINT utility to print the file in background mode later.

Flow chart options

The Flow chart options setting determines the way in which the flow chart will appear. Full mode shows every logic step line by line, and there's also a Compressed mode that groups several lines of code together. There are also options here to change font size and to specify whether to include the source code's comments in the printout.

Tree chart options

The Tree chart options menu determines the output for tree charts. You can choose a number between 2 and 8, which is the number of sub-levels you require. You can have the boxes printed in outline or in 3D, and again you can turn off source code comments if you wish.

Source Print Option

The only option is to have logic lines on or off. These lines simply join up IF ELSE – ENDIF and DO WHILE loops etc.

Custom Titles

This menu option allows you to specify a job title, description, name of project, programmer's name and date. This information will then be added to printouts if you wish. You can also specify whether to use high or low resolution printing, and the number of copies to be printed (up to 9). Once all the options have been chosen

Colin Pratt-Hooson develops dBase applications for American Express and can be contacted through the .EXE editorial office. Clear is available from In Touch, in Mid Glam, on 0222 882334. It costs £95 for the dot matrix version, and £145 for the HP Laserjet version.

Let's go halves

See us at SOFTWARE TOOLS'88
Stand 27
14-16 JUNE 1988
WEMBLEY EXHIBITION CENTRE

In half a day we'll help you start revolutionising your systems development activities.

Isn't it time you found out how practical and powerful an automated software engineering approach can be?

Already, a large and growing number of systems developers have discovered that AUTO-MATE PLUS can pay for itself in the first phase of a single project.

- Excellent support for modern top-down structured methods, particularly SSADM*/LSDM
- Automatic generation of systems through on-line prototyping, with automatic DBMS syntax generation and automatic program code generation
- The fullest set of analysis, design and specification facilities available in the CASE market.

Discover more about AUTO-MATE PLUS at one of our half day presentations. To book places, return the coupon today or call Caroline Kiely or Diane Allum on 01-636 4213. Dates and locations are listed opposite.

**SSADM was jointly developed by the CCTA and LBMS.*

REE PRESENTATIONS:
5 JUNE - LONDON 21 SEPT - LONDON

LBMS
Learmonth & Burchett Management Systems Plc
Evelyn House, 62 Oxford Street, London WIN 9LF
Telephone: 01-636 4213 Telex: 261960 Fax: 01-636 2708

CIRCLE NO 19

It's time I discovered more about AUTO-MATE PLUS!

NAME ____________________

POSITION ____________________

COMPANY ____________________

ADDRESS ____________________

____

TEL ____________ POSTCODE ____________

NUMBER OF PLACES ____________________

DATE OF PRESENTATION ____________________

EX/6

for the selected source file, there's a special confirmation screen so that you can check all your settings before setting the printer going. It's well worth making sure that you have done everything right, as printing takes some time. I'll discuss just how long, later on.

Starting the printer

Once all the setup options have been chosen, selecting the GO option from the main menu makes the printer spring into life.

On screen you are presented with an interesting status display graphically showing how far the processing has reached. Any errors encountered in the source file are highlighted here.

Other Uses

Clear has several other features that I found useful, and these include viewing flowcharts in trace mode. This allows the user to display the flow chart on the monitor, and then to be able to step through the logical flow of the system following both Yes and No decisions, which is an extremely useful feature when debugging.

Clear can also be used in the system development cycle in the following ways: Design implementation, testing, integration, documentation and maintenance.

Design

Design is the most critical stage in Software development and Clear can help you in designing the system, as well as in communicating this design to other members of your team. Clear can help you to visualise the hierarchy of your future system with a tree chart. You can quickly generate a tree chart from only sketchy source code, and iron out potential design problems before significant programming to provide you with a graphical support for your system design concepts with very modest effort. At this stage of design Clear serves as a logic debugger (using its trace mode). It enables you to improve the basic program structure by drawing the user's attention to the logical inconsistencies in the design of the program.

Implementation

During the implementation stage of your application development, Clear can be effectively used as a checking tool. Every time you change your program procedures, you can quickly produce new flow charts. Compare the new diagrams with the ones generated during the design stage to make sure you are following the original design. Looking at the diagrams will help you to evaluate the advantages and disadvantages of the two versions. Flow charts of completed programs should be kept handy during the implementation of the rest of the system modules. This will help to avoid needless repetitions and possible logical contradictions in your code.

Testing

Although Clear is not a compiler, it can be used for consistency checking. It reports all the logical errors encountered in the process of reading and parsing the programs selected for flow chart generation. In particular Clear acknowledges errors such as open logical blocks (missing ENDIF, ENDDO or ENDCASE statements) incorrectly nested logical structures, non-executable code and some others.

Integration

When all the system components are implemented and tested, the integration stage of software development begins. It is very important to make sure that all the necessary components have been incorporated into the system. Clear helps in this respect by pointing out any unidentified and non-existing procedures on the tree

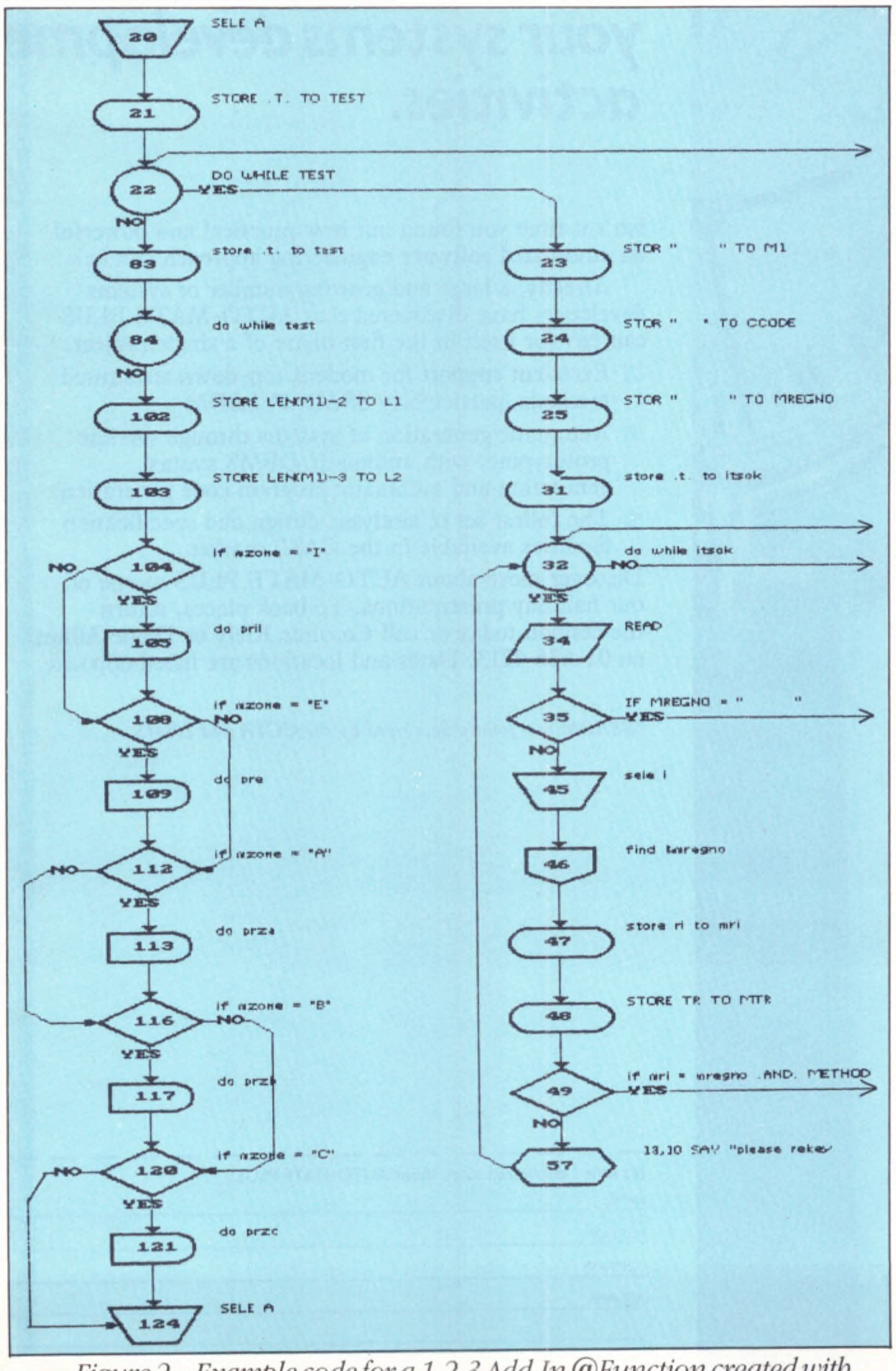


Figure 2 – Example code for a 1-2-3 Add-In @Function created with Lotus Developer Tools (continued)

chart of the main procedure. These must be written or otherwise incorporated into your application. In addition, you can view flow charts of all the system components in trace mode to make sure that all the calls are made in correct places, and all the necessary parameters are set when control is transferred.

Documentation

Documentation is typically the most time consuming and perhaps, the most tedious stage of software development. With Clear this is somewhat less tiresome with its documentation tasks – drawing the diagrams to visualise the flow of the programs and the architecture of the system. You can also use Clear to produce source listings of the whole application. These printouts will have sequential line numbering corresponding to the flow chart representation, will be paged by procedures and will also feature customised titles.

Maintenance

If, like me, you find it necessary to support other people's applications, Clear can be very useful to the user for examining the system flow.

Printers and Printing

As I have mentioned, printing takes a long time. My application (a 4.5kb file), took 55 minutes to print and produced 12 sheets of paper on my Epson dot matrix printer. Hopefully, the Laserjet version will improve on this performance. By the way, Clear prints in landscape mode only, though a portrait option will be available shortly.

Hardware Requirements

Clear is designed to run on IBM PC/XT/AT or any 100% compatible systems requiring at least 512 RAM. Monitor cards supported are CGA, EGA, Hercules. If your system has an IBM mono adapter, only hard copy and file output is possible. If your system has a Hercules graphics card you must initialise full Hercules mode prior to running Clear. If you intend to print to disk, a hard disk is essential.

System Limitations

Clear's capacity to generate complex diagrams mostly depends on the system memory that you have available. The following limitations however, prevent Clear from generating the drawing (even if your system has enough free memory). First, the largest source file that the system can read is 256K, and a maximum of 256 symbols per single tree chart (or procedures per single application, if the number of levels is not limited). A flow chart can have 128 symbols.

While working with Clear you should remember at all times that Clear is not a debugger. Though it does report encountered problems in some cases, Clear is guaranteed to work successfully only with source code which is operational and free of syntactical errors.

Verdict

If you require useful, meaningful and decent flow and tree charts for the documentation of your system and you can wait a while for output, Clear is a very useful package and it is to be recommended. If you have a laser printer, it will be even more useful as the hard copy will not take as long to arrive. I personally found Clear to be a unique and useful piece of software, and I believe it will help to improve the quality of the systems that I produce. I fully intend to hang on to the review copy of the software and to continue using it. EXE

ADDISON-WESLEY

Please tick box

☐ **PRINCIPLES OF SOFTWARE ENGINEERING MANAGEMENT** *Tom Gilb*
This practical guide is designed to assist professionals with the problems involved in developing complex software systems. The authors present a set of guidelines and tools to manage the technical and organizational aspects of software engineering projects. Some familiarity with computer organization and programming is required.
1988/463pp/0 201 192462/paper/*£17.95*

☐ **COST ESTIMATION FOR SOFTWARE DEVELOPMENT** *Bernard Londeix*
This is a practical guide to cost estimation techniques. The book begins with a survey of the role of estimation in the software development process. It guides the reader through actual estimating practice, describing effective methods for estimating software development costs.
1987/209pp/0 201 17451 0/hard/*£16.95*

ADDISON-WESLEY PUBLISHERS LTD.,
Finchampstead Road, Wokingham,
Berkshire RG11 2NZ.
Tel. (0734) 794000

FOR FURTHER INFORMATION ON TITLES IN:
Software Engineering
Operating Systems – UNIX, PICK
Programming Languages – MODULA-2, PASCAL
CIM
MIS
Artificial Intelligence etc.,

☐ Please ask for a free copy of our COMPLETE COMPUTING CATALOGUE

Please order from your local bookshop.
In case of difficulty, order direct from Addison-Wesley.

Name ______________________
Address ______________________
____

Please enclose £1.50 P&P
I enclose my cheque for £__________ OR
Please debit my *American Express/Visa/Diners Club/Access/Barclaycard (*delete as appropriate)

Account No. ______________________
Expiry Date __________ Date __________
Signature ______________________

CIRCLE NO 20

Lotus Developer Tools

In which Adam Denning looks at the Lotus Developer Tools – a product that allows you to write add-in code and @functions for 1-2-3 and Symphony.

Writing add-ins for Lotus 1-2-3 is not new. As the marketing people from 4-5-6 Word will tell you, it's been going on for a long time. However, the way that add-ins are produced has changed.

Early versions of HAL, before it was bought by Lotus, were developed by people who disassembled the 1-2-3 code by hand and added links to their own functions. Nowadays, Lotus sells a kit especially designed to help develop such add-ins. The Lotus Developer Tools, the subject of this review, are rarely seen in Lotus advertising, though, and few dealers stock them. However, if you're involved in using Lotus for commercial or in-house applications, they may be worth a look.

Terminology

In order to understand this review, you need to be aware of some of the 1-2-3 jargon. A 1-2-3 worksheet is comprised of cells, marked by column and row position within the worksheet. Each cell can be assigned a value or a formula which causes its value to be calculated automatically from the values entered into other cells. Such a formula is one form of a 1-2-3 macro, where a macro is a series of cell entries in a single worksheet column. Macros may include 1-2-3 commands and any type of cell entry, and they may also simulate special keypresses. Macros may also make use of 1-2-3 @functions, which are built-in functions which take arguments and return results, just as you'd expect from a function in any language. These 'at-functions' are specified in a formula by typing '@' followed immediately by the function name. For example, if a cell were to be filled with the sum of all the cells in the range B1 to B23, you may enter the formula

```
@SUM(B1..B23)
```

into the relevant cell. 1-2-3 will then calculate the value of that cell and display the value in it. The @function being used is @SUM and the parameter passed to it is the 'range' B1 to B23. Ranges form a fundamental part of the worksheet philosophy.

Add-Ins

A feature given to 1-2-3 (and Symphony) but not talked about much until now is the ability to attach 'add-in' applications to the program. This is where the Lotus Developer Tools come in, as it is the suite of programs which enables the programmer to take advantage of the add-in feature. Add-ins may comprise applications, as in new commands, new menus and so forth, or new @functions. An add-in can contain up to 48 new @functions. In order to allow 1-2-3 to use an add-in application, it must first be attached. This is done on a per-session basis, although a certain amount of auto-loading may be performed at start-up in the normal Lotus way. To attach an add-in, the add-in manager must first be installed in the 1-2-3 'driver set'. The add-in manager is supplied with the Developer Tools disc as the file ADN_MGR.DRV. It may be added to or removed from your desig-

ABS	ASCII-TO-FLOAT	MODULUS
ACOS	DROP	MULTIPLICATION
ADDITION	DUP	NATURAL LOGARITHM
ASIN	EXP	OVER
ATAN	FLOAT-TO-FIXED	POP
COMPARISON	FLOAT-TO-INT	PUSH
COMPARISON WITH 0	DIVISION-BY-TWO	SIN
COS	TRUNCATION	SQUARE ROOT
FLOAT-TO-LONG	INT-TO-FLOAT	SWAP
LONG-TO-FLOAT	EXPONENTIATION	TANGENT
DIVISION	LOGARITHM	
FLOAT-TO-ASCII	SUBTRACTION	

Figure 1 – Functions available to the developer in the calculator.

The Lotus Developer Tools (assembly language version) costs £126.50 including VAT. You have to order it from Lotus in Windsor, on 0753 840281. Adam Denning is a programmer with GPT MicroScope in Maidenhead, specialising in data communications such as X.25 switches and PADs. Borland has stated, by the way, that it is developing a similar package for Quattro.

"Don't buy it, David. Copy mine..."

STOP IT WITH SOFTWARE SENTINEL

EVERY DEVELOPER'S NIGHTMARE

Software theft in the UK is currently estimated to cost developers over £150m in lost revenue every year.

It happens every day.
It happens to you.

Yet you can stop unauthorised use of your software so easily with the Software Sentinel key.

GET DAVID'S MONEY

Software Sentinel is a software key which plugs into any personal computer's parallel printer port to authorise the execution of your software program. Software Sentinel controls access. Users can make unlimited copies of the software and run them wherever and whenever they wish – as long as each copy is accompanied by the appropriate Software Sentinel key.

Software Sentinel runs under DOS and versions of XENIX and on all IBM PC and PS/2 models and compatibles. Using the algorithm technique it is over 100 times faster than fixed response security devices. It does not tamper with the operating system, requires minimal implementation and is totally transparent to the user.

Software Sentinel is the world's most widely used software key with over 250,000 units worldwide already protecting £200 million of software.

SEND FOR YOUR EVALUATION KIT

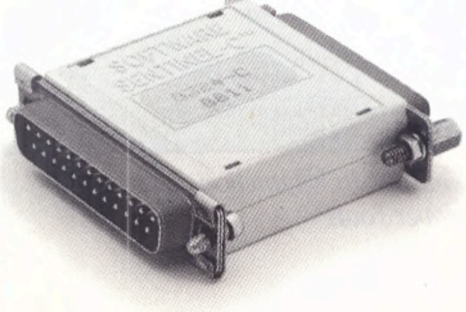
See how you can protect your revenue. Try Software Sentinel for yourself. Simply return the Freepost coupon with a cheque for £50 + VAT or telephone for further details.

Software Sentinel from California Software. Don't let David get away with it.

CALIFORNIA SOFTWARE PRODUCTS, LTD.

Shirley Lodge, 470 London Road,
Slough, Berkshire SL3 8QY, UK

Tel: 0753 41278



Registered trademarks: Software Sentinel – Rainbow Technologies Inc; IBM PC and PS/2 – International Business Machines Corporation; XENIX – Microsoft Corporation.

CIRCLE NO 21

CALIFORNIA SOFTWARE PRODUCTS, LTD.

Freepost 989
470 London Road,
Slough, Berks., SL3 8BR

YES. I WANT TO PUT A STOP TO UNATHORISED USE OF MY SOFTWARE

☐ PLEASE SEND ME AN EVALUATION KIT. I ENCLOSE A CHEQUE FOR £50 + VAT PAYABLE TO CALIFORNIA SOFTWARE PRODUCTS LTD.

☐ PLEASE TELEPHONE ME TO DISCUSS MY REQUIREMENTS.

☐ PLEASE SEND ME MORE INFORMATION.

NAME..

POSITION..

COMPANY...

ADDRESS...

....

........................TELEPHONE.................................

```
;Purpose:      This sample add-in is provided to show you, the
;              developer, the different ways you can use the
;              tools library procedures, and demonstrates the
;              basics of the add-in interface.  It also gives
;              you a guideline for structuring your own add-in
;              application.
;
;Structure:    This sample add-in is divided into the
;              following modules:
;
;              @PEEK(segment,offset) -- returns the value of the
;                      byte at a specified segment/offset
;                      memory location.  All values are decimal.
;              @POKE(segment,offset,value) -- returns the
;                       previous value of the byte
;                       at segment/offset memory location and
;                       changes the byte at the specified
;                       location.  All values are decimal.
;              @FACTORIAL(value) -- returns int(value)!
;
;              These modules are structured in such
;              a way that you can use all of them or only some
;              of them in other add-ins you may write.
;
;Potential
;Uses:         This add-in lets you look at a
;              memory location specified by a segment and offset
;              address, and put a specified value in that
;              location.  You can also calculate a
;              factorial of a specified value.
;
;NOTE:         This sample add-in is an example only.  It is not
;              intended for use as a complete product.  There
;              are many things not included in here that should
;              be in your own add-ins.  Some of its weaknesses
;              are the following:  it has very little error
;              handling included, it lacks a polished user
;              interface, and it has very limited capabilities.
;
               INCLUDE 123_mac.asm      ;bring in add-in macros
               INCLUDE 123_head.asm     ;bring in add-in equates

;developer's required entry points

               PUBLIC  funcnum          ;number of functions in add-in
               PUBLIC  functab          ;function definition table
               PUBLIC  init             ;initialization procedure
               PUBLIC  term             ;termination procedure

               EXTRN   ds_123:WORD      ;initialized in INIT procedure

;**********
; define code segment here
;**********

               startcs                  ;see 123_mac.asm for macro code

;  NOTE:  There is no data segment (startds).  A 1-2-3 Add-In
;  @Function file is a binary formatted file, not an executable file.

funcnum  equu  3       ;There are 3 @functions in this add-in
functab  equ  this byte

FUNC_DEF       @peek,_peek,2            ;Peek at segment,offset
FUNC_DEF       @poke,_poke,3            ;Poke value at segment,offset
FUNC_DEF       @fact,_fact,1            ;Factorial of number
FUNC_DEF       end

;Function names as they appear on the spreadsheet

_peek   db '@PEEK(',0
_poke   db '@POKE(',0
_fact   db '@FACTORIAL(',0

pokeval db ?              ;used to store value to poke into memory

;**********
; @function specific variables go here.
;**********

      EXTRN    flt2int:NEAR     ;pops TOS to signed integer
      EXTRN    int2flt:NEAR     ;stores integer as a float at TOS
      EXTRN    mul_flt:NEAR     ;multiplies TOS by TOS-1
      EXTRN    stackadj:NEAR    ;checks specified arguments on stack

; @PEEK(segment,offset) -- returns the value of the byte at a
; specified location.
; All values are decimal.

@peek  proc   far            ;Peek at memory at segment,offset
       mov    ax,2           ;NOTE: Put number of expected
                             ;arguments in AX.  The number
                             ;should correspond to the number
                             ;in the FUNCDEF Macro.

                             ;For all @function entry
                             ;points, start by setting AX
                             ;to the number of arguments in
                             ;the @function, call STACKADJ
       call   stackadj       ;to verify that at least
                             ;that number of arguments is
       jc     abort1         ;present, and if not then
                             ;abort processing.

       start_func            ;The START_FUNC macro
                             ;preserves 123's DS register
                             ;and makes the current DS
                             ;equal to CS to eliminate the
                             ;need for CS overrides.
```

Figure 2 – Example code for a 1-2-3 Add-In @Function created with Lotus Developer Tools.

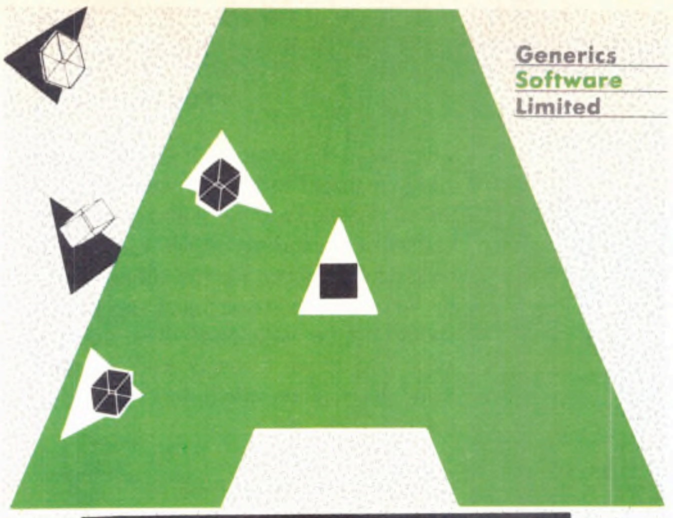
nated driver set with the utilities ADD_MGR.EXE and DEL_MGR.EXE respectively, both of which are supplied with the Developer Tools.

Lotus Developer Tools comes with software for attaching add-ins to both 1-2-3 and Symphony, and it is used in conjunction with the Microsoft macro assembler V3.00 or higher, the Microsoft linker and the EXE2BIN utility. A driver set is a 1-2-3 file which contains the device drivers for each of the input and output devices used in a given 1-2-3 configuration. For example, it would contain the Hercules screen adaptor driver if you had a Hercules video card on your system, and it would contain the Epson printer driver if you're using 1- 2-3 with an Epson printer. The add-in manager, once added to your driver set, causes the keypress ALT-F10 to have a function.

The Developer Tools package also allows the developer of add-in @functions to perform various degrees of worksheet manipulation.

When this key combination is pressed during 'READY' mode, a new menu is displayed. This menu has options Attach, Detach, Invoke, Clear, Setup and Quit.

The Attach option allows you to attach an add-in to the current 1-2-3 session. If selected, it provides a list of files found in the current directory which have the extension .ADN and allows one to be selected. This selected add-in file is then loaded and its contents become, effectively, part of 1-2-3. The detach option allows a previously-detached add-in to be removed from the system (ie from the current session of 1-2-3). Add-ins comprised of @functions may not be detached. The third option, Invoke, allows an attached add-in to be run. It presents a list of the add-ins which have been attached and allows you to select one; once a selection has been made, the chosen add-in is executed. If the selected add-in contains @functions, it is not an application as such



AnimAID

DISPLAY THE POWER OF ADA™

AnimAID lets you build better operator efficiency into your Ada application through the use of sophisticated graphical interfacing techniques. Combining object-oriented applications building blocks with a graphical interface generation tool, AnimAID brings a wealth of ingenuity to your Ada applications.

AnimAID facilitates the tailorability of the applications interface to ever changing end user requirements. Here are some of the benefits of using AnimAID.

- Configurable graphics user interface
- Standardisation of development method
- Reusability of applications modules
- Highly reliable package generation
- Rapid prototyping of graphical interfaces
- Graphic-oriented applications prototyping
- Maximum use of graphics with no direct graphics programming

We'd like to tell you more about AnimAID. Please call us.

Please send me an information pack:

Name ______________________

Position ______________________

Company ______________________

Address ______________________

Postcode ______________________

GENERICS

Ada is a registered trademark of the U.S. Government (AJPO)
Generics and AnimAID are trademarks of Generics Software Limited

Generics Software Limited,
7 Leopardstown Office Park,
Foxrock, Dublin 18 Ireland.
Telephone 0001 954012. Telex 2265871 MONREF G
Quoting ref 74:EIM229. Fax 0001 954011

CIRCLE NO 22

System Science

C COMPILERS

AZTEC

AZTEC C86/PROF (MS-DOS) large mem, assm, link	£145.00
AZTEC C86/DEV (MOS-DOS) adds source debug, editor	£195.00
AZTEC C86/COMM (MS-DOS) adds ROM support, lib source	£295.00
AZTEC C68/COMM MAC, AMIGA, ATARI ST	£295.00
AZTEC C for PC/M-80, Apple Dos, Prodos	£ call
LATTICE C Compiler ver 3.21	£235.00
LATTICE C Sprite debugger	£129.00

MICROSOFT C 5.0 with Code View, Quick C large mem, source debugger, 8087	£245.00
MICROSOFT QUICK/C with debugger & editor	£65.00
TURBO C from Borland Editor, Linker, Large mem options	£65.00
RUN/C- PROFESSIONAL Interp	£125.00
C CROSS Compilers 6809, 680?0, 8051, 6301, 80386 8096, 63HC11	£Call

C LIBRARIES

GRAPHICS

METALWINDOWS — many languages	£115.00
ESSENTIAL GRAPHICS Library	£175.00
GRAPHIC — source, colour, EGA	£245.00
HALO — specify compiler	£175.00

ISAM & DATALIFE

C TREE — Faircom (source)B tree lib.	£225.00
R TREE — Reports generator for C Tree	£175.00
Btrieve — database library, many lang	£175.00
Btrieve/N — Novell, PC-NET networks	£395.00
C-ISAM for Microsoft or Lattice C	£225.00
LATTICE dBC-III Dbase library	£175.00
LATTICE dBC-III Plus for networks	£395.00

PHOENIX

PLINK-86 Plus — overlay cacheing	£245.00
PMATE-86 programmers editor	£120.00
PRE-C Lint utility	£175.00
PFIX-86 Plus debugger	£195.00
PDISK — backup, disk cache & utils	£95.00

EDITORS

BRIEF multi-file macro editor	£145.00
dBRIEF addition for DBASE programmers	£75.00
EPSILON — Emacs style editor	£145.00
PC/VI screen editor	£125.00
Lattice Screen Editor	£95.00

SCREEN and DATA ENTRY

PANEL entry screens — most languages	£215.00
DATAWINDOWS with source	£175.00
Vitamin C — Screen, print layout code	£145.00
Windows for Data, Windows for C	£215.00

GENERAL

GREENLEAF General Functions Dos, Disk, Video, Strings, Date, Keybd	£125.00
PforCe–source lib, comms, dbase, screen etc. The C programmers toolkit	£225.00
BLAISE C TOOLS PLUS — source DOS, Dos, screen, windows, keyboard etc	£125.00
C Scientific Subroutines — source Matrix, polynomial, diff. eqn. etc	£145.00
/RESIDENT/ by ESSENTIAL TSR LIBRARY	£125.00

COMMUNICATIONS

GREENLEAF Communications	£115.00
BLAISE Asynch Manager	£125.00
ESSENTIAL Communications	£125.00
ESSENTIAL Breakout for Comms	£95.00

MKS Toolkit lots of Unix goodies, inc VI	£95.00
QUILT Software Revision Management	£95.00
PC-LINT by Gimpel	£99.00
CLIPPER — DBase III Compiler	£545.00
PCVS Personal by Polytron	£115.00

LMI Forth-83

PC/FORTH asm, Editor, DOS access	£125.00
PC/FORTH Programmers package with 8087 graphics	£195.00
UR-FORTH high perf., 8087, graphics (ver for 386)	£245.00
METACOMPILERS 8080, Z80, 8096, 6303 etc etc	

ASSEMBLERS and CROSS-ASM

Microsoft 8086 macro Asm (CODEVIEW & LINK)	£95.00
Cross Assemblers 68xx, 68000, Z80, 8080, 6502. 8048, 8051 etc	
Simulators – Z80, 8048, 8051 etc	£ call
Quelo 68000 and 68020 cross assemblers, simulators	

TURBO PASCAL 4.0

TURBO PASCAL 4.0	£55.00
Turbo Tutor	£45.00
Turbo Database or **Graphics** or **Editor** toolbox	£55.00
Turbo Numerical toolbox	£55.00
Turbo Pascal Developers Library	£215.00
Blaise Power tools plus	£85.00
Blaise Asynch plus	£85.00
Turbopower Professional source	£75.00
T-Debug plus	£75.00

FORTRAN/PASCAL/BASIC

Fortran Graphics, Scientific Libraries	£ Call
Microsoft FORTRAN-77 with Codeview	£245.00
Pro-FORTRAN-77	£275.00
RM-FORTRAN-77 with RM-FORTE toolset	£475.00
PRO-PASCAL MS-DOS	£275.00
Microsoft Pascal.	£185.00
QUICK BASIC — Microsoft (PC-DOS only)	£65.00
PASCAL 2 full features, Turbo compatible	£325.00
TURBO BASIC FROM Borland	£65.00

LISP and PROLOG

MuLISP-87 Interpreter	£175.00
MuLISP-87 Interpreter & Compiler	£225.00
PC Scheme — Integrated LISP with OOPS from TI	£95.00
Golden Common LISP, TRANSLIP & more available	
TURBO PROLOG (PC only)	£69.00
micro-PROLOG PROF (full mem., wind.)	£245.00
micro-PROLOG entry level version	£95.00

MATHS etc

MuMATH-83 Algebraic maths system	£225.00
LINDO Linear programming	£385.00
MATH-CAD ver 2.0 engineers graphics and spreadsheet	£345.00
EUREKA from Borland	£85.00
Microstat from Ecosoft	£295.00
STATGRAPHICS	£595.00
METRISTAT for quality control statistics	£695.00

TEXT WINDOWS, COMMUNICATION, DISK

VENTURA Desktop Publishing System	£795.00
FINAL WORD 2— author's WP new ver	£275.00
MicroTEX — scientific typesetting	from £395.00
WINDOWS II by Microsoft	£75.00
WINDOWS II Software Dev. kit	£295.00
SUPERKEY	£65.00
SIDEKICK PLUS (new)	£125.00

ZAP Communications—VT100, 4010 emulation	£85.00
PC/Intercomm — VT100, 102 Kermit etc	£95.00
UNIFORM — PC r/w, format CP/M disks	£65.00
CONVERT — PC add formats	£89.00
UNIDOS run CP/M software on PC	£65.00
NORTON UTILITIES advanced	£95.00
Dan Bricklin's DEMO program	£65.00

MODULA-2

Pecan Modula-2 Powersystem	£95.00
Pecan Modula-2 Advanced Pak	£195.00
Logitech Modula-2/86 Compiler	£69.00
Logitech Modula-2/86 Development System	£145.00

OS/2 & 386 PRODUCTS

Many new products are starting to appear for the **386. Assemblers, C Fortran, and Pascal** Compilers Operating systems and extensions, **Pharlap, PC-MOS, XENIS 386, VM386 etc.**

All prices are exclusive of VAT but inclusive of Inland shipping.

A selection from our extensive range. Please enquire if the item you require is not listed

**6-7 West Smithfield,
London EC1A 9JX
Tel: 01-248 0962
BTGOLD 76: CJJ028**

CIRCLE NO 23

Figure 2 continued – Example code for a 1-2-3 Add-In @Function created with Lotus Developer Tools.

```
        call    flt2int         ;get offset off stack into ax
        push    ax              ;and push the offset

        call    flt2int         ;get segment off stack into ax
        push    ax              ;and push the segment

        xor     ax,ax           ;zero out ax before peek
        pop     es              ;pop segment register into es
        pop     si              ;pop offset register into si

        mov     al,byte ptr es:[si]     ;get byte at es:[si]
        call    int2flt                 ;and put it on the stack

                end_func                ;Use this macro as the last
                                        ;step for all procedures that
                                        ;are @function calls. It
                                        ;restores DS to 123's DS.

abort1:                                 ;if not enough arguments
                                        ;passed by function
                ret                     ;return to 123.

@peek           endp
;
; @POKE(segment,offset,value) -- returns the previous value of the byte
;                                at segment,offset memory location and
;                                changes the byte at the specified
;                                location. All values are decimal.
;
@poke           proc    far             ;Poke at memory at segment,offset
                mov     ax,3            ;Number of expected arguments
                call    stackadj        ;check number of arguments
                jc      abort2          ;not enough parameters
                start_func              ;preserve and switch DS
                call    flt2int         ;get the byte to poke into ax and
                mov     pokeval,al      ;save in byte variable
                call    flt2int         ;get offset off stack into ax
                push    ax              ;push offset onto stack
                call    flt2int         ;get segment off stack into ax
                push    ax              ;push segment onto stack
                xor     ax,ax           ;zero out ax before peek
                pop     es              ;pop segment register into es
                pop     si              ;pop offset register into si
                push    si              ;preserve si for poke
                push    es              ;preserve es for poke
                mov     al,byte ptr es:[si]     ;get byte at es:[si]
                call    int2flt                 ;and put it on the stack
                pop     es                      ;restore es for poke
                pop     si                      ;restore si for poke
                xor     ax,ax                   ;clear ax for poke value in al
                mov     al,pokeval              ;put byte to poke in al
                mov     byte ptr es:[si],al     ;and put it in memory
                end_func                        ;restore DS
abort2:                                         ;if not enough arguments
                ret                             ;return to 123.
@poke           endp
;
;@FACTORIAL(value) -- returns int(value)!
;
@fact           proc    far
                mov     ax,1            ;Number of expected arguments
                call    stackadj        ;check number of arguments
                jc      abort3          ;not enough parameters
                start_func              ;preserve and switch DS
                call flt2int            ;get value off stack into ax
                push ax                 ;and preserve on program stack
                mov  ax,1               ;put 1 on number stack
                call int2flt            ;as floating point and
                pop  ax                 ;restore ax to original value
falp:           cmp ax,1                ;check to see if we are done
                jle fend                ;ax <= 1 then end
                push ax                 ;preserve ax
                call int2flt            ;and put it back on the number stack
                call mul_flt            ;for multiplication TOS * TOS-1
                pop ax                  ;restore ax for next pass
                dec ax                  ;at lower value
                jmp falp                ;and loop
fend:           end_func                ;restore DS
abort3:                                 ;if not enough arguments
                ret
@fact   endp
;**********
;  End of the @function procedure area
;**********
init    proc    far                     ;dummy routine for initialization
                                        ;during the add-in loading sequence
                                        ;prior to availability of the
                                        ;123_TOOL.LIB procedures.
                                        ;This point is also called
                                        ;when returning to 1-2-3 from the
                                        ;System command.
        xor     ax,ax                   ;zero AX to return with error code clear
        ret
init    endp
term    proc    far                     ;dummy routine for termination
                                        ;processing when either
                                        ;/System or /Quit is called
        ret
term    endp
        endcs                           ;macro to end the segment

     END
```

and so cannot be executed. In this instance, an error is generated. The Clear option causes all currently-attached add-ins to be removed from the system, while the Setup option allows up to eight add-ins to be autoloaded by the add-in manager whenever 1-2-3 is started. It also allows the add-in manager to auto-invoke any one of the auto-loaded add-ins. It also allows one of the three keypresses ALT-F7 to ALT-F9 to be assigned to given add-ins, although add-ins with more than one function generally should not be assigned to these function keys. The add-in manager itself contains some @functions which may be used in macros. These are:

@ISAAF("function_name")

returns 0 if the function named as "function_name" is not attached and 1 if it is.

@ISAPP("application_name")

which returns 0 if the add-in application named as "application_name" is not attached and 1 if it is.

Naturally enough, sample add-ins are provided with the Developer Tools system. The example application add-in adds three potentially useful debugging commands to 1-2-3. The first dumps the current contents of the microprocessor's register to the worksheet. The second dumps an area of the stack to the worksheet and the third dumps an area of memory in hex and ASCII to the worksheet. This add-in also makes use of 1-2-3's context- sensitive on-line help facility to show you how to access the help system from within an add-in application. The example @functions add-in provides three new @functions: @PEEK, which takes a segment and offset and returns the byte contents of the specified location, @POKE, which takes a segment, an offset and a byte value and places the byte value into the specified memory location, and @FACTORIAL, which returns the integer factorial of its parameter.

Adding @functions

To create add-in @functions for 1-2-3, the Developer Tools comes with a skeleton assembly language source file, with copious comments and 'holes' in which you can place your code. The task is simplified by the provision of numerous library routines to implement almost all of the functionality a developer needs. For example, a classic floating-point calculator is provided which uses the floating-point stack mechanism to give the developer the functions listed in Figure 1.

Other functions provided by the Developer Tools package allow the developer of add-in @functions to perform various degrees of worksheet manipulation. For example, it is a simple subroutine call to get the contents of a specified cell, and a routine is provided to perform argument checking and stack adjustment, making it very easy for an @function to abort if it detects it does not have enough arguments.

To generate an @function add-in for 1-2-3, one takes the @function add-in skeleton assembler source file, which includes two Lotus header files; one contains macro definitions and the other contains equates. The module must provide four public entry points in order to be attached; the first is an absolute number representing the number of @functions in the add-in, the second is the start of the function definition table, the third is the module's initialisation routine and last is the address of the module's termination processing routine. The function definition table comprises a list of a pointer to each function's name and entry point, followed by the number of arguments expected by the @function. The list is terminated by a double word of -1, and Lotus provides a macro to make the table definition a very straightforward operation. The function name itself must be stored as a null-terminated ASCII string starting with '@' and ending with an opening parenthesis. Developer Tools library routines are used by declaring them as external and specifying the library at link time. With MASM 5, the INCLUDELIB directive could be used to specify the Developer Tools library as a default library. Other external routines, such as some you may have developed in other contexts or in other add-ins, may also be incorporated.

The entry point for each procedure must follow a set of simple rules. For example, each procedure must be a FAR procedure, the STACKADJ, START_FUNC and END_FUNC macros must be called in the appropriate places and a value must be returned on the stack. Apart from that, @function routines may do almost anything. The initialisation and termination routines do not need to do much in @function add-ins, so they are essentially dummy procedures. The developer of the add-in application does not have such a straightforward task, but it is still fairly simple to code an add-in. Both application and @function add-ins are binary images created from the linker output using EXE2BIN, so total code AND data space is limited to 64K. It seems rather unlikely that this would ever be a problem. The add-in application must provide a whole lot of public entry points, the majority of which may be dummied to a single return instruction unless your application actually needs to provide them. Whenever an add-in is invoked, either by selecting the Invoke option from the add-in manager's menu or by pressing the requisite function key, the public add-in routine xdv_main is called. Whenever the user performs a given operation or set of operations on a worksheet, your application may be informed through one of its public xdv_ routines. For example, if the user selects a command which moves ranges of worksheet cells, such as Copy or Insert, 1-2-3 adjusts any affected ranges and calls xdv_adjust inside your add-in. Likewise, if the user clears an entire worksheet, you are informed through your public routine xdv_clear. If your add-in application needs to be invoked on a regular or frequent basis, the public routine xdv_poll will be called automatically by 1-2-3 whenever 1-2-3 is waiting for keyboard input. Most usefully, from the user's point of view, is that xdv_help is called if the user presses HELP when your add-in is active.

Apart from the floating-point calculator, the Developer Tools library provides routines to move the cell pointer about the worksheet, to scan a rage of cells and perform an add-in action on each active cell found, to get the size of the worksheet and of the text screen, to get the current directory and the current worksheet name, to put information and get information from a cell, to execute macros and so forth. There also is a whole set of routines which allow a logo to be displayed as well as memory allocation routines and extensive (albeit 1-2-3-style) menu-processing. There really doesn't seem to be very much that the developer cannot do inside an add-in application. Presumably all of the DOS is available to you as well, but this isn't covered in the manual. It would be quite interesting to write an add-in which allowed Quattro to be invoked from inside 1-2-3, using the DOS EXEC system call! Perhaps that's taking an add-in application a little too far.

Lotus Developer Tools includes a set of batch files for linking object files with the requisite libraries, generating an add-in application or set of @functions as appropriate. Two discs are supplied, one or 1-2-3 and one for Symphony. Symphony has rather more functionality than 1-2-3 so more tools are available to the 1-2-3 developer. The manual supplied with the package is generally very helpful but seems incredibly thin. At least a third of the documentation inside the A5-ish binder seems to be taken up with addenda or licence agreement details.

If you, as an application developer, believe that there's money to be made (or productivity to be enhanced if you're developing for in-house needs) in this field, then it seems that Lotus Developer Tools is an essential requirement. One thing for sure, it certainly makes Lotus 1-2-3 far more interesting!

.EXE

Version Numbers

This list will be published monthly.
Software vendors should send updates to:
The Editor at 10 Barley Mow Passage, London W4.

Contact details:

Ashton Tate (0628) 33123;

Borland (0734) 320022;

Digital Research (0635) 35304;

IBM 01 928 1777;

Management Innovation (0634) 827755.

Microsoft (0734) 500741;

Nantucket (0992) 554621;

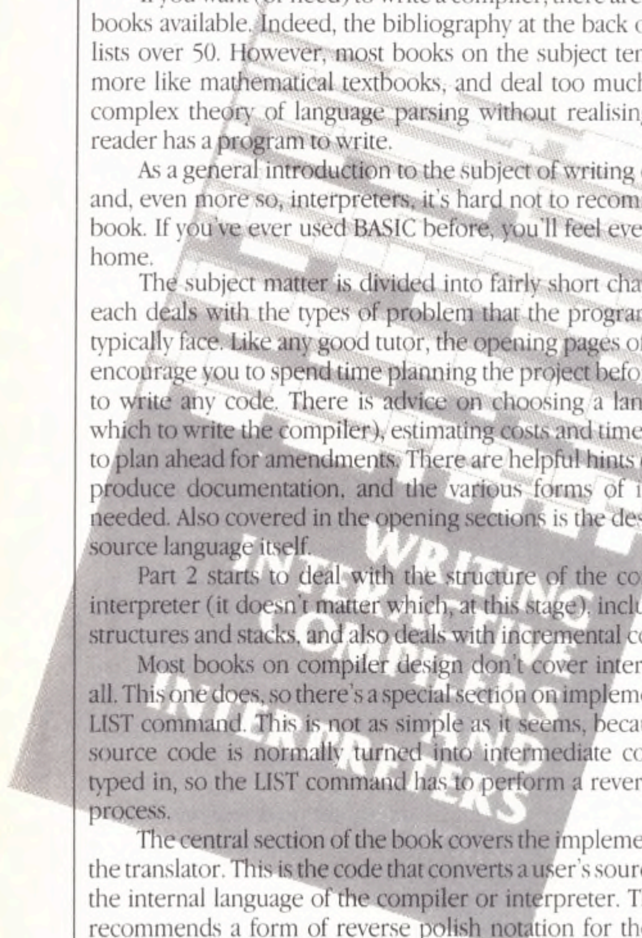
Real Time Products (021236) 8070

Sky Software (0527) 36299;

UnderWare (via Grey Matter) (0364) 53499;

Package	Publisher	Latest Version	Improvements over previous version
Brief	UnderWare	2.0	---
C	Microsoft	5.0	Improved optimisation. Includes Quick C
Clipper	Nantucket	Sum '87	More functions. Faster indexing/search
Cobol	Microsoft	2.2	---
dBase III	Ashton Tate	1.1	Increased limits on file sizes etc
dBase II	Ashton Tate	2.34	---
dBase III+	Ashton Tate	1.1	Increased limits on file sizes etc
Fortran	Microsoft	4.01	---
Foxbase	Mgmt Invtn.	2.0	73% faster than 1.21
Gem Desktop	Dig. Research	3.0	Faster. More fonts. Still single-tasking
MASM	Microsoft	5.0	80386 & Codeview support
MS-DOS/PC-DOS	MS/IBM	3.31	Special Compaq version for HDD > 32MB
OS/2	MS/IBM	1.0	1st release. No presentation manager
Pascal	Microsoft	3.32	---
Pascal-2 MS-DOS Native	Oregon S/W	2.1M	Improved optimisations, more sophisticated debugger, ROMable library code.
Pascal-2 MS-DOS 68000 Cross Compiler	Oregon S/W	2.1J	Highly optimising compiler, certified to ISO standard,includes stand-alone library.
Quick C	Microsoft	1.0	---
QuickBasic	Microsoft	4.0	Hercules & 8087/287 support, faster
Sidekick	Borland	1.56B	Calculator bug cured from 1.56A
Skybase 4	Sky S/W	23.3	Increased functions and Xenix support
Turbo C	Borland	1.5	Graphics libraries improved
Turbo Pascal	Borland	4.0	Programs can be > 64K.
Turbo Prolog	Borland	1.1	---
Turbo Basic	Borland	1.1	Hercules support
Windows	Microsoft	2.03	Overlapping windows. VGA support.

Books



Writing Interactive Compilers and Interpreters

If you want (or need) to write a compiler, there are dozens of books available. Indeed, the bibliography at the back of this one lists over 50. However, most books on the subject tend to read more like mathematical textbooks, and deal too much with the complex theory of language parsing without realising that the reader has a program to write.

As a general introduction to the subject of writing compilers and, even more so, interpreters, it's hard not to recommend this book. If you've ever used BASIC before, you'll feel even more at home.

The subject matter is divided into fairly short chapters, and each deals with the types of problem that the programmer will typically face. Like any good tutor, the opening pages of the book encourage you to spend time planning the project before starting to write any code. There is advice on choosing a language (in which to write the compiler), estimating costs and time, and how to plan ahead for amendments. There are helpful hints on how to produce documentation, and the various forms of it that are needed. Also covered in the opening sections is the design of the source language itself.

Part 2 starts to deal with the structure of the compiler or interpreter (it doesn't matter which, at this stage), including data structures and stacks, and also deals with incremental compiling.

Most books on compiler design don't cover interpreters at all. This one does, so there's a special section on implementing the LIST command. This is not as simple as it seems, because users' source code is normally turned into intermediate code as it's typed in, so the LIST command has to perform a reversal of this process.

The central section of the book covers the implementation of the translator. This is the code that converts a user's source code to the internal language of the compiler or interpreter. The author recommends a form of reverse polish notation for the internal language. Much information is presented on how to parse lines and ensure that they conform to the syntax on the source language, and what to do if they don't. This includes the generation of error messages, and the recovery from them, as well as how to tell a user exactly where an error occurred in his listing. This is the part of compiler design that tends to be treated mathematically by other books – this one keeps everything simple and practical instead.

Anyone who has to design and write a compiler or interpreter for any programming language will find this book an excellent starting point. However, don't expect to find pages of source code at the back – the book tells you how to do the job, but doesn't do it for you. Incidentally, if you want to see what complete interpreters or compilers look like, there are a few available on bulletin boards that can be downloaded. Two that I know of are TBASIC, a very small Basic interpreter written in 8086 assembly language, and Augusta, a tiny Ada compiler written in Turbo Pascal version 3.

Author: Peter Brown
Publisher: John Wiley
Price : £8.95
ISBN : 0-471-10072-2
265 pp

Programmer's guide to PC and PS/2 Video Systems.

This is one of the few Microsoft books which describes only non-Microsoft products. It attempts to pull together most relevant information for the programmer involved in programming IBM standard graphics adaptors. It doesn't cover the techniques needed by those using operating system facilities; instead it limits its attentions to the wonderful world of the BIOS and below.

Richard Wilton has approached the problem from a well-trodden route. Until the advent of this book, graphics programming for IBM involved several stages. First, the official technical documentation has to be deciphered, then various programs are written which demonstrate many of the features the technical references cover inadequately, and finally the target software gets produced. Along the way, enough background information has been picked up to write a book. Which is what's happened with this guide.

From MDA to VGA, most common adaptors are covered. This includes the fairly rare (at least in the UK) Hercules' InColor card but not the PGA. There is a detailed discussion of BIOS services and hardware considerations, together with a plethora of program fragments demonstrating the points made in the text. If you're a C programmer, by the way, be warned that all the code is in assembly language.

The information is presented as parallel threads belonging to each card, so that in the chapter headed Lines a description of various algorithms is followed by separate implementation details for each adaptor. Although this is a good way to present the large amounts of specific information needed, the layout can sometimes slightly obscure the boundaries between each lump, making reading the book from cover to cover quite hard work. Also, the index is a little weak – five pages for a five hundred page book means that sometimes some burrowing is needed, but the information is almost always there in the end. It's presented in a usable form, too.

There is stuff here for video games programmers as well as for business or presentation purposes. Sections on animation, circles, bit-oriented text painting and high-level language considerations complement the more basic information, and the 'Advanced' chapter covers the real esoterica such as light-pens and EGA split screen modes. About the only thing I've ever needed to know which I couldn't find in this book was a list of tricks to work out what mode an adaptor is in – some programs set modes without using the BIOS, and determining whether this has happened is a tricky business.

These are slight criticisms of an otherwise impeccable work. Wilton makes the point that 'Video programming is like swimming: it's one thing to read about it but quite another to try it yourself.' Quite so, and this book is not so much a lifebelt, more a six-month course with Duncan Goodhew.

It's nice to be able to end a review with the two words – 'Buy it!'.

Author: Richard Wilton
Publisher: Microsoft Press
Price : £22.95
ISBN : 1-55615-103-9
530 pp

REAL TIME SYSTEMS

C CROSS DEVELOPMENT KITS

INTRODUCING A RANGE OF PC-HOSTED DEVELOPMENT TOOLS

- ★ Draft ANSI-conforming C compiler
- ★ XA8 - hand coding macro assembler
- ★ MIMIC - microprocessor simulator/debugger
- ★ Object module tools
- ★ C DOCUMENTOR & C SCAN - C source development tools

TARGET PROCESSOR SUPPORTED INCLUDE-

- ★ MC68000
- ★ MC68020
- ★ I8086/80186/80286
- ★ MC6809
- ★ MC68HC11
- ★ TMS7000

Real Time Systems
PO Box 70
Viking House
Nelson St.
Douglas I.O.M.

CONTACT:
SALES 0624-26021

CIRCLE NO 27

NOW SHIPPING ... NOW SHIPPING ... NOW SHIPPING ... NOW

designer C++

SPEED

Inheritance and virtual functions mean you can develop re-usable bug-free code quickly.
C++ is the fastest executing object-oriented language so far invented.

QUALITY

C++ was developed by AT&T as the successor to C
C++ is a compatible superset of proposed ANSI C
C++ fixes the problems that large-scale projects encounter using C
Glockenspiel's test suite contains over seven megabytes of C++ source code
designer C++ has been in regular use at Microsoft, Lotus and Ashton-Tate for several months.

WINDOWS

designer C++ works with Microsoft C 5.0, Windows, Quick C and the OS/2 Developer's Kit.

- far and near objects
- Local and Global object heaps
- far and near methods
- dynamically linked method libraries

Glockenspiel

designer C++ is the only choice for Windows and OS/2 developers
19 BELVEDERE PLACE, DUBLIN 1, IRELAND. TEL: +353-1-364515 (FROM UK) 0001-364515.
FAX: +353-1-365238 (FROM UK) 0001-365238.

CIRCLE NO 28

TRAINING FOR EMBEDDED MICROPROCESSOR SOFTWARE DESIGN

● THE COURSES

- **C Programming for microprocessor development.**.....5 Days
- **Embedded microprocessor programming.**...1 Day
- **Microprocessor programming in C.**...................3 Days
- **Advanced C for embedded systems.** ....1 Day
- **High-level debugging workshop.** ....2 Days

● WHO SHOULD ATTEND?

- Any engineer or manager responsible for developing electronic or computer based products.

● WHAT DO WE OFFER?

- Rigorous training in a stimulating interactive atmosphere.

● THE SCHEDULE

- **PLEASE CALL US FOR DETAILS**

MICROTEC RESEARCH

Ringway House, Bell Road
Daneshill, Basingstoke
RG24 OFB England
Telephone: (0256) 57551
Telex: 858893
Fax: (0256) 57553

CIRCLE NO 29

Regina v. Schifreen & Gold

The following is the complete text of the judgement in the case of The Great Prince Philip Prestel Hack, as handed down in the House of Lords on 21st April 1988. One year earlier, the hackers had been acquitted of the offence in the High Court, and the appeal which this judgement dismisses was brought by British Telecom in an attempt to have the charges re-imposed. The appeal was heard by 5 Law Lords. The following text was prepared by just one of them, namely Lord Brandon of Oakbrook, and each of the 4 other Lords simply indicated his agreement in a single sentence, which is not repeated here.

" My Lords,

On 24 April 1986 in the Crown Court at Southwark (Judge Butler Q.C. and a jury) the respondents Schifreen and Gold were convicted of a number of offences under the Forgery and Counterfeiting Act 1981. On 17 July 1981 the Court of Appeal (Criminal Division) (Lord Lane C.J., Leonard Rose JJ.) allowed the respondents' appeals and quashed their convictions [1988] Q.B. 1116. On 31 July 1987 the Court, on the application of the Prosecution, certified that points of law of general public importance were involved in its decision to allow the appeals, but refused leave to appeal to you Lordships' House. On 16th November 1987 the House gave leave to appeal.

The case arose out of certain activities, known as "computer hacking", which the respondents carried out between October 1984 and January 1985. These activities consisted in the obtaining of unauthorised access to various computers comprised in the Prestel Computer Network owned and operated by British Telecommunications Plc.

The indictment of which the respondents were convicted contained nine specimen counts, five against Schifreen and 4 against Gold. Except for the date and the particular computer involved the particulars of offence in each count were in similar terms, alleging that the respondent concerned

"made a false instrument namely a device on or in which information is recorded or stored by electronic means with the intention of using it to induce the Prestel computer to accept it as genuine and by reason of so accepting it to do an act to the prejudice of British Telecommunications Plc."

I take the relevant facts mainly from the judgement of the Court of Appeal (Criminal Division) delivered by Lord Lane C.J. Prestel is an information service which allows persons who have the necessary micro-computers to call up a computer data base and to receive information from it. Access to the Prestel computer is by way of the British Telecom Telephone System. When the computer's number is dialled, the telephone system connects the dialler to the appropriate Prestel centre.

In order legitimately to obtain information from the Prestel data base, it is necessary to register as a user by filling in an application form and paying a rental charge. Once that is done, the user is allocated a customer identification number (CIN) consisting of ten numerals and also a password.

The user has, as part of his equipment, a television screen and a keyboard. After he has become connected to the particular Prestel Computer which he has dialled, a picture will appear on his television screen inviting him to "log- on". He then types out his CIN which is verified by the computer. Next the user types out his password which is also verified by the computer. If the computer has been able to match up the CIN and the password with the user information which it has in its memory, the user is then admitted to those parts of the Prestel data base which he has been authorised to use. He can then type out his request for information on the keyboard.

The detail of the process by which the computer verifies the CIN and the password is this. When the user telephones the computer, the call is answered by a device called a "port". A port is analogous to a doorway into the system. There is one port for each telephone line. The port tells the computer that a new call has been made and that the user needs to be shown the log-in frame. The log-in frame is then show to the user on his television screen. The user then types out his CIN. This passes down the telephone line as a series of electronic impulses. There is a part of each computer which is reserved for dealing with the input and output and the control of each port.

The CIN is received in a particular area reserved for that port alone. That area is called a user segment. Each user segment has three areas: (1) the input buffer (2) the control area and (3) the output buffer. The word buffer in this context is used to describe an area in a computer which is used to receive information from, or to transmit information to, the outside world. When the user types out his CIN, that is received in the input buffer and is then immediately moved into the control area and retained there for the duration of the log-in procedure. This may be a very brief time indeed.

In order to verify the customer's identity and the password, the user's details must be accessed from the data base to determine whether he is a valid user or not. It is the function of the log-in procedure to compare the CIN with the information already contained in the user file. If there is no CIN on the user file which matches that which has been typed out by the user, then the user is given up to three attempts to repeat his CIN. The same procedure is then adopted with the password. If these two checks are completed successfully, the user is then allowed into the Prestel computer.

After these operations are over, the areas in the user segment that were used for the CIN and the password are cleared. Thereafter the input buffer is used to receive the instructions being typed out by the user and the output buffer is used to show the information that has been requested on the user's television screen.

On numerous occasions between October 1984 and January 1985 each of the respondents, using his own microcomputer at his home, gained unauthorised access to a number of prestel computers. They did so by obtaining and using the CINs and passwords of others without their permission. Having gained such access they obtained information to which they were not entitled; made unauthorised access to stored data; and caused charges to be made to account- holders without their knowledge or consent.

The respondents' object in carrying on these activities was not so much to gain any profit for themselves as to demonstrate their skill as "hackers". It never occurred to them that they might be committing any offences under the Forgery and Counterfeiting Act 1981.

That Act provides, so far as material, as follows:

1. A person is guilty of forgery if he makes a false instrument, with the intention that he or another shall use it to induce somebody to accept it as genuine, and by reason of so accepting it to do or not to do some act to his or any other person's prejudice.

Interpretation of Part 1

8.(1) Subject to subsection (2) below, in this Part of this Act 'instrument' means –

(d) any disc, tape, sound track or other device on or in which information is recorded or stored by mechanical, electronic or other means.

9.(1) An instrument is false for the purposes of this Part of this Act

(a) if it purports to have been made in the form in which it is made by a person who did not in fact make it in that form; or

(b) if it purports to have been made in the form in which it is made on the authority of a person who did not in fact authorise its making in that form;

10.(3) In this part of the Act references to inducing somebody to accept a false instrument as genuine, ... include references to inducing a machine to respond to the instrument ... as if it were a genuine instrument....

(4) Where subsection (3) above applies, the act or omission intended to be induced by the machine responding to the instrument... shall be treated as an act or omission to a person's prejudice.

My Lords, as appears from Section 1 of the Act of 1981 set out above, the primary ingredient of the offence of forgery is the making of a false instrument. Moreover, the instrument must be one which comes within the definition of that expression in section 8(1); and it must be false within the definition of that expression in section 9(1). The difficulty in the present case is to identify the false instrument which the respondents were accused of having made.

According to the common language of the nine counts in the indictment the false instrument made by the respondents was a "device on or in which information is recorded or stored by electronic means". This language was clearly used in an attempt to bring the case within the definition of the expression "instrument" contained in section 8(1)(d), but it gives little indication of the precise nature of the instrument concerned. Before the trial no particulars of the precise nature of the alleged device were asked for or given; and at the trial prosecuting counsel does dot seem to have dealt specifically with this matter either in his opening address to the jury or in the course of the considerable amount of technical evidence which was adduced. It was apparently the trial judge who raised the matter for the first time during final submissions at the close of the prosecution's case. He said to the prosecuting counsel "If I were to say to you 'what is the instrument?' what would you reply?" To which prosecuting counsel replied: "The answer I would have to give would be the user segment."

In the course of his summing up to the jury the trial judge referred to the nature of the false instrument alleged to have been made in two main passages. In the first passage he said:

'Now I have already told you that 'instrument' may be any disc, tape, sound track or other device on or in which information is recorded or stored by mechanical, electronic or other means. All I will say at this stage as to that, but I will return to it, is that it is for you to say on the evidence you have heard whether it be the position that a false instrument or false instruments were here made by one or other or both defendants as the prosecution say".

In a later passage he said:

'Now during the log-on process, these electronic impulses have to be held, albeit for a very short time indeed, in what I believe is called the input buffer which is in the user segment and checked for veracity, if you like, by the computer with the information it is already holding. It wants to see that everything is right, that everything matches, so it looks into itself to see that everything is right.

Now this checking process happens both with the customer's identity number and with his password. No doubt it all takes a fraction of a second. So there is made by the caller, say the prosecution, an instrument. By this process, they say there has been created a device on or in which information is stored by electronic means.

Now it is argued on behalf of the defendants such a construction is wholly artificial and unwarranted. Electronic impulses, they say, cannot be a device on or in which information is stored. That is their approach. Say the prosecution, that approach is too narrow an approach.

Now I do not – I would be going too far if I did – say what is my view of the matter, nor do I direct you that in law you must find one way or the other on that. It is an issue of fact. As an issue of fact, it is for you..."

The matter having been left to the jury in that way, they brought in a verdict of guilty against both the defendants on all the counts on which each of them was respectively charged.

On the defendants' appeal to the Court of Appeal (Criminal Division) it was contended for them, on various grounds, that what they had been proved to have done could not, as a matter of law, amount to the offence of forgery under the Act of 1981. The Court, which had been referred during the course of the argument to the Law Commission (Law Com. No. 55) Criminal Law Report on Forgery and Counterfeit Currency of 1973, accepted most of the contention put forward for the defendants. Lord Lane C.J., giving the judgement of the court said [1988] Q.B. 1116,1124:

'In our judgement the user segment in the instant case does not carry the necessary two types of message to bring it within the ambit of forgery at all. Moreover, neither the report nor the Act, so it seems to us, seeks to deal with information that is held for a moment whilst automatic checking takes place and is then expunged. That process is not one to which the words 'recorded or stored' can properly be applied, suggesting as they do a degree of continuance.

There is a further difficulty. The prosecution had to prove that the appellants intended that someone should accept as genuine the false instrument which they had made. The suggestion here is that it was a machine (under section 10(3)) which the appellants had intended to induce to respond to the false instrument. But the machine (ie the user segment) which was intended, so it was said, to be induced seems to be the very thing which was said to be the false instrument (ie the user segment) which was inducing the belief. If that is a correct analysis, the prosecution case is reduced to an absurdity.

We have accordingly come to the conclusion that the language of the Act was not intended to apply to the situation which was shown to exist in this case. The submissions at the close of the prosecution case should have succeeded.

It is a conclusion which we reach without regret. The Procrustean attempt to force these facts into the language of an Act not designed to fit them produced grave difficulties for both judge and jury which we would not wish to see repeated.

The appellants' conduct amounted in essence, as already stated, to dishonestly gaining access to the relevant Prestel data bank by a trick. That is not a criminal offence. If it is though desirable to make it so, that is a matter for the legislature rather than the courts. We express no view on the matter.

Our decision on the aspect of this case makes it unnecessary to determine the other issues raised by the appellants, in particular the submission that they should be found not guilty of forgery when there was no evidence that either of them had any inkling that what they were doing might amount to a contravention of the Act."

The reference in the first paragraph of the passage set out above to the user segment not carrying the two types of message necessary to bring it within the ambit of forgery is a reference to paragraph 22 of the Law Commission Report mentioned earlier. In that paragraph it was pointed out that, in the straightforward case of forgery, a document contains two messages of two distinct kinds. First a message about the document itself (eg that it is a cheque), and secondly, a message to be found in the words of the document that it is to be accepted and acted upon (eg that a banker is to pay x); and that it is only documents which contain both types of message that require protection by the law of forgery.

My Lords, the point of law of general public importance certified by the Court of Appeal (Criminal Division) as being involved in its decision to allow the appeals were these:

1. Whether on a true construction of sections 1, 8, 9 and 10 of the Forgery and Counterfeiting Act 1981, a false instrument is made in the following circumstances -

(a) a person keys into part of a computer (the user segment) a customer identification number and password of another, without the authority of that other,

(b) with the intention of causing the same computer to allow unauthorised access to its data base , and

(c) the user segment, upon receiving such information (in the form of electronic impulses), stores or records it for a very brief period whilst it checks it against similar information held in the user file of the data base of the same computer.

2. Whether, in order to constitute a false instrument within the meaning of the said Act, an instrument must contain -

(a) a message about the instrument itself, and

(b) a message to be found in the words of the instrument that it is to be accepted and acted upon.

3. Whether, in order for a person to be found guilty of forgery within the meaning of the said Act, he must be proved to have been aware of the relevant facts which constitute the making of a false instrument.

4. Whether the offence is made out if the 'somebody' whom the appellants allegedly intended should accept the false instrument as genuine (in this case - under section 10(3) – a machine) is the same machine as that which was said to be the false instrument, namely the user segment."

Point 1 comprises, potentially at least, two questions. The first question, which has to be answered in any case, is whether, in the circumstances specified in paragraphs (a) (b) (c) an instrument as defined in section 8(1) of the Act is made at all. The second question , which only arises if an affirmative answer is given to the first, is whether the instrument so made is a false instrument as defined in 9(1).

The case for the Crown on the first question was this. The relevant instrument was the control area of the user segment of the relevant Prestel computer whilst it had recorded and/or stored within it the electronic impulses purporting to be a CIN and a password. That control area of the user segment consisted of semi-conductor chips and/or magnetic cores, either or both of which are devices "on or in which information is recorded or stored by ... electronic means" within the meaning of section 8(1)(d) of the Act. Such an instrument was made by each respondent when he keyed into the control area of the user segment through a telephone line the electronic impulses which constituted the CIN and the password.

The case for the respondents on the first question was this. The recording and storage of information referred to in section 8(1)(d) are processes of a lasting and continuous nature. The process relied on by the Crown involved no more than the CIN and the password being held momentarily in the control area of the user segment while the checking of them was carried out, and then being totally and irretrievably expunged. The process did not, therefore, amount to the recording or storage of the CIN and the password within the meaning of section 8(1)(d).

My Lords, section 8(1)(d) contemplates that information may be recorded or stored by electronic means on or in (i) a disk, (ii) a tape, (iii) a sound track (presumably of a film) and (iv) devices other than those three having a similar capacity. The words "recorded" and "stored" are words in common use which should be given their ordinary and natural meaning. In my opinion both words in their ordinary and natural meaning connote the preservation of the thing which is the subject matter of them for an appreciable time with the object of subsequent retrieval or recovery. Further, in relation to information recorded or stored on, or in a disc, tape or sound track, that is the meaning of the two expressions which appears to me to be clearly intended. For both these reasons I have reached the conclusion that the respondents' case on the first question is right and that the Crown's case on it is wrong. Moreover I share the view of the Court of Appeal (Criminal Division), as expressed by Lord Lane C.J., that there is no reason to regret the failure of what he aptly described as the Procrustean attempt to force the facts of the present case into the language of an Act not designed to fit them.

On the footing that the respondents' acts did not amount to the making of any instruments as defined in section 8(1) at all, point 1 of the certified points of law must be answered in the negative; and it is unnecessary to consider whether, if they had done so, the instruments so made would have been false instruments as defined in section 9(1).

Once point 1 of the certified points has been answered in the negative, points 2, 3 and 4 become academic. That being so, it is unnecessary to give answers to them either.

I would dismiss the appeal. ”

Hampstead House · Town Centre · Basingstoke · Hampshire RG21 1LG
Tel. (0256) 840646

Creative Data Systems Ltd.

SOFTWARE DEVELOPMENT TOOLS
FOR MICROPROCESSOR TARGETS

Compilers
Mod-2 68020
Mod-2 68000
Mod-2 6801
Mod-2 6301
Mod-2 6809
Mod-2 68HC11
Mod-2 32000
C-68020/881
C-68000/08/10
C-32000
C-80386
C-8086/186
C-Z80/64180
C-8051
C-6502
C-6801

Compilers
C-6301
C-68HC11
C-6809
P-68000
P-68020
P-8086/186
P-80386
PLM-8086
PLM-8085
PLM-Z80
PLM-8051
PLM-64180

Assemblers
80386
8086/186
8096

Assemblers
8085
8051
8048
32000
68020/881
68000/08/10
6800/01
6805
6809
68HC11
6301
6305
Z-80/64180
6502
9900/99000

Simulators
68020
68000/10

Simulators
6809
68HC11
6801
6301
6805
8086/186
8085
8051
Z-80

Real-Time Executives
8086
8096
8051
Z-80
68000
68020
68HC11

Floating Point Libraries
80386
8086
8096
8051
8085
Z-80
68000
6809
68HC11
8087 Emulation

Hi-Level Debuggers
8086/186/187
80386
68000
68020

CIRCLE NO 25

The Code Page

The Code Page is a regular series for low-level software hackers. We aim to provide snippets of code or information which give a greater insight into the workings of a particular machine or piece of software. This month, Robert Schifreen explains how to ensure that resident programs get loaded only once.

If you're writing a memory-resident program for MS-DOS, it's important to ensure that the user can't load the software into memory more than once. To do this, the program must start by checking whether it is already resident and, if it is, refuse to install itself again.

There are two reasons why programs should include such a check in their installation routine. Firstly, it makes the program (and the programmer) look more professional from the user's point of view. Secondly, it will help to preserve RAM. Of course, there may be times when the user deliberately wants to load another copy of a program (for example, if something else has stolen an interrupt vector), in which case you can simply provide a command line switch in your program that bypasses the initial checks.

Official Lines

There is no support in MS-DOS for detecting the presence of a resident program, and there is no recommended Microsoft way of doing it, either. Well, that's not entirely true. There is a recommended way, but few programs use it. The method revolves around Int 2Fh. Resident programs are supposed to add a handler to this interrupt that allows them to announce their presence to the world, and to communicate with other TSRs. This interrupt is documented in few places (the MS-DOS Encyclopaedia being one), and not many programs apart from PRINT.COM use it.

Over the years, programmers have come up with a number of other methods and the purpose of this article is to outline each of them. Each method has its advantages, and some are easier than others to implement. Once you know the alternatives, you can decide for yourself which to use.

Using Vacant Vectors

Only the very earliest IBM PCs had a cassette port. Therefore, few PCs normally use Int 15h, which was designed to provide BIOS support for cassette tape I/O. (Be aware that some software, notably SideKick, hangs off this interrupt, though).

If the vector is unused on your machine, or the machine on which your software is to be run, simply place a value in the Int 15H vector. Then, when your resident program installs itself, have it check that the contents of vector 15h are zero.

Adding a handler

As an extension of the above, you can add a handler for Int 15h. For example, add a handler that will do something special when Int 15h is called with AH=99 and BL=50. That way, the resident program can make the relevant function call and check for the desired results.

What to do?

There are a number of things that your interrupt handler can do to signal its presence. The simplest is to make it set a register to zero. Before you make the interrupt call, set the register to, say, 5, and see if it becomes zero after the call.

You can also make use of the ICA. That's the Inter-Application Communication Area, which is a block of 0Fh bytes starting at 0040:00F0h. DOS doesn't use this area, and few applications apart from IBM's badged ones (like Display Write) do, either. Write an interrupt handler which, when called, increments a specific byte in the ICA. This is easy to check for, and hard for other programs to interfere with.

If you decide to hook into an already-used interrupt rather than interrupt 15h, use one that is not called frequently by DOS. Don't add an extra function to Int 21h, for example, as your handler (or at least the part that checks for a specific function) will be called so often as to slow down the machine. Don't touch disk interrupts either, as you don't really want to do anything that could affect timing loops. The same goes for printer and RS-232 handlers. It's better to add something to an existing function. For example, add a routine so that whenever DOS uses the get-date function (part of Int 1Ah), something else happens, too.

Signatures

If you want to keep things simple, you can avoid writing interrupt handlers by just using a signature. This involves embedding a known string within the code of your program and then searching through memory for it before installation. The easiest way is to put a signature in the ICA, but it's possible that other software may conflict with that, or use the ICA for its own purposes.

There is a known problem that can occur with the use of signatures. It's possible that a signature might appear in places other than the one it's supposed to. For example, part of your program might still be in a disk buffer somewhere, and the checking routine may mistake the buffer's contents for an installed copy of the resident program. The same thing will happen if the user keeps a copy of the software on a RAM disk – the signature will appear in memory but the software will not be installed.

There are two solutions. Firstly, most resident programs have a far jump as the first instruction, as they usually have the installation code at the end. Put the signature immediately after the jump, then your checking routine knows where to look for it. Of course, you'll have to check through the entire chain of handlers attached to an interrupt, and not just to the one whose address is in the vector.

The second solution is to use a self-modifying signature. The program should hold a corrupted version of the signature. For example, transpose a couple of the letters. Then, when the program loads, the first thing it does is to alter its own code (by reference to the value in CS) to return the signature to its real value. This way, only the signature in the program will be the real one, and all others that happen to be in buffers or RAM disks will fail the test.

Listing resident programs

While we're on the subject, you may be interested to know of 2 utilities that will produce a list of all resident programs currently installed, including their names, starting addresses and even the interrupts which they hook into. Both are available for downloading from bulletin boards, including Cix. One is called SMAP, and the other is SNOOP. SNOOP does more than SMAP, in that it will also show you device drivers, hardware configuration, memory control blocks and more. It comes with MASM source code, too. Note that you'll need DOS 3.0 or above to use such programs, as versions prior to this do not keep a record of a resident program's name once it has been installed. If you're not a member of Cix, call 01 390 8446 for details.

Contributions for the Code Page are welcomed. Please send articles or suggestions to the Editor. See the Letters page for details on how to contact us.

UNIX Supplement: Contents

The UNIX Supplement will reemerge in future issues of .EXE Magazine. This issue was edited by Mark Adams of Text 100 Ltd, and a member of .EXE's editorial advisory board.

ISSN 0953-9735

Editorial

UNIX At Another In A String Of Crossroads

In early 1982 I was in the privileged position of helping to organise a conference attended by Bill Gates from Microsoft. He was in Europe with Paul Allen, Microsoft's co-founder, to launch MS-DOS and to promote Xenix. We were accompanied by Adrian King, then of Logica and now VP of Systems Software at Microsoft in the US. During the day, it emerged that somebody big, I think it was Tandy, was to license Xenix. The import of this was phenomenal. At about 11 pm, I phoned Bill's room to clarify some points and we ended up discussing it together over a beer. At that point, I knew that Xenix was a very, very major product.

In the six years since then, similar discussions have occurred across the world by people infinitely more important than me almost daily. The elite groups who have huddled over fireplaces and talked of AT&T's new initiatives, Intel's commitment to UNIX (is it Xenix or is it UNIX?), which will win, and so on, must now number billions. Talking about the day UNIX will make it is now an occupational hazard, not a privileged insight.

So now I try and refrain from getting excited about UNIX. In market terms, it's always been second rate and the market doesn't look to be changing very much. But that's not what the manufacturers think. At this year's European UNIX User Show, they're as fired up as blazes.

The excitement centres on the clarity of the song now on the hymnsheet. UNIX, at long last has a focus. AT&T is pulling Berkeley, Xenix and SVID together. User interfaces are set to be sorted out, and many manufacturers have made vociferous and strategic commitments to UNIX.

In this UNIX supplement we've pulled together some of the key issues as identified by us at the .EXE offices. Yet, we gave an open brief to companies such as ICL, Unify and NCR to tell us about UNIX in the 1990s. They came back with opinions which covered almost identical topics – portability, ABI and SPARC. Perhaps the most significant observation is that UNIX people are starting to say that soon, there will be a range of UNIX software equivalent to that available on the PC.

This is important, not because it will or will not happen, but because manufacturers think it is possible. Simply because they've got the whiff of success in their nostrils, they are starting new development projects, announcing new products and strategic alliances with more frequency than ever. With everyone flapping around so fast, some of it's bound to rub off on users.

But lest we get too excited, let's remind ourselves that there has rarely been so much acrimony in the industry as there is today between AT&T/Sun and the non-SPARC licensees like HP, DEC and Apollo. That could lead to further muddy waters. And there's another fly in the ointment. It's a big fat juicy one with bulgy eyes and it's called OS/2. For every UNIX initiative, there is already an OS/2 initiative – network standards, user interface standards, binary compatibility, you name it. And though vendors will not readily admit it, the important trendsetters are choosing between OS/2 and UNIX as we speak.

For fear of repeating ourselves, let's not say that this year UNIX is going to make it, for things are genuinely as unclear as ever they we're. It's just that now the unclarity has got an unusually aggressive and competitive air to it. And we all like a good scrap don't we?

With **XOREN IPL-11** Software Data Communications is Easy.

- Interlinks any quantity or combination of PCs, UNIX/XENIX based systems, VAXes, PDP-11s and their micro equivalents.
- Allows networks to be controlled either automatically through command files or directly by users.
- Transfers data bi-directionally and reliably with full CRC error-checking.
- Connects computers asynchronously over telephone lines, direct lines and networks.
- Incorporates terminal emulation and terminal route-through capabilities.

Applications IPL-11 covers the spectrum of data communications requirements, from large **Wide Area Networks:** connecting hundreds of geographically dispersed PCs and one or more central hosts, with either the hosts or the PCs automatically controlling all telephone dialling and data transfer; to small **Local Networks:** linking as few as two computers to transfer data for back-up and storage purposes.

Method of Use An IPL-11 package is installed on each computer or PC, appropriate to its operating system. Any number of computers can be connected, and any user can control a link to one or more of the other computers. Data can be transferred automatically by using command files or directly by a user's commands from a keyboard.

Support IPL-11 is a fully supported British product developed by Xoren Computing. It is supported in the UK by its development team who are therefore well placed to ensure the software is installed correctly and that it meets users' current and future requirements.

Pedigree IPL-11 Communications Software products have been available for VAX and PDP-11 users since 1980. Now new compatible IPL-11 versions bring the power and benefits of this product range to PC, UNIX/XENIX and MicroVAX users. IPL-11 is currently used in over 20 countries by over 400 organisations on thousands of computers.

Price IPL-11 is a particularly low-cost method of achieving communications and networking solutions. Since standard asynchronous ports are used no expensive hardware is required. Volume prices range from £90.00 per PC to £495.00 per VAX with a 30 day acceptance period.

XOREN COMPUTING LTD
28 Maddox Street
London W1R 9PF
Telephone 01-629 5932
Telex 263663 XOREN G

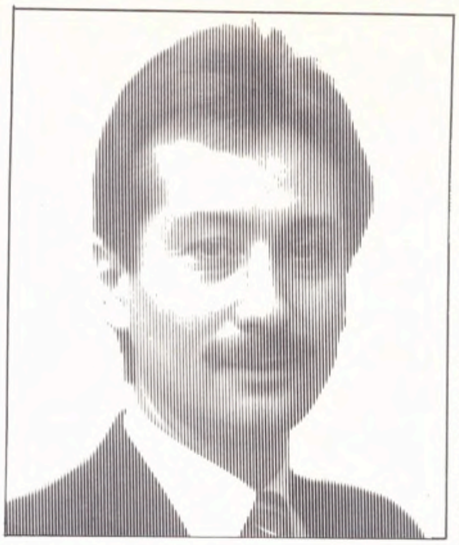
Please send me information on IPL-11:

name ______________________
company ______________________
address ______________________
____
____
telephone ______________________

☐ Telephone me to discuss IPL-11

Return to: Xoren Computing, 28 Maddox Street, London W1R 9PF, England.

VAX, MicroVAX, and PDP-11 are trademarks of Digital Equipment Corp.; UNIX is a trademark of A T & T Bell Laboratories; XENIX is a trademark of Microsoft Corp.; PC is a trademark of International Business Machines.



The Post-1990, UNIX World Will Not Be Dull

Steve Cooper, NCR UK's UNIX Product Manager believes transaction processing, fault tolerance and secure UNIX will ensure that UNIX is still the centre of debate after 1990, and hints at strategic alliances between vendors.

Sitting between centralised mainframes and PCs strewn across every department, few would dispute the pivotal role of distributed systems in the applications arena of the early 90's. Yet apprehension still centres around the massive investment in hardware, software and manpower that will be needed to make those systems a reality. What end-users are demanding today are solutions portfolios that can guarantee them continuity over greatly extended life-cycles. Yet which remain sufficiently open to track advances in technology.

The Key Standards

Industry standards are a key element in this mix. The era when one manufacturer could impose his own standards with impunity on a market has passed. NCR has nailed its colours to the mast with UNIX, and is currently in the process of delivering an implementation of AT&T's System V, Version 3 across the Tower line. In due course to be followed by the IEEE POSIX standards, phased over a two to five year timescale as these standards evolve.

Though OSI conformance will become more and more important as the turn of the decade approaches, any vendor ignores at his peril established de facto standards such as IBM's SNA. The same level choice has to exist in the application interface area proper: SNA sharing the ground with X-Open common applications environment, and AT&T's System V's interface definition.

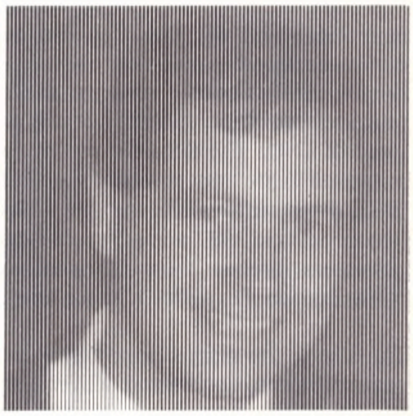
And so the list goes on, through to embracing the MIT X-Windows user interface, AT&T's RFS and Sun Microsystems' NFS with IBM's SQL alongside – this last being under investigation as to the possibilities of inclusion into NCR's UNIX kernal. All are designed to provide the end user with a series of options, in what could virtually be termed the "Lego Brick" approach to building systems.

Strategic Alliances

Strategies such as these have provoked the industry for its part to draw together – consciously or unconsciously – companies with complimentary strengths. The most obvious partners for the PC sector being IBM, Microsoft and Intel. Equally, it is not outside the bounds of probability to see equivalent alliances emerge elsewhere. Say between Motorola, NCR and Unisoft.

Will the customer really benefit though? The answer has to be 'Yes!', if only because the standards movement now has enough momentum to ride roughshod over any group wanting to stand in its way. But what we can expect is an increase of functionality through improved liaison between interested parties. RISC is but one aspect – NCR in particular is evaluating both Sun's SPARC and the Motorola 88000 for future products.

The post-1990 standards-ruled world will not be a dull one. Certainly in NCR's view, added value in the form of transaction processing capability, fault tolerant and secure Unixes, should see to that.



A Commitment to the Open Processor Architecture

ICL is one of the first licensees for Sun's SPARC chip. Peter Cunningham, ICL's Business Strategy Manager, explains the strategic reasons behind the move and its implications on ICL's departmental product range into the 1990s.

Strategic Products

When ICL became the first European IS company to commit the emergent technology from AT&T and Sun, it was based on the benefits that computer users worldwide would gain from a standard computer platform for the 1990s. Furthermore, ICL is becoming actively involved with AT&T and Sun in taking forward AT&T's UNIX Systems V Operating System and SUN's SPARC RISC processor technologies. Together, ICL believes these are poised to become a major high performance platform for open computer systems design for the 1990s.

ICL intends to incorporate SPARC technology into high-end UNIX computers. It will do this through an evolutionary development of its existing UNIX-based open systems machines. This underlines ICL's continuing support for open systems computing based on UNIX.

AT&T and Sun are the acknowledged front runners in the movement to establish an open processor architecture for building computer systems. ICL's resource investment in this initiative will help bring open systems sooner to our users.

Portability

X/Open provides, through its CAE, industry standards covering data management, languages, communication and a number of other areas. This allows source level portability between different computer architectures. SPARC RISC technology, on the other hand, develops an application binary interface (ABI), which specifies the operating and compiler interfaces. When implemented on a particular processor family, this allows binary compatibility. This really means the ability to run applications interchangeably on different vendor's hardware with the same processor family using the same operating system.

Such compatibility between computers in networks, similar to the compatibility which exists in the PC market, will provide more open competition, delivering important user benefits, including long term protection of applications investment and greater value from both hardware and software.

It also represents a great opportunity for the European IT industry. Semiconductor houses can openly licence SPARC processors. Systems integrators, can develop platforms and applications based upon the single large market that exists for a standard architecture.

ICL's commitment to open systems is also reflected in its involvement with such international bodies as the X/Open group, which is establishing a Common Applications Environment (CAE), and with Open Systems Interconnection standards bodies who are deriving a common interworking environment. ICL thinks that the development of this new open processor technology is complimentary to, not competitive with its X/Open work.

Migrating to SPARC

It is our intention to ensure that the ICL systems which will use SPARC technology, will be compliant with the standards contained in the X/Open Portability Guide in order to ensure CAE compliance.

ICL already offers UNIX on the DRS 300 and CLAN, developed in accordance with the X/Open CAE standards. This means that ICL software developed for any DRS 300 or CLAN will be portable across the systems range, as well as to all other systems developed in line with the CAE standards. Later developments of CLAN will integrate SPARC RISC technology and will be compatible with the current versions of CLAN.

Naturally, a new technology such as this cannot be introduced overnight. ICL is developing this new architecture for a range of products for the 1990s, and the first release of the product utilising the SPARC RISC technology is confidently expected in 1990.

Applications Need More than X/Open

Paul Beard of Unify sees X/Open and POSIX addressing just one software development issue. He believes more is required.

The UNIX world has come a long way since the heady days of the early 1980s with the attendant wild predictions that the world would see only two multi-user operating systems by the end of the decade; IBM proprietary offerings and UNIX. Time has brought a more reasonable perspective but an equally compelling endorsement of UNIX. The debate over the viability of UNIX has been answered in the most emphatic way – proven sales. Total sales of UNIX systems in the UK by the end of 1986 amounted to 40,500. US licence sales in 1987 numbered 180,000. By 1990 even the most pessimistic forecasts show US annual sales at double this number whilst Europe is expected to triple during the same period.

The size of the market opportunity has driven every hardware manufacturer to declare and deliver a family of machines running the UNIX operating system. Most notable recent endorsements being IBM's unveiling of the AIX UNIX strategy from PS/2 machines to the 3090 and 9370 families. Equally significant are Apple's A/UX announcements and Digital's current heavy development investment in its UNIX offering – Ultrix.

Unix is the environment of choice for multi-user systems from £10,000 to £1 million because of the price-performance of the hardware on offer and the protection of users' application software investment provided by the portability of UNIX. Perhaps the single most compelling stimulus to the commercial acceptance of UNIX, is the quest for hardware vendor independence. This is, in part, the motivation behind the standardisation initiatives which will be the significant trend in the immediate future development of commercial UNIX.

The joint activities of the IEEE POSIX Committee and the European X/Open group reflect the overwhelming pressure towards a common open standard environment for application development across multi-vendor UNIX systems.

Such standardisation will do much to protect commercial users' software investment but addresses only one aspect of the cost of commercial application development. The more costly issue, that of on-going software maintenance, has led to rapid growth in sales of relational databases and fourth generation application development tools. Themselves the subject of much current standardisation, they represent not only complete portability and therefore protection, but also bring benefits of development productivity and reduced maintenance costs on the resultant applications.

The future of commercial multi-user applications lies with standard UNIX and these independent fourth generation development environments.

Paul Beard is Vice President for European Operations at Unify Corporation.

PC's To Acquire UNIX Windowing Capability

Sarah Keefe looks at the preferred UNIX user interface, X Windows, and sees if the PC market has got anything to learn from it.

The recent emergence of the X Window System and its acceptance as the *de facto* standard for controlling windows and graphics on workstations, PC's and intelligent terminals, has resulted in it receiving significant attention from a large number of hardware manufacturers and software authors, especially workstation vendors and UNIX afficionados. But will the X Window System make an impact in the PC arena where proprietary windowing systems have already been accepted by users? If so, how and what are the implications for the average PC user? I'd like to start with a short overview of X Windows, or X.

Inside X

X is the outcome of a project initiated by the Massachussets Institute of Technology to investigate the educational application of a large network of workstations in a multi-vendor environment. Project Athena, as it became known, had one main aim and that was the portability of applications and system software, including the user interface. In meeting the design requirements of Project Athena, X was developed with the following features:

It operates on any bitmapped screen architecture. ie. it is not just character based.
The system is network transparent.
Multiple applications operate concurrently on one screen.
It supports multiple user interface styles.
It supports overlapping and obscured windows.
Applications can use many windows.
It provides high quality text, graphics and imaging.
The system is extensible.

X is not a single program and is not written in any particular language. It does not specify any operating system environment. Rather, X is a series of definitions, requirements and protocols which can be implemented in a wide variety of ways. X sits between the operating system and the application program and defines how application programs can communicate with the user in a graphical and visual way.

X Architecture

The X architecture separates windowing into two distinct parts: the application program or 'client' and the display station (which includes display hardware, mouse and keyboard) or 'server'. The client and server may reside on the same system or they may communicate across a network, sometimes over a great distance. For example, a powerful supercomputer running a client application might display graphics on an intelligent terminal running an X server on the user's desk.

Several client programs may communicate with single server and each client application can control one or more windows on the screen. It is the job of the server program to ensure that applications do not interfere with each other. Users of X programs over a network such as Ethernet benefit from the sharing of valuable resources located at a considerable distance from the user.

One of the main reasons for X's acceptance is that it is hardware independent. X achieves this by the separation of windowing into the two distinct parts and by providing the communications layer for different types of applications to exist in a heterogeneous network. When X is ported to new hardware, only the X server is implemented on the display hardware in question. Once an application has been written using the X11 programmer's library interface (XLIB), it does not need to be modified to accommodate new display hardware – porting an application is usually a matter of recompiling.

Implicit in this architecture is the requirement that two processes run concurrently, so that hardware must in some sense be multitasking. Partly as a result of this, most implementations have been on UNIX systems. The fact that most of the hardware involved in Project Athena ran UNIX is also responsible. Nevertheless, the complete X system has also been implemented under Digital's VMS and several UNIX work-alikes such as Ultrix and Helios.

X and the PC market

There is currently a burgeoning market for 80386 based PC's running UNIX with the result that X has been implemented under various 'species' of UNIX for this sector of the PC market. X servers running under UNIX together with client applications (exploiting XLIB or simply running in VT100 emulation windows) are emerging. A major benefit of using X on a local UNIX based PC is the possibility of interworking with other host systems on the network without restricting local functions. It is evident that a growing number of PC users are no longer satisfied with the capabilities of the genteel PC and are now looking to powerful micros capable of multitasking, handling graphics and networking.

The normal rationale for windowing is to separate the output from several concurrent tasks on a users screen. On most of the existing PC population, the operating system is MS-DOS which effectively prohibits true multitasking. However, since X is split into two parts, it is quite conceivable, and indeed possible, to run an X server only under MS-DOS. This then communicates with the clients program running on a UNIX host over a network. Output from the tasks running remotely is interleaved by the server. In this way, the user of a PC compatible can connect into a host running X clients and operate them with full X functionality at a fraction of the cost of using a complete workstation.

Various X servers have been produced for graphics-equipped computers such as the IBM PC, one such product is PC Xsight from Locus Computing. It allows an Intel 8086-based PC compatible to connect into a host running C clients and operate as a graphics terminal. This maximises utilisation of systems resources by converting PCs into multitasking workstations.

Despite having the unfortunate nickname "half an operating system", OS/2 on many desktop Intel based PC's will provide multitasking with Presentation Manager as the user interface. It is clear, however, that users wishing to interwork with non-Intel systems will use X to drive the user interface.

Some people intend to dispense with the Presentation Manager interface and run an X server on the OS/2 system. Others are developing a bridge software to allow an X based application on a local or remote machine to "see" a standard X server interface, which is in fact an X emulation running under Presentation Manager.

All these methods will allow a PC user to access networked applications in a seamless way, just as Project Athena envisaged. The only issue in making a choice will be the simultaneous local computing requirements of the user.

Why is a standard important?

Software developers always try to widen the potential market for their software by selecting popular operating systems and hardware platforms, and sticking to de-facto standards wherever possible. In the PC market, MS-DOS on the Intel 8086 chip has become such a standard environment for software development and delivery is beginning to offer a stable and consistent base across a wide range of hardware.

Most UNIX workstations tend to provide their own proprietary windowing and graphics interface, one such interface is NeWS or Network Extensible Window System from Sun. Like X, NeWS is a client/server based system designed to allow applications to be split up from the display and is based on Sun's implementation of the PostScript page description language. Although Sun feels that NeWS is superior to X and has pushed it as a contender for a standard window environment, NeWS has not been accepted with the same degree of enthusiasm as X. An important argument in X's favour is that it is publicly available software that carries no licensing restrictions, in other words no vendor has "a lock on it". Perhaps UNIX vendors over the years have learnt to be wary of vendor designated standards which they feel would become vendor strangleholds.

The lack of a standard windowing and graphics interface on UNIX workstations has led to a fragmentation of the UNIX market, making the transfer of software from one system to another difficult and expensive.

X has started to sweep away that fragmentation, leading to a plethora of opportunities for software developers and a reduction in the level of confusion amongst users. The idea of a Common Application Environment across all hardware is attractive to software developers. Now that X is available across a range of operating systems and hardware, that idea is one step nearer to reality. X/OPEN is adding X to its portability guide providing that MS-DOS and OS/2 implementations of X prove efficient and reliable, it will be possible to produce portable applications which include a portable user interface.

For the owner of a humble standalone 8086 based PC, X means little, but for anyone with access to networked multi-tasking processor power, X means that their PC is no longer becoming rapidly obsolete. Instead it can be their window into the next generation of application software.

Sarah Keefe is Marketing Director at IXI, a porting and X Windows consulting specialist. She can be contacted on 044223 68749.

X Marks the Spot

Which route should a hardware manufacturer or applications developer take when deciding upon a windowing system?

For the hardware manufacturer, the answer must be X11, at least in the short term. The X11 standard has been fixed for the next three years by a group of companies known as the X consortium, encompassing 85% of the world computer market. It has thus become a de facto standard, although it is technically inferior to Sun's NeWS.

Manufacturers' investment in X will be protected for other reasons. Sun has announced its intention to produce a NeWS/X11 merged window system. This will be open – the source code will be available to other manufacturers. Manufacturers who wish to allow their customers to take advantage of the extensible features of NeWS will thus be able to upgrade their systems whilst protecting other customers' software development investment using the X system. This route offers the best of both worlds.

AT&T's Open Look graphical user interface is a system designed to give UNIX applications a common graphical look and feel. The application programming interface will be available for both X11 and NeWS. Again, investment in X will therefore be safe.

The situation is not so clear cut for application developers. A windowing system is made up of three layers of software, the low-level imaging model (provided by the X interface Xlib), the toolkit layer (providing standard user interface components such as scroll bars) and possibly a higher layer concerned with user interface management (UIM). Applications developers use either one of the top two layers. There are currently at least three toolkits available with the X window system including Xtk from MIT. All are available on the standard X distribution tape.

There is, as yet, no clear winner emerging from these toolkits. AT&T's Open Look is based upon the lowest level of Xtk, which will lend some weight to this toolkit. It is too early, however, to predict market adoption of any particular toolkit or UIMS. All contenders have good and bad features. The Andrew toolkit and UIMS such as Apollo's Open Dialog deserve investigation.

To summarise, for a hardware manufacturer, the X window system is in the short term the most valid choice. X is here now, is the de facto standard and does not preclude upgrading to the merged X11/NeWS system or Open Look in the future. Applications developers have more of a problem in choosing a toolkit upon which to base development. Technical merits and application requirements should undoubtedly be taken into account in this decision, although AT&T's decision to base the X part of the Open Look applications programming interface Xtk lends weight to this toolkit.

Mark Collins-Cope, The Instruction Set.

The ABC of ABI

Derek Williams, Technical Director of UniSoft, describes the benefits and technicalities of the Applications Binary Interface.

One of the most talked about UNIX issues of 1988, the Applications Binary Interface (ABI), was not even part of the vocabulary 12 months ago. The reason for such brouhaha is that the proponents if UNIX, and these now include virtually every leading computer manufacturer in the world, see in ABI the means to bring about the same degree of UNIX applications portability as currently exists in the PC marketplace.

ABIs will not just accelerate the already rapid convergence of UNIX around System V, they will bring benefits to all parts of the market. Not only to chip makers and OEMs but also to software houses and end-users. Manufacturers will be able to bring their UNIX systems to market faster, software houses will be able to address a much wider customer base and end-users will have a much greater choice of high quality UNIX applications. With market research companies estimating, even before the advent of ABIs, that UNIX will have taken 20% of the computer market by the early 1990s, the future of UNIX now looks guaranteed.

It is ironic that the development of ABIs is neither new or technically difficult. Indeed, those who have been working closely with UniSoft will be aware that much of the work is already complete on ABIs for the Motorola 68000 and 88000 series.

After two years of preparation and circulation, the final draft, of what, until now, we have referred to as the 68000 Binary Compatibility Standard (BCS), will be available at this year's European UNIX User Show. ABIs for other chip sets are being defined by their respective manufacturers.

What is ABI?

In essence, the 68000 ABI will allow any conforming 68000 based computer to run identical applications software, without the need for re-compilation and regardless of the hardware manufacturer.

This will be achieved by laying down two conventions. Firstly, for the interfaces for the binary executable file and the operating system, and secondly, for the data interchange standards for removable media.

For the UNIX industry as a whole, ABIs should be seen as an important step in the creation of a single homogeneous market. This is necessary because the way in which UNIX was originally licensed lead to many different versions being developed. In addition, since UNIX has never been tied to any single proprietary computer manufacturer, the standardisation that might have come about through the exercise of commercial clout, has not been fully achieved.

SVID (the System V Interface Definition), announced in the mid-eighties, was AT&T's successful first step to define what UNIX was or was not, and it has played a decisive role in converging the leading variants of UNIX; BSD, Xenix and System V.

The SVID standard was brought into the public domain by the standards bureau of the IEEE (International Electronics and Electrical Engineers) when in 1985 it set up POSIX, the Portable Operating System for Computer Environments. POSIX took the SVID standard and added one or two extensions in the area of signal handling and job control.

Both SVID and POSIX attempt to cushion the applications developer from the hardware differences of target systems. These define a functional standard interface between the application program and the operating system and set down a minimum set of requirements for applications portability. SVID and POSIX are definitions and appear in the form of a book and they define UNIX at the level of source code. So two different conformant computers are able to share the same applications software – though each will need to be compiled differently on the target machine.

The Application Binary Interface, on the other hand, goes much further in its guarantee of compatibility. Thus applications software that runs on one ABI-conformant computer will run on all other ABI-conformant computers.

The key to the understanding of ABI is in the letter B for binary. Once compiled, an application program consists of machine code that runs directly in user space and calls into the kernel to provide operating systems services. The code that runs in user space is always compatible across machines using the same chip set. It is defined by the instruction set of the chip set in use. The exact interface to the operating systems is not constrained in this way. UNIX implementors can choose their own convention as long as the compilers, libraries and kernel on any one machine agree.

In an ABI, there is agreement at the binary level as to these otherwise arbitrary decisions. So with two processors that are ABI compatible, the binary code for a particular application program will be identical.

With two computers that are merely POSIX or SVID compatible, there is no agreement below the 'functional' level and so the system calls will not be in agreement, neither will any detail below this level. So the final binary codes will not correspond. Furthermore, whereas as POSIX or SVID defines what must be present at the system call level, ABI specifies not only what must be present but how it is to be executed. The what and the how apply to each of the three levels of system call interface, data alignment and loader interface.

The ABI then is a definition of what an implementation of UNIX will look like. Expert engineers meet together and agree an ABI. Technically it is not a difficult task. But in practice, the creation of the definition goes through many drafts as it has to find agreement with a number of parties: the chip manufacturer, AT&T, the UNIX porting house and most importantly, the hardware manufacturers.

Implementations

Meanwhile, hardware implementations are tending to converge because of the integration of memory management units with CPU chips. Once the ABI is finalised, a conformant implementation on UNIX can be built. This is a longer task but it still only consists of modifying the interface to an existing UNIX implementation.

To the applications software writer, the ABI is unimportant, since he writes the software to conform at the source level to SVID/POSIX without needing to be aware

"As you can see, our company had a multitude of reasons for buying the IBM 6150 microcomputer." The IBM 6150 can expand easily from few to many users and is the ideal system for your company to grow into. It allows transparent sharing of data with integrity, both locally and remotely, and as the 6150 runs IBM's AIX,™ a UNIX™ implementation, existing applications can be readily ported to the 6150. AIX will also conform to the emerging POSIX standard. Together with its wide range of proven software, this makes the IBM 6150 suitable for an equally wide range of firms: from Comet to Radio 210 and from Thistle Hotels to The Danish Bacon Company, for example. And, thanks to IBM's unique RISC architecture, the 6150 also has an outstanding price performance. Talk to your IBM Authorised Dealer; he has the application software and expertise to tailor-make a solution for your business. And that's the best reason of all for choosing the 6150. For your nearest IBM 6150 Authorised Dealer, contact Sheila Franks, National Enquiry Centre, IBM United Kingdom Limited, FREEPOST, London W4 5BR, or ring 01-578 4399 during working hours. "I think, therefore IBM."

IBM

AIX is a trademark of International Business Machines Corporation. UNIX is a trademark of A T and T Bell Laboratories.

CIRCLE NO 39

of the presence of the ABI. However to the people who market the application, the ABI is of fundamental importance. Once the source code is compiled on one ABI conformant machine, the application will run on any machine using the same ABI. Thus for the software house, support costs are drastically reduced since they no longer need to purchase and maintain different manufacturers' computers on which to compile and test their programs. One computer running each of the ABIs is sufficient for all computers using the same chip set.

More importantly, all software will run on a much wider range of machines. Software houses will find themselves able to sell into a much larger market and, simultaneously, distribution costs will fall as the market becomes unified and is able to be supported with single products. The availability of the best software to the greatest number of users will benefit the whole industry.

The effect on hardware manufacturers is more difficult to predict. But if all computers built on the same chip set run the same software, manufacturers will have to discover different ways to differentiate their products. Ultimately this will accelerate the trend towards hardware becoming a commodity product. Hardware manufacturers will move towards becoming total systems suppliers, adding value in services rather than in unique hardware.

Companies like UniSoft, too, will be effected dramatically. Most importantly they will be able to transfer more of their efforts from the porting of existing applications to innovative software development.

In the end, the advent of ABIs will release the creativity of literally thousands of UNIX programmers, from reinventing the wheel to fundamentally advancing the computer industry.

AT&T and Motorola – Hand in Glove on ABI

Motorola and AT&T jointly announced the finalisation of an agreement for specifications on ABIs for the 88000 and 68000 microprocessor families on April 22, 1988. As a result of this agreement, Motorola is guaranteed full and early access to UNIX System V release 4.0 and later releases, ensuring timely availability of the specific implementations for Motorola microprocessor architectures. Motorola has also agreed to distribute UNIX SYstem V Release 4.0 and update releases as the preferred UNIX system for its two microprocessor architecture families.

Murray Goldman, Senior VP and General Manager of Motorola's Microprocessor Products Group said, "Our full and early access to UNIX System V Release 4.0 and later versions allows our customers to develop systems with optimum performance. Coupled with our 88000 RISC family, full and early UNIX system availability should continue to position Motorola as the front-runner in computing technology."

Unisoft Corp will assist Motorola in developing the specific implementation of UNIX System V Release 4.0 and update releases for the Motorola architectures. UniSoft will also assist in the development of the associated ABI specifications. Motorola will receive from AT&T licenses to use the trademark UNIX to identify the operating system software for the Motorola architectures.

"We are in favour of ABI's because, like other standards, ABIs will let the best architecture win," said Jack Browne, Director of Marketing, Motorola High-End Microprocessor Division. "We believe that the new 88000 RISC family and the 68000 family will stand the test of competition and greatly benefit from ABIs."

AT&T will make the ABI specifications and associated implementations of UNIX System V Release 4.0 for the Motorola architectures available to the industry at large through its UNIX software licensing programme. The first UNIX SYstem V Release is expected to be available in late 1989.

RISC in Perspective

Allister Mannion, Senior Technical Consultant at TIS, has an objective ponder on the RISC debate and offers a few caveats.

RISC manufacturers are targeting their machines at users of traditional commercial CISC architectures from suppliers such as DEC and IBM. For the programmer, does RISC create or solve problems when creating business applications? Or, put another way, what difference does it make what CPU architecture the machine uses?

In the ideal world, the answer should be 'none', but things are rarely that simple. The first problem is that machines currently utilising RISC technology fall into two groups, workstations and minicomputers, both with different evolutions and target markets. The one thing they have in common is UNIX, but that's where the similarity ends.

But why is RISC itself fundamentally different in application development to CISC? The simple answer lies in the compiler technology. CPUs of the type now categorised as CISC have, in general, grown out of the need for computing power on a wide range of applications and so have a great deal in common. Thus, the latest generation of CISC microprocessors, like Intel's 80386, have a similar instruction set to that found in much older CPUs contained in machines like DEC's VAX or Data General's Eclipse range.

The designs of these CISC CPUs arose from a philosophy of providing programmers with more and more support at the instruction level. Programmers of the same generation as these minicomputers have been forced to write totally in assembler, machine level instructions, because of tight memory constraints and relatively slow CPUs contained in the target machines. As compiler technology developed, the authors of the original optimising compilers found that the support given to application programmers in the form of large instruction sets was not useful to the compiler. Compilers generating code for a limited number of high level statements were found to use only a small number of instructions, about 20 per cent, for 80 per cent of the time.

Languages and Compilers

The reason that RISC gains its staggering performance advantages over CISC is that the compiler is written first. With this in place, the optimal instruction set to take advantage of these instructions can be chosen and then be set into hardware. This instruction set need not carry the burden of unused instructions and this need not be reflected either in machine performance or price. The compiler, then, is central to this success story.

For the business user, the key to using the full potential of RISC is whether the application is written in one of a number of targeted high level languages. In contrast to the minicomputer market, in the workstation market most applications are relatively new, and so are written in languages that have already received attention by the raft of RISC manufacturers, many of which are only a few years old. Thus, for the workstation manufacturer and users, the transition is much easier.

In the commercial world, a different set of constraints applies. Application software is more mature and has appeared in numerous older languages. About 40% of applications software still exists in, and is being developed in, COBOL. Without a state-of-the-art RISC compiler, the user has no way of running existing applications software on the new machine.

So is writing a compiler for a RISC CPU any different from writing for a conventional CPU? Unfortunately, yes very. With a typical RISC CPU executing ten million plus instructions per second, it is vital that the machine's memory is fast enough to supply the CPU. Memory technology, however, is the same for CISC or RISC machines.

Memory Problems

If RISC manufacturers were to use very fast memory alone then they would immediately lose the price advantage over conventional architectures. For the RISC manufacturer entering the commercial arena, there are a number of memory-related problems to be overcome. In a commercial system, jobs are switched rapidly and this requirement enters into the strategy of memory caching. If a CPU has a large number of registers for data then each time the job is changed the program has to save them all. Because of these constraints, a RISC manufacturer can unconsciously design a very good single user workstation but a very bad multi-user computer. In the commercial, multi-user environment various strategies are used to smooth memory demand and help create a viable system that optimises the power of the RISC CPU.

Some use large numbers of CPU registers to hold temporary data, others employ elaborate caching mechanisms and depend on pipelining. Whichever is used, the compiler is responsible for making maximum use of the individual strategy. The compiler writer now has a lot more to think about in terms of optimising CPU useage.

The RISC compiler writer is now writing very much for a particular CPU, not a general class of microprocessor. The end result is that one RISC compiler is very different from another RISC compiler.

A recent benchmark of SUN's SPARC-based SUN-4 workstation demonstrated this. Neal Nelson & Associates ran its set of commercially-orientated benchmarks and showed that the SUN-3, a 4 MIPS 68020-based machine, outperformed the 20 MIPS SUN-4 in eight out of eighteen tests. This indicates that Sun did not consider, for whatever reasons, the commercial market when designing the SPARC and it's associated compiler.

What RISC demonstrates is that in applying any technology to a job, all components vital to the function for which the product is intended must be equally matched. The commercial world demands the right applications at the right price. In the case of RISC, this means the availability of the right compilers and whether they make careful consideration of the multi-user demands on the CPU.

What does a CPU need to run UNIX: The NS32532

The CPU

The architecture of the CPU needs a variety of features in order to support UNIX successfully. These include:

High-level Language Oriented Instruction Set. Portability has been a major factor in making UNIX so successful. That portability was achieved in writing as much as possible of the operating system in C. So if UNIX is to execute efficiently, it is a prerequisite that the processor's instruction set should match closely the requirements of high-level block-structured languages.

Separate User and Supervisor Modes. Any multi-user operating system must forbid individual users direct control of the fundamental I/O devices. Access to I/O must be restricted to the operating system itself – otherwise it would be impossible from preventing two users from accessing a device at the same time, resulting in chaos. For this and other reasons, the operating system must be able to carry out operations that the user cannot (special instructions which alter the state of the system, access to specific areas of memory, etc.). Some processors have been designed with multiple levels of privilege, but UNIX only requires two, referred to as user and supervisor mode. Of course, it is important to get right exactly how supervisor mode operates. for example, interrupts and operating system kernel routines should not use the same stack as user applications, so the processor should automatically change stack pointer on entry to an interrupt service or system call.

Paged virtual memory management

UNIX requires a virtual memory scheme. All recent UNIX implementations use a mechanism called 'demand paged virtual memory' where the continuous logical addresses of a process are split up into relatively small fixed-size 'pages', each of which has an independent translation to a page of physical memory. This scheme is quite complicated to implement but has significant performance advantages over older 'segmented' approaches.

It is vital that paged virtual memory be part of the defined processor architecture. Preferably, memory management instructions should be included in the instruction set. National Semiconductor introduced the NS32082 memory management unit at the same time as its first generation CPU's, the 32016 and 32032, so that there has always been a well defined memory-management architecture for the Series 32000 family. The NS32382 was introduced to support the NS32332 second generation CPU, and compatible memory-management hardware was brought onto the CPU chip with the third generation NS32532

Floating point

It is clear that many UNIX systems are bought to support technical and scientific users, whose application programs require high-performance floating point hardware. In addition, good quality graphics need fast floating point. Thus all the graphics workstations, such as those in the emerging desktop publishing market need this support too. National offers the NS32081, NS32381 and NS32580 floating-point slave processors, the last being an interface to the Weitek 3164 with a peak performance of 15 MFlops. Again it is important that the floating-point instructions should be included in the CPU's instruction set. This provides a core of floating point operations that maintain compatibility between different systems based on the same processor family.

Interrupt control

UNIX uses two interrupts for two purposes. Firstly I/O devices are handled through interrupt service routines, and secondly the whole process scheduling mechanism is driven by a single time interrupt which normally occurs fifty times per second. To implement these, a fairly sophisticated interrupt controller and a timer interrupt generator are required. In the 32000 family, these functions are included in the same part, the interrupt NS32202 interrupt control unit.

Extremely fast interrupt handling is not usually a requirement in UNIX systems. However, the overhead of a complete task switch should be kept to a reasonable level. Otherwise a systems running many tasks can get bogged down. This ought to be considered as a counter balance to the "the more registers the better" argument put forward by proponents of some RISC architectures, since all user registers must be saved on a process switch.

Compilers

Even when the CPU's instruction set is well designed to execute high-level languages there is a great deal of difference between a smart compiler and a dumb one. Smart compilers can allocate frequently used variables to registers, speed up calls to procedures, eliminate redundant code, and make a variety of other optimisations. National's Compiler Technology Project has found that our optimising compilers can make a difference of between 30% and 200% to the execution time of CPU-bound code. Remember that this benefit applies not only to user application programs, but also to the operating systems itself, since it is largely written in C.

UNIX Software

The final element of the UNIX formula is the availability of generic UNIX ports for the processor architecture, and compatible compilers and software development tools exist, porting UNIX to the system becomes a routine task. The systems designer can then concentrate on those parts of the product that distinguish it from others in the market, instead of having to struggle to implement the basic functionality.

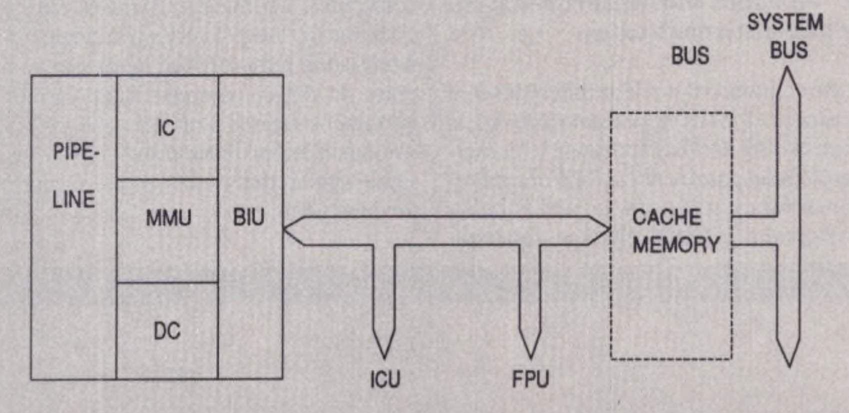


Figure 1 - The 32532 Cluster.

What does a CPU need to run UNIX: The MIPS R2000

UNIX is the unsung hero of technological development in computer architecture. Without such an accepted and portable operating systems, the chance that a new microprocessor architecture could gain widespread acceptance in the market place is very slight indeed, given the close links between proprietary hardware and system software.

UNIX has changed the traditional ground rules, however, bringing to developers, and so to users, the benefits of new and unfettered technologies. RISC design is proving to be the most notable of these, bringing new and unmatched levels of price performance.

In system architecture design, a newcomer may have a new clean slate as far as implementation and instruction set is concerned, but must take into account the nature and behaviour of the programs that will execute on this design. An architecture team which does not have a wide range of experience, not least in the UNIX kernel, is in danger of producing an elegant design that fails to perform in the real world.

In its early stages of architecture planning, MIPS Computer Systems, soon saw the necessity to obtain inputs from operating systems experts and optimising compiler specialists in addition to the hardware implementers. The result is an architecture that is now the fastest in any given technol-

GKS THE GRAPHICS STANDARD

GKS is currently the only internationally recognised standard for computer graphics. It allows portable graphical applications to be developed for use on a wide range of systems and output devices.

S-GKS: the first fully ISO certified edition of the Graphical Kernel System (GKS) to be written in C. The power and flexibility of C gives S-GKS significant performance gains over Fortran based packages.

S-GKS is currently in use on PCs, VAX and numerous Unix based workstations eg. SUN , Apollo, IBM 6150 . Level 2b GKS functionality is provided in all environments (MS-DOS, Unix, VMS, Ultrix) .

Drivers are available for many popular graphics devices : Tektronix, HP, Calcomp, Zeta, DEC, Postscript laser printers etc.

Language bindings are supplied for both C and Fortran .

Also available ...

Draft Editor: a general purpose drawing package using the features of S-GKS for versatillity and portability. User friendly, menu driven, suitable for OHP slide production, 2D plans etc.

C-GUL: allows existing applications which use Calcomp plotting libraries (and compatibles) to be linked directly to any GKS thereby offering enhanced output capabilities and support for a wider range of devices (including graphics terminals) .

Scientific Software Limited
Rose Industrial Estate, 54 Marlow Bottom, Marlow, Bucks. SL7 3ND **Tel. (0628) 890011**

CIRCLE NO 31

Computer Manuals Limited

Suppliers of books
Literature
Manuals
Software
Training Programmes

30 Lincoln Road Olton Birmingham B27 6PA England
Telephone: 021·706 6000
Telex: 334361 Compco G

CIRCLE NO 32

OS-9//88 CONFERENCE

HEATHROW PENTA HOTEL
3rd, 4th, 5th October

Following the success of the first European OS-9 Conference in 1987, Vivaway Limited, will be holding the second OS-9 Conference at the Heathrow Penta Hotel in October.

Seminars will be held on all three days from 10.00 am to 5.00 pm and will include New Product Announcements from Microware, Technical Seminars and Guest Speakers from a variety of industries.

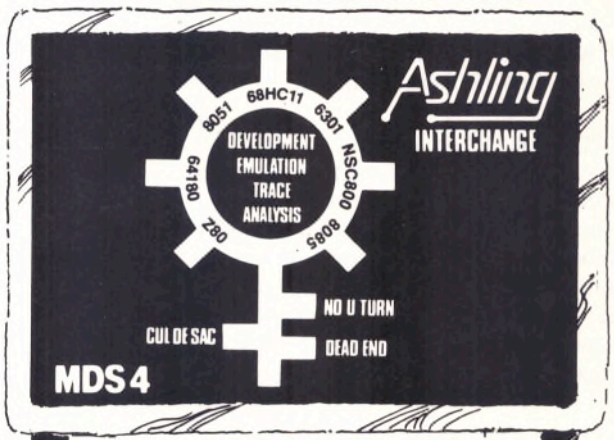
The OS-9 exhibition will be running on the 4th and 5th October. Companies exhibiting last year included Amplicon Electronics, Bicc-Vero Microsystems, Crellon Microsystems, Force Computers, H.T.E.C., Integrated Micro Products, Koral Microsystems, M.P.D., OS-9 European User Group, Plessey Microsystems, Spectra-Tek UK, Syntel Microsystems, U-Micro Computers and Windrush Microsystems.

To reserve your stand or a place at the Conference, please contact:
Rita Poulter,
Vivaway Limited,
Software House,
Burr Street, Luton,
Bedfordshire LU2 0HN.
Telephone: (0582) 405511

OS-9 is a trademark of Microware Systems Corporation

CIRCLE NO 33

Do Yourself a Good Turn



Choose the MDS Route that offers Choice

We've designed our Microprocessor Development Systems so that you can easily reconfigure them for a range of devices or for device derivatives or for device manufacturer variations.
So that you can use the devices that are exactly right for your current design projects.
And know that the same applies in the future.
Ashling Interchangeability. It lets you take the design route you want.

Ashling Microsystems Ltd., Unit 4, Switchback Centre, Gardner Road, Maidenhead, Berks SL6 7XA, UK.

Tel: (0628) 773070
Fax: (0628) 773009
Telex: 8950511 for attn. MBX 25260001

Ashling
The Development Systems Company

CIRCLE NO 34

ogy to date, across a broad range of applications.

In the multi user environment, efficient support for the UNIX kernel is key. The kernel's job is to manage the resources of the system. If the kernel implementation is slow, therefore, all users suffer. To overcome this, the architecture design must offer performance in the most frequently used areas, namely resource management (scheduling CPU and FPU, handling peripheral devices, allocating physical and virtual memory) and the procedural call mechanism, including **SYSCALL** or **TRAP**.

These translate into four principal areas of architecture and VLSI implementation – exception handling, virtual memory and caching, context switching and procedure calling. The following outlines the MIPS rationale relevant to each of these features.

Exception Handling

Exceptions – including the interrupt which is just a special case exception – occur with great frequency, and use large chunks of common code within the kernel. The MIPS implementation despatches an exception within a little over one clock cycle, through either of only two run-time exception vendors. The majority of exceptions are despatched through just one of those, presenting the cause (or vector) to the exception handler in a machine register. This is just what the UNIX kernel requires.

Other machines have a multitude of exception vendors, which take time to construct. Moreover, examination of the code vectors to show the unique logging of a unique number and then a jump to common code. This is a demonstrable illustration of an idea which, though deemed attractive by hardware designers, is ignored by UNIX because of the time penalty it introduces.

When the kernel encounters an exception very little machine state is preserved (2 bits in fact) before entering the exception code to decide what state to preserve, instead of the machine wasting time saving a vast array of registers to memory only to have them almost immediately restored on exit from what may have been a minimal exception routine.

Virtual memory and caching

In the current MIPS VLSI implementation all address references are mapped via the on-chip Translation Lookaside Buffer (TLB). The MIPS virtual memory scheme is shown in Fig 1. In user mode (**kuseg**) 20-48 Mbytes are available. On entering kernel mode, **kuseg** is still available allowing the UNIX kernel to examine the current process context without remapping. In high performance systems, the kernel is not a necessity. To avoid the kernel occupying valuable TLB entries it can be assigned **KsegO** addresses, and so it is direct mapped to physical memory. The **ksegl** is identical to **kegO** except that memory references do not update the caches. Device I/O address space is assigned to this area.

Context switching

With context switch, all state particular to the outgoing process that will be used by the incoming process must be saved. To speed this process the MIPS system implements a fast single-cycle store, with memory buffer to accept data at CPU clock rate. Data and instruction caches are physical, that is they are indexed by physical addresses so there is no need to flush them on context switch. This advantageous cache design is made possible by the speed resulting from the on chip TLB.

The alternative of virtual caches is easier to implement. However, these have to be flushed forcing the incoming process to fault in all its instruction and data, even though they may already be held there from the last time it executed. The TLB can be looked upon as a virtual cache needing to be flushed.

MIPS overcomes the potential performance problem of the TLB flushing by using a technique whereby each entry is tagged with an ID of a particular process. This ID is used during translation to check that the entry being used does indeed belong to the requesting process, thus reducing the need to flush the TLB.

Procedural Calling

Initially hardware help in the form of 'register windows' for fast procedure calls looks attractive, until the nature and depth of kernel calling is examined. From experience the kernel is erratic in its calling, often nesting procedures deeply and very quickly, only to return to the upper levels before plunging again.

This action often defeats the maximum depth of calls in hardware register windows, forcing large and frequent register saves to memory, only to restore them again as the kernel returns to the upper level. The MIPS solution is to implement the procedure call standard in software, allowing the compilers to determine the registers to save and restore at any given point.

In summary, none of the above features is a specific function of traditional RISC design. In the MIPS architecture, where the goal was for maximum system performance, the focus was on the critical paths for fast operating systems implementation, but linked to the need to get a well-balanced systems that reflects the realities of the application environment.

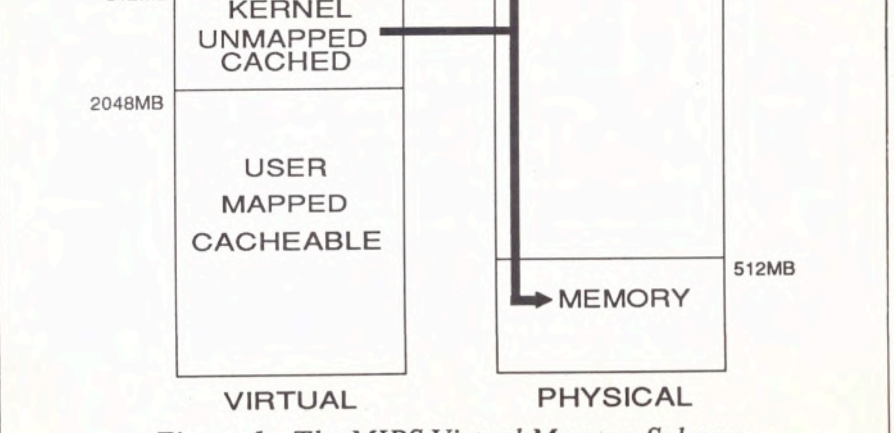


Figure 1: The MIPS Virtual Memory Scheme

semantics

Presenting our CASE

The cost of using CASE technology just fell through the floor. **£425** *Equip the whole design department and watch productivity rocket.*

* ADVANCED CASE TOOLS	Support software design using graphics based structured design methodologies
* MODULAR	Design tool + Rules tool + Data dictionary tool.
* LOW ENTRY COST	Just £425 gets you up and running with the design tool (1200 for all three)
* NO TRAINING REQUIRED	Our three tutorials and extensive on line help will get you up to speed in hours.
* OPEN ARCHITECTURE	ASCII export/import functions with all file formats provided.
* CONTINUED SUPPORT	With over 4000 installations worldwide you can be sure of continued development to keep you in touch with CASE evolution.

NO OTHER STRUCTURED DESIGN TOOL EVEN COMES CLOSE TO OUR PRICE/PERFORMANCE

JUDGE FOR YOURSELF

Telephone 0279 626111 for more information

Semantics Ltd. Wych Elm House, Wych Elm, Harlow, Essex, CM20 1QR

CIRCLE NO 35

Consultancy

A small, flexible consultancy specialising in UNIX

On Site Courses

- Introduction to UNIX
- C Programming
- UNIX Systems Programming
- Bourne and 'C' Shells
- Make & SCCS

Presentation can be tailored to suit client's requirements

Software

- UNIX Device Drivers
- Windows
- Networking

BRAZIER

SYSTEMS AND CONSULTANTS LTD

8 Valley Rise Sarisbury Green Southampton SO3 6BN
Telephone: (04895) 3442

CIRCLE NO 36

PRENTICE HALL ARE PLEASED TO ANNOUNCE THE NEW EDITION OF *THE* CLASSIC TEXT ON C

KERNIGHAN & RITCHIE'S

THE C PROGRAMMING LANGUAGE
SECOND EDITION

(Based on draft-proposed ANSI Standard)

During its 10 years of publication, The C Programming Language (1/E) written by the developers of the C language – Brian W. Kernighan and Dennis M. Ritchie – has become the best selling standard reference to the language. Originally designed for use on the UNIX operating system, this fast, efficient and versatile language is now being used on many other operating systems including MS-DOS.

In this new book, the authors incorporate the latest changes in the ANSI Standardisation of C and is now based on the draft-proposed ANSI Standard.

Kernighan and Ritchie have completely rewritten their seminal text to –

include new constructs that have entered the language

improve upon their original presentation by giving greater clarity to sections of the book, and replacing many of the problem sets with new exercises

discuss Function Prototypes which adds to * the language * the ability to check function arguments for consistency and * C libraries – enabling access to different program facilities and utilities such as maths, I/O functions, operating systems etc.

NOTE: The First Edition will remain in print for programmers using the original compiler.

1988 256 pages 13-110362-8 £24.95

For information on this and other titles
SEE US ON STAND 31 AT THE

EUROPEAN UNIX® USER SHOW

7–9 JUNE 1988 ALEXANDRA PALACE, LONDON

CIRCLE NO 37

Your Clear Path to X-Windows

We can help you exploit this important new software standard:

- ✔ Documentation
- ✔ Training
- ✔ X11 Source Code
- ✔ X Servers
- ✔ X for IBM PC
- ✔ X Software

and much more ....

Call us now on

0223 68749

IXI Limited

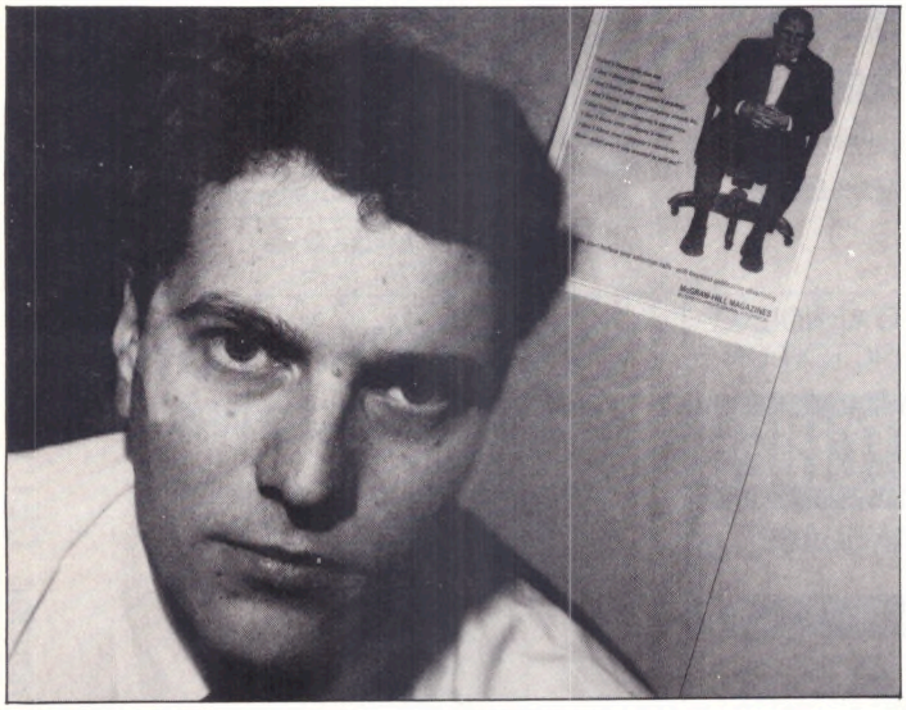
X-Window System is a trademark of MIT

IXI Limited, 6 Wellington Court, Cambridge CB1 1HZ

CIRCLE NO 38

Sun – In Transition from Workstation Vendor to UNIX Standards Maker

User interfaces, software interfaces and microprocessors are at the top of Sun's hit list for openness and standardisation. It reckons it's got until 1990, when OS/2 becomes a major force, to make the UNIX/SPARC/Open Look combination an industry standard. Mark Adams interviews Sun Microsystems Europe's Marketing Manager, Alex Osadzinski as the race begins.



Sun at the Crossroads

Sun is not very popular within the UNIX industry at the moment, except perhaps with its customers and with AT&T. No hardware manufacturers, least of all Apollo, liked the ominous way the company grew so quickly in the workstation market. But you can't really admit that you don't like success. So when AT&T announced that future versions of UNIX would incorporate Sun's SunOS operating system, the industry had a perfect opportunity for a legitimate moan. And now, Apollo, DEC, HP and IBM are all accusing Sun and AT&T of doing a dirty on them.

Sun has vaulted itself slap-bang into the centre of some of the most important and most politically sensitive areas in the entire computer industry, not just with this recent furore. Its original workstation architecture (Ethernet, VME and UNIX) became a standard followed by Apollo, HP, Tektronix and many others. The company's RISC chip architecture, SPARC, has found itself on every major hardware manufacturer's shortlist for their next generation machines. With 250 licensees, it's NFS is a de facto industry standard. It's operating system, SunOS is being pulled into the next official release of UNIX. And even where Sun fails to make standards, such as the false start on its windowing system NeWS, it comes back to try again.

Last year, Sun was a workstation vendor. This year, Sun is a developer and promoter of industry standards. And it knows it's become a major high-level player simply by the reaction from other vendors to its close links with AT&T.

Openness – The Route to Growth

I started by asking Osadzinski about the areas in which Sun was looking to grow in the future, to find that the UNIX market has some severe limitations for Sun. Osadzinski says, "Our original market was Bill Joy's friends buying Suns and everyone said we'd go bust when they'd all bought one. Now we are quite rapidly operating in a wider market than where we originally started, into some quite surprising locations, purely commercial installations and large IBM shops. The trouble with the workstation market, or the UNIX market, is that it's growing at maybe 70% per year, whereas Sun has been growing at, so far, a compound rate of 100% per year. The UNIX market is only about $4-$6 billion, which is just a tiny pimple on the backside of the overall DP market.

"What we're really trying to do is to broaden the scope and attractiveness of the open systems computing market. We want to enlarge the playing field in which companies like us are playing. We don't see any point in focussing all our efforts on capturing market share from other competitors in this fairly small market."

Osadzinski sees Sun's role in 'growing the market' with almost missionary fervour. "We want to really act as a powerful force to change the way the computer industry works. The activities we have been engaged in, say with AT&T and many of our other partners, are designed to grow the open systems industry at the expense of the closed proprietary vendors. We make absolutely no bones about that.

"In a way, we're making a rod for our own backs, making it more competitive. But on the other hand, we'd rather do that, growing the market exponentially, that just fighting in some small proprietary market. I think if the workstation business hadn't really gone open (and every vendor is now selling some variant on the open systems model now, or at least paying lip service to that) I doubt very much if it would grow."

Standards and Openness

Osadzinski believes Sun can maintain a lead in this market by ensuring that its products compete well on price, performance, reliability, and serviceability. Yet Sun's role in creating standards is a keystone in its strategy for growth. Osadzinski refers liberally to the need for standards and openness, but never actually says that Sun want to create standards, though this is clearly what Sun is trying to do. Before I raised the concern of some manufacturers that Sun and AT&T were about to make UNIX closed, I asked Osadzinski what other standards the industry had needed and which had been provided to date. The first two were networking and software interfaces.

"Five years ago, the messages we were pushing were UNIX, Ethernet and VME. Nowadays they're just run of the mill stuff. Standardisation was definitely needed in networking. Five years ago, a person switched on to networking was running on Ethernet using TCP/IP. That's what everyone running UNIX on a VAX was doing. There was a need for some distributed capabilities which is where NFS came in.

Software Interfaces

"There was also a need for standardisation on software interfaces, and that's still not finished. Five years ago, if you were writing a dumb terminal application for UNIX you could sort of write it to either IV.1, or was it IV.0 at the time, or System V which was just about around then. It only ran on VAXes and on workstations from this funny company called Sun who had just shipped 50 machines. I guess Xenix was really ruling the roost then and over that time, a lot of people have seen the need to standardise in UNIX, in software interfaces. I guess that effort is three quarters complete now. Things have really accelerated rapidly since X/Open got together to do serious work in clarifying and setting down these interfaces and making the member companies commit to these. Now we're seeing a lot of companies in Europe and the US asking us questions like 'What about the X/Open CAE, should we be writing to that?', 'What about POSIX?', 'What about SVID?' People are really sitting up and taking notice of these."

Yet, the whole software interface standardisation effort, 75% finished, according to Sun, is quite a long way from completion. The field is still wide open. Osadzinski's two other major areas of standardisation are the user interface and microprocessors. On user interfaces, most hardware vendors would acknowledge that X-Windows was now a clear system of choice. Not so Sun, which is still desperate to promote its own offering, NeWS, to the position of interface standard.

I can't say today that we will offer OS/2 to our users. Vast components of it are very proprietary.

The User Interface

"The current hot topic of debate for the next 18 months is the user interface, and it still isn't clear how the dust will settle. Clearly X.11 will be important, though now it looks as though NeWs will be important in its own right, but even more so as a merged product.

"We announced NeWS in a blaze of optimism, assuming rather naively that because it was technically superior, people would beat a path to our door and just immediately adopt it. It was an interesting object lesson in the fact that exciting new technology which wipes the floor with old style technology doesn't automatically win. It's just that the momentum behind X.11 was quite strong. The reaction took us quite by surprise, yet even the staunchest proponents of X.11 would accept that NeWS was technically superior. When we announced the NeWS/X.11 merge the arguments seemed to go away because it became clear that Sun's commitment to X.11 was quite strong."

Osadzinski cites the need for "true applications portability" to explain why anyone needs NeWs. "It's going to be extremely difficult to make X.11 support the next generation of graphics devices," he says. "I mean, they're all working on very high performance graphics devices, and with Postscript which is the graphics language used by NeWS we can address these devices without worrying what the actual output end is. Using Postscript, we can address future products from Sun and also future products from other vendors. While the pace of product innovation is so rapid, the last thing we want to do is to rewrite an application. Our goal is that an application running on, say, a dumb Sun, runs fine, but if you plug in a 2D or 3D graphics accelerator, it just goes much faster. If you stick a high resolution colour monitor which the application takes advantage of, it goes faster as well. That's infinitely easier using Postscript than it is using X protocol.

"We don't want to see the world stuck on X.11. We have a strong commitment to it, but it just would be a great shame if we slowed ourselves down and everybody else down just at the point where we have the opportunity to grasp real applications portability."

Microprocessors

The final important standards area for Sun is microprocessors, where the company has the strong whiff of a fierce battle on its hands. Of the four major architectures licensed on the open market, Osadzinski contends that only two, and maybe just one, will win out.

"The world is moving to RISC. I don't think it's just fashion, there really are clear advantages in most applications in moving to RISC, not just lower cost and higher performance.

"The number of people selling RISC architectures number about 10. Now many of those are closed. If you look at the ones people are really selling on the open market, there's AMD, Motorola, MIPS and Sun. Those are the four pushing today. Out of those, I contend that no more than two of them will win out as being major forces. There are advantages to having standard microprocessors, just because of the body of software that exists around them. In a way, the work we're doing with AT&T, and they're doing a lot of it on different ABI's, reduces that pressure because it leads to applications that can move between different machines of the same architecture without change."

But Osadzinski recognises that winning, not just competing, in the market is vital to Sun's growth. "Companies are now making judgements on RISC chips based on technical merits but the overall discussion always comes down to 'Which company should we work with and which will win?'. So that's the hardest thing to sell. You can't say I guarantee you we will win. But it's still a fiercely competitive market and we're going after market share the same as anybody else." But even before the European UNIX User Show in June 1988, SPARC had shown itself to be very close to winning the two-horse race in which the Motorola 88000 is the other horse. Already, companies like Xerox, Fujitsu, ICL and Sun themselves are planning or producing SPARC-based workstation, while companies like LSI Logic, Cypress Semiconductor and BIT have taken licensing agreements.

Outside the Workstation World

If user interfaces, software interfaces, networking and microprocessors are the major standards in the workstation world, what happens when Sun expands beyond this area? Osadzinski has already explained that the workstation market is limiting for Sun, and the announcement of the 386i microcomputer, based on Intel's 80386 processor, is a step in this direction. Outside the workstation market, Osadzinski still sees networking as a major force, and places great importance on the existing operating system standard, MS-DOS, and it's successor, OS/2.

Even the staunchest proponents of X11 would accept that NeWS was technically superior. We don't want to see the world stuck on X11.

On networks, he makes a poignant comment, showing Sun to be flexible on standards issues where required. "We see that there are more powerful forces at work, for example in the OSI world. Some are closed and proprietary, which we don't like, but we know our customers want to talk to, and others which we see as being important powerful forces for the future. So we were one of the earliest companies to bring OSI products to market. In fact, our implementation has been used as a reference for OSI performance testing. We participate in many standards making efforts, such as CCITT and ISO, and we're slowly migrating our product base towards them. It's only a personal guess, but I would say five years from now, most Sun networks will be running ISO protocols rather than TCP/IP. That's just a natural evolution, we're not fighting it, we're not trying to push it forward fast either."

The OS/2 dilemma

The real issue facing Sun, however, is what to do about MS-DOS and OS/2. It's not difficult to imagine Bill Joy, Sun's architect and now AT&T's brainchild on UNIX V, 4.0, being physically sick at the prospect of Sun offering MS-DOS, which is clearly regarded as a non-serious, toy operating system by the UNIX cogniscenti. So imagine the boardroom meetings (and this is important, as Osadzinski says later) where Sun has got to consider not just MS-DOS, but OS/2 as well. Now OS/2 isn't a toy, but it's far too close to UNIX to make it an obvious product to offer. And imagine the difficulty Sun might have in reconciling a product offering comprising OS/2 and the Windows / Presentation Manager user interface, opposite a UNIX/NeWS/Open Look product. Not only would they compete head on, but OS/2 is quite likely to greatly undermine the work Sun has already done on UNIX and Open Look. At this stage, Sun is not saying that it definitely will not offer OS/2 (and it's not saying that it will either).

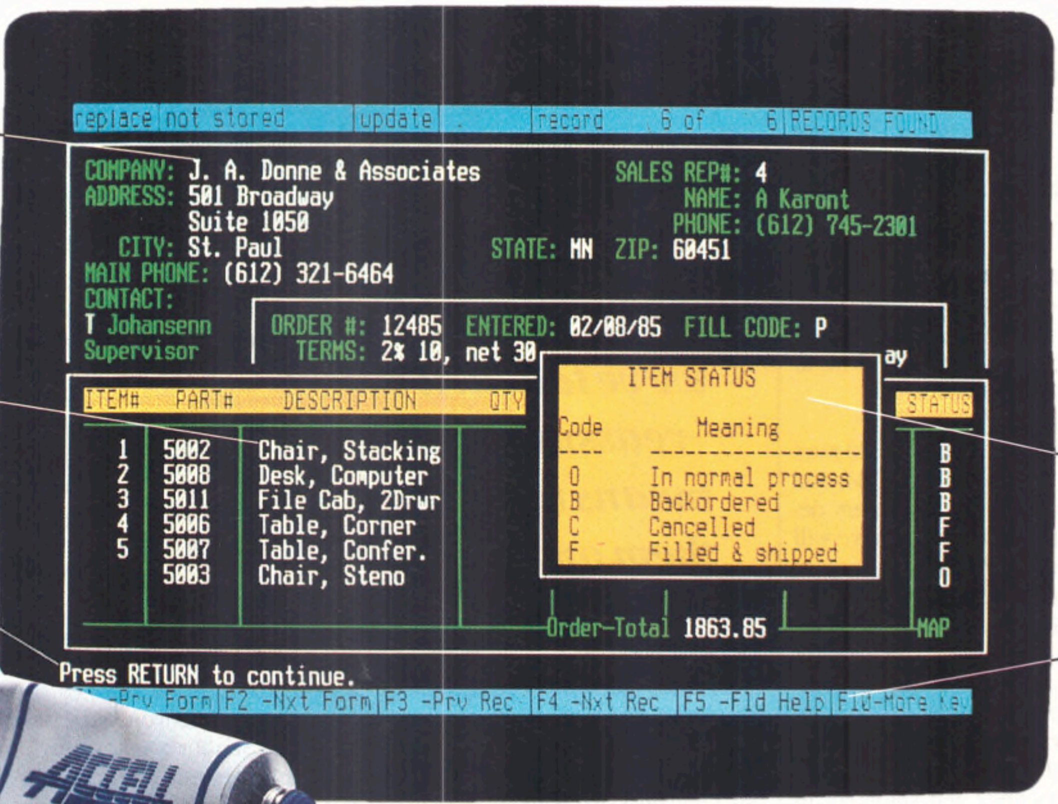
A FEW POINTS IN FAVOUR OF ACCELL

Integrated Development System from Unify.

Query By Forms quickly finds data with just a few keystrokes.

Vertical and horizontal scrolling fields make data location easy.

Field-sensitive helps let even novice users make veteran choices.



```
replace|not stored       |update|  .     |record     6 of     6|RECORDS FOUND

COMPANY: J. A. Donne & Associates                SALES REP#: 4
ADDRESS: 501 Broadway                                  NAME: A Karont
         Suite 1050                                   PHONE: (612) 745-2301
   CITY: St. Paul                      STATE: MN  ZIP: 60451
MAIN PHONE: (612) 321-6464
CONTACT:
T Johansenn      ORDER #: 12485  ENTERED: 02/08/85  FILL CODE: P
Supervisor          TERMS: 2% 10, net 30                        ay

ITEM#  PART#   DESCRIPTION      QTY       ITEM STATUS          STATUS
                                      Code        Meaning
  1    5002    Chair, Stacking        ----   -----------------     B
  2    5008    Desk, Computer          0     In normal process     B
  3    5011    File Cab, 2Drwr         B     Backordered           B
  4    5006    Table, Corner           C     Cancelled             F
  5    5007    Table, Confer.          F     Filled & shipped      F
       5003    Chair, Steno                                        O

                                Order-Total 1863.85               MAP
Press RETURN to continue.
   -Prv Form|F2 -Nxt Form|F3 -Prv Rec |F4 -Nxt Rec |F5 -Fld Help|F10-More Key
```

Exclusive ZoomView feature lets users scan and retrieve data from elsewhere in the data base without leaving the original screen.

Interactive windows, for helps and prompts, can be conditional or automatic.

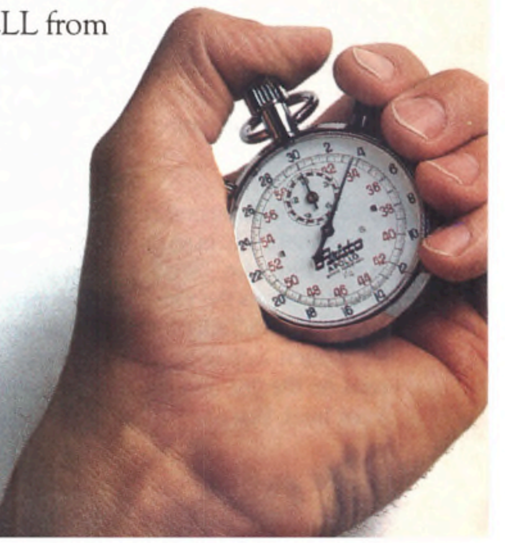
Function key labels let users quickly change records or get detailed help and additional functions.

ACCELL lets you make changes with the speed and simplicity of a brushstroke.

Applications that look good and perform at 'run-time.'

Get to know more about ACCELL from Unify Corporation.

STAND No. 23d



UNIFY CORPORATION

Unify Corporation
Haleworth House, Tite Hill
Egham, SURREY TW20 0LT
Telephone: 0784 71021
(International +44 784 71021)
FAX: 0784 37564

ACCELL is an integrated application generator, automatic windowing interface, complete 4GL and SQL Relational Database.

10,000 professional developers have already got the point – using ACCELL to deliver applications to their users on schedule.

CIRCLE NO 30

OS/2 brought with it a rash of announcements from major hardware and software vendors who said they would port their product to it, or that it would run on their hardware. The industry support has been consistently high. I found it interesting that a company with such a forward looking view on the world as Sun, had yet to decide on OS/2. I put this point to Osadzinski.

"With our 386i product, we have a multitasking DOS and UNIX workstation. And so we're giving to people today what they hope they'll get from OS/2 towards the end of the year, and some things they'll never get from OS/2, like multi-tasking DOS. And corporate developers are making decisions now about how to move to OS/2, and even whether or not to move to OS/2. The transition certainly isn't painless. It gives us an opportunity because the transition from MS-DOS to OS/2 is about as painful as the transition from MS-DOS to UNIX. UNIX does everything that OS/2 will do today, and more."

Sun is playing an awful fudging game on OS/2. It is torn between sticking with UNIX and offering what users may want in OS/2. If it offers OS/2 it may turn out to be to Sun's long term detriment.

"We're already looking at a number of ways of providing compatibility with OS/2 and UNIX and certainly if our customers demand it, which seems likely, then we will be doing that in much the same way as we're providing compatibility with MS-DOS and UNIX. But I can't say today that we will offer OS/2 to our users. The decision isn't taken, and one factor is 'will OS/2 be sufficiently important' or 'will our customers want us to provide OS/2 compatibility?'. I think that the delay between the announcement of OS/2 and shipment in final form is giving UNIX an opportunity to capture large amounts of market share at the low end, just in that period. A year is an awfully long time.

Osadzinski had previously answered this question for himself. "OS/2 will undoubtedly be important and I'm sure it will have over 1 million installations by the end of the year, if not before, and we will have to provide a compatibility path for our users.

OS/2 also goes against the corporate grain of openness, and in so doing commits what is a cardinal sin. "We are using the open systems versus the closed and proprietary argument. OS/2's limiting in the sense that it's written largely in assembler and only runs on the 286 and 386 and future chips from Intel. Vast components of it are very proprietary. You can't get source code for it, which doesn't bother that many people, but it does mean it's more tightly controlled than UNIX.

OS/2 is giving the company serious concern. It's a product that looks as though the market will demand, and by its own admission, Sun will then have to offer. On the other hand, it breaks Sun's fundamental philosophy of openness, and threatens the strong position that Sun has built with UNIX.

A Year in the Life...

But as Osadzinski says ominously, 'A year is a very long time.' implying that Sun has set its sights firmly on making some fundamental changes to ensure the success of UNIX in the short term. Perhaps that's what the current flurry of activity on user interfaces, on AT&T buying shares in Sun and licensing Sun's microprocessors is all about. Osadzinski explains,

Some of the 'closed' vendors have reacted very violently to the leads we've taken. It's reassuring, it means we've got them on the run.

"We've already seen how some of the 'closed' vendors have reacted very violently towards some of the leads we've taken, very aggressively indeed. It's sort of scary but it's also reassuring. It means we've got them on the run, it seems that what we're doing is right.

"The major criticism that we've had levelled at us is that we're trying to close things off, which is frankly extremely irritating because we're not. If we were the computer industry as a whole, it would be quite reasonable to assume that we have some ulterior motives in trying to do something nasty where in fact we haven't. So if we're to do any of the things people are accusing us of planning; of closing UNIX off and so on, then we'd deserve terrible retribution brought down on us."

All this focusses the attention back on the so-called Hamilton group of manufacturers who moaned so loudly at Sun and AT&T. What exactly are they complaining about? HP and Apollo could say no more than they we're concerned about UNIX being closed or about Sun getting pre-releases of new UNIXes, thus giving them an unfair competitive advantage. These are flimsy arguments which have yet to be substantiated.

It's difficult to see who would really win by UNIX being closed, and the only way Sun would benefit is if it made money from future sales of UNIX. "The role of Sun in the production of System V release 4.0 is very much as a subcontractor to AT&T. We're not putting an iron grip around the throat of UNIX forever." says Osadzinski. Karen Rohack, official Sun spokesperson in the US confirms that Sun will not derive royalties from future releases of UNIX.

Perhaps the concern over Sun and AT&T making UNIX closed is really just a communication problem. "When we announced the Sun/AT&T agreement, it was unfortunate that we didn't make much clearer the fact that the SPARC ABI is just one of many, that the reference release for UNIX would still be on the 3B. We will of course optimise UNIX for SPARC and AT&T will be doing similar work for the 386. Presumably Unisoft and Motorola will do the same for the Motorola architecture and so on."

A Personal Ending

Osadzinski concludes by saying that "Unless you, or anyone else, personally worked at Sun for a year or two, and attended the strategy meetings and the board meetings, that would be the only way to convince you we're not trying to do something nasty. I can't convince you we're not doing anything like that, you'll have to trust us."

At the moment, we'll have to take Osadzinski's word that Sun's keeping UNIX open. If the outcry against Sun and AT&T so far has been aggressive, there's no telling what would happen if elements of UNIX were to become proprietary.

Osadzinski was interviewed on 4th May 1988. Developments after 12th May may change the context of the above article. Mark Adams is a regular contributor to .EXE, and writes extensively on software development issues.

TO FIND OUT MORE ABOUT ADVERTISING IN THE .EXE RECRUITMENT SECTION TELEPHONE OLIVER HILL ON 01 994 6477

Understanding people...

As the UK's leading independent UNIX Recruitment Consultancy, naturally enough we have very many UK based vacancies for UNIX and 'C' literate people on our books. Vacancies for people at all levels of experience and at all locations.

However, it seems that our reputation has spread further afield as we have also been retained to recruit for companies in Italy and the USA.

In addition to the 'C' UNIX engineers who are the mainstay of our business, we are particularly keen to hear from:

'C' UNIX and COMS Consultants with in-depth systems skills able to relocate to the States.

Senior UNIX S/W engineers with large project management ability, able to relocate to Italy.

A UNIX Product Engineering Manager who must have excellent interpersonal and man-management skills, to be based in Italy.

A heavyweight Systems Software Architect, to be based in Italy.

If you are thinking of a career move in the UNIX arena the help and advice of knowledgeable insiders is always useful. Call us, in confidence, on the numbers below or visit us on stand 32 at the EUROPEAN UNIX USER SHOW, Alexandra Palace, London, and we will be happy to assist.

Write or phone...

Linda Elisha
LJE UNIX Recruitment
48-50 Gainsford Street
Butlers Wharf, London SE1 2NE

Daytime Telephone (01) 378 7588
Evening Telephone (01) 317 8472

LJE UNIX™ recruitment

UNIX is a trade mark of AT+T in the USA and other countries.

MARK Out Your Next Career Move with ONE of Our Opportunities!!

Technical Project Managers, Consultants & Designers
To £35k + Car + Benefits

Probably the world's leading UNIX Systems House seek Degree educated Computer Professionals. The ideal candidates will have a minimum of 3 years in-depth UNIX and 'C' experience and a fundamental understanding of ISO/OSI standards in Complex Distributed Systems and Networks.

Software Designers (London & Hants)
To £25k + Car + Benefits + Relocation

This major International Software House seek experienced Software Designers with a background in Realtime Networks, Comms of Office Automation. Software is produced in 'C', Pascal, Coral or ADA in Banking, Finance and Defence Sectors. These are exceptional career opportunities for the right candidates. A minimum two years relevant experience is required in UNIX/XENIX or Tandem environments.

Senior Software Engineers/Software Engineers
To £18k + Car + Benefits

This leading International Software House seek Software Engineers with a minimum of 2 years experience in 'C' or Assembler in a UNIX, MS-DOS, OS/2 or Networks environment. Candidates, ideally with exp. of CASE tools and 'workbench' concepts, will assist in the development of Leading Edge Products for the DP Industry.

For further information about these openings, please send your CV to us at the address below or contact us direct up to 7.30pm weekdays on 01-387-4699 or 01-388-3111 ext 2273/2190/2272

MARK ONE RECRUITMENT
THE PROFESSIONALS CHOICE

Management and Personnel Recruitment Consultants

Mark One Recruitment Ltd

Euston House, 81-103 Euston Street, London, NW1 2ET.

* UNIX IN THE CITY *

Due to continued expansion, this prestigious international city based company have a number of challenging vacancies for UNIX professionals:-

Technical Manager

With a technical background and possessing good inter-personal skills. Database experience a distinct advantage.

Project Leader

UNIX and Unify skills with a desire to move into a more managerial role.

Programmer

UNIX/C expertise wishing to expand general technical knowledge, with particular emphasis on databases.

An excellent package is offered, along with a well-defined career path.

Call (01) 439 1188 Amanda Reed or Martin Congreve.

Regency Computer Services
Premier House, 77 Oxford Street, London W1R 1RB
(Recruitment Consultancy)

UNIX + 'C'

'C' MOVE INTO FINANCE

Herts **£14-£18,000**

Our client is one of the country's leading Life Assurance companies covering the whole spectrum of personal and business finance. Committed to further development of their UNIX based systems they now require a number of highly motivated individuals at Programmer and Analyst/Programmer level. Candidates should have a minimum of 18 months 'C' experience gained in either a UNIX or MS-DOS environment. The successful candidates will join a project team involved in the development of both financial and MIS applications and will receive full training where required, whilst working within a structured environment. EX051/3949

UNIX & 'C' ANALYST/PROGRAMMER

London **£15-£20,000**

Our client, leading their field in the design, manufacture and installation of sophisticated high tech security systems, currently require an Analyst/Programmer with a minimum of 18 months experience of 'C' and UNIX gained in any environment, to become involved in the design and development of a variety of unique and highly interesting systems. The successful candidate will develop new software skills such as ESQL and INFORMIX on a variety of UNIX boxes. On top of this you will enjoy an excellent team atmosphere, along with highly competitive remuneration. EX051/3952

'C' OPPORTUNITIES

London/Home Counties **to £20K**

We currently have a large selection of software houses and a variety of end user sites who require people with varying levels of experience in 'C' to work in a wide range of fields, including communications, the financial sector and commercially oriented environments. Candidates should ideally have experience in one of the following: UNIX, VMS, MS-DOS, along with a minimum of 12 months 'C' programming experience. These positions offer highly competitive salaries and benefits, along with the opportunity to progress your career to its full potential. EX051/3948

Price Gunning Associates
Recruiting Professionals

BUCKINGHAM COURT
78 BUCKINGHAM GATE
LONDON SW1E 6PE

01 · 222 7151
OFFICE HOURS (9am to 9pm)
EVENINGS & WEEKENDS
01 · 679 7882
01 · 747 3647

RNW ENGINEERING RECRUITMENT are technical recruitment specialists covering the Home Counties, M3 and M4 corridors. We are currently seeking to fill the following vacancies.

SOFTWARE ENGINEERS **HANTS** **£11,00-22,000**
Comms products. Various levels. C, PASCAL, DSP, microcontrollers.

ANALYST/PROGRAMMER **SURREY** **to £17,000**
Information storage/retrieval products. 2-3 years C essential. UNIX, LANS, Windows, 8086 desirable.

SOFTWARE ENGINEER **HANTS** **to £20,000**
Network applications software. Protocol converters, C, X25, 4GL's.

SOFTWARE ENGINEER **BERKS** **to £17,000**
CAD packages. SUN workstations, C, UNIX, MS-DOS, ICE.

SOFTWARE DEVELOPMENT ENGINEER **HANTS** **£15,000**
Radio comms. BSc. 18 mths experience C, UNIX, MS-DOS, ICE.

SOFTWARE QUALITY ENGINEER **MIDDX** **£13,00**
Thermal imaging systems. HNC. Experience in control of software. AQAP-1.

SOFTWARE ENGINEER **SURREY** **£neg**
Systems design consultancy. C, PASCAL, UNIX, BSc plus 2 years experience.

SOFTWARE ENGINEER **BERKS** **circa £12,000**
Telecomms products. Monitoring and control applications. C, PASCAL, UNIX.

To register for these or many other vacancies, or to discuss your own requirements, please call **Mike Patton** or **Dave Lee** on the number below.

RNW Engineering Recruitment

Harlen House, 44 London Road, Staines, Middlesex TW18 4HD
Telephone: Staines (0784) 63731 (24 hours).

PROGRAMMER ANALYST

Central London **To £20,000**

*** UNIX * C * ASSEMBLER * VMS**

Due to the continued success of its products and the rising demand for its services our client, an established systems house, are currently looking to recruit an additional Programmer Analyst for the technical department.

This department handles a wide range of functions which include:

* Designing and writing specialist software in C and Assembler
* Testing new hardware and software products
* Developing standards and utility programs for use throughout the company.
* Installing, upgrading and supporting DEC operating systems and layered products (PDP-11 and VAX).
* Installing and supporting network communication equipment and software
* Solving on-site operating software problems.

Ideally you will be a graduate in a numerate subject (A levels/HNC/HND will be considered) with upwards of 18 months programming experience in C and/or Assembler.

Of equal importance is self motivation and the ability to work in a dynamic environment where individual flair and contributions are rewarded with excellent career prospects.

For further information and an early interview contact
Rachael Huddleston on 01-686 9216.

BRIDGEWATER COMPUTER SERVICES LTD
FREEPOST FIRST CLASS 49 SOUTH END, CROYDON, SURREY CR9 9ET

Greenfields opportunities for

I.T. PROFESSIONALS

Hants **£13-25K neg.**

Our client, the Data Networks Division of a very successful International Group, has developed a world beating state of the art Data Communications system that will put the rest in the shade. Incorporating the latest X.25 technology with planned integration of ISDN this system will challenge even the best I.T. Professionals.
Opportunities exist for

• SYSTEMS DESIGNERS •
• SOFTWARE ENGINEERS •
• HARDWARE INTEGRATION ENGINEERS •
• SYSTEMS SUPPORT ENGINEERS •
• OPERATIONS SUPERVISORS •
• NETWORK CONTROLLERS •

HARDWARE	LANGUAGES
TELEMATICS IBM LARGE SYSTEMS CMS	"C" COBOL PASCAL
PROTOCOLS	**ENVIRONMENT**
X.25 SNA IPA CO3	TRAX INF MVS/XA UNIX

If you possess any of these skills, are exceptionally numerate and literate, are a good communicator and are service orientated, ideally with experience of working within Government environments, send your current C.V. to either FREEPOST address below, or telephone for an application form on
FREEFONE "ENGINEERING RECRUITMENT"

EES COMPUTER RECRUITMENT

FREEPOST
East Grinstead
W Sussex RH19 3ZA
0342 315033

FREEPOST 491
Rickmansworth
Herts WD3 6FP.
0923 777966.

WEST YORKSHIRE

UNIX experienced Analyst Programmer for user moving to new Apricot VX9000 (training given on ACCELL 4GL and UNIFY database)	£12,000
'C' Programmer to develop new 4GL environments.	£15,000
Analyst/Project Leader with UNIX or Sculptor 4GL experience	£14,000 + Car
Analyst programmer with 'C' or 4GL under UNIX or XENIX in commercial environment to learn PROGRESS 4GL	£13,500
Analyst Programmer with ORACLE database/4GL experience	£17,000 + Car

If you are a Programmer, Analyst programmer or Project Leader seeking the UNIX environment best suited to your future career development in West Yorkshire then telephone:

Vincent Atherton on

Leeds (0532) 504560

9 am - 9 pm

AIREDALE RECRUITMENT

THE ROUNDED ROLE OPPORTUNITY

FOR

'C' SOFTWARE DESIGNERS

WITH A

MAJOR INTERNATIONAL DATA COMMUNICATION COMPANY

c. £18,500 **Thames Valley**

We are retained to advise on the appointment of calibre individuals to be part of a small yet important group developing facilities for major data network products.

You will require fluent personal communications skills to effectively interface with customer and company personnel. This interface leads to the design and development of critical softwares to meet challenging requirements. You will be responsible for performing verification and testing to the approved performance, documentation and assisting in field trials through to initial customer support.

Ideally, you will offer:

- graduate level education
- three years plus development experience of communications software.
- proven 'C' application development
- knowledge and appreciation of database environments
- experience of real time software and communication protocols.

The time may be right for you to use all of your skills and abilities in a well rounded and technically responsible role. As a leading player in the international data communications market, our Client offers an advanced technological environment, excellent training and career development prospects. As you would expect, the realistic salary is accompanied by a generous benefits package, with relocation assistance available.

Contact us by telephone for prompt attention or forward your C.V. quoting reference CTV/293d.

Personnel Resourcing Consultants
COMMSOURCE
Graphic House, Station Approach
Chipstead, Surrey CR3 3TD.
Fax: 0737 554916
Telephone: 0737 552211

CATCHERS
RECRUITMENT

THE FRESH NAME TO RECRUITMENT IN THE COMPUTING FIELD, WE ARE AIMING TO PROVIDE A HIGH QUALITY SERVICE AT SUCH A LOW COST YOU MAY HAVE THOUGHT IT IMPOSSIBLE FROM AN AGENCY.
WHY NOT TELEPHONE JOHN HILL FOR FURTHER DETAILS ABOUT A MORE REASONABLE SOLUTIONS TO YOUR- STAFFING REQUIREMENTS

6 The Market Lanes
Terminus Road
Littlehampton
West Sussex
BN17 5BS
Telephone: (0903) 726970

UNIX/ COMMUNICATIONS DEVELOPMENT
Berkshire **To £22K**

Within the R & D Department of Europe's most successful manufacturer of super mini's, a unique opportunity exists to join either the Systems Software or Data Communications Development Group. Graduate calibre individuals within a minimum of one years UNIX, C, or OSI experience are required to maintain the remarkable progress already made by this highly profitable organisation. In exchange for your total committment, our client will provide the freedom (and financial rewards) necessary to maximise your potential.

To find out more about this exciting range of opportunities, please contact **DAVID JONES,** B.Sc, M.Sc, on **0442-231691** during office hours or at home on **0442-212650** (evenings and weekends). Alternatively write to me at the FREEPOST address below.

Executive Recruitment Services
SOFTWARE DIVISION
Freepost, Hemel Hempstead, Herts., HP2 4BR.

AMANUENSIS RECRUITMENT

EAST ANGLIA, THE MIDLANDS, HOME COUNTIES

PROJECT LEADERS
SOFTWARE ENGINEERS/PROGRAMMERS
SYSTEMS PROGRAMMERS
TECHNICAL AUTHORS
£12-25k

The following skills are in constant demand by our clients - please call or send us your Curriculum Vitae if you are interested in seeking alternative employment and have ANY of the skills in ANY combinations. Our Consultants will be happy to talk to you at home out of office hours if you find this more convenient.

* Pascal	* Artificial Intelligence
* C	* UNIX
* Assemblers (16 & 32 bit)	* MS-DOS
* Basic	* VMS
* FORTRAN	* VAX/VMS
* Modula-2	* CAD/CAM
* Datacomms	* Telecomms

Please contact **Jean Sandercock** (home no: 0638 663696) or **Richard Woodward** (home no: 0223 329509)

Technical Authors, especially with experience of UNIX, CAD/CAM and PCs, who have written user manuals from initiation through to printing should contact **Elaine Forster** (home no: 0223 853532)

AMANUENSIS EXECUTIVE SERVICES LIMITED
20NEWMARKET ROAD CAMBRIDGE CB5 8DT ☎ 0223 354978

FOREST COMPUTER STAFF **THAMES VALLEY OPPORTUNITIES** FRES MEMBER

SOFTWARE ENGINEER – GRAPHIC ARTS
Development of RT Software for control of electronic equipment. DEC VAX/VMS useful. Stimulating development department.
Salary: c£15K+
Ref: C.8374

SOFTWARE ENGINEER – CABLE TV
Development of network control, data management, plus design and implementation for next generation systems. DEC VAX, Hi-level language + RT desirable.
Salary: c£16k
Ref: C.8376

SENIOR SOFTWARE ENGINEER
Implementation of RT control solutions to automation using PLC's, PC's, Minis. C, UNIX, VAX/PDP useful.
Salary: £15-18k + Benefits
Ref: C.8409

SOFTWARE ENGINEER
Maintainence and development on telecomms/telemetry products. Varied duties, UNIX, C, PASCAL, DEC, IBM PC all useful.
Salary: c£12k + Benefits
Ref: C.8425

TECHNICAL SUPPORT ENGINEER – UNIX/XENIX SPECIALIST
Variety of special projects. Opportunities for development in the future will arise. UNIX/XENIX, C, PC knowledge all desirable plus good interpersonal skills.
Salary: £15-18k + Car
Ref: C.8426

These are just a small selection of our current vacancies. We have positions for ALL COMPUTER PROFESSIONALS. If you have MS-DOS, UNIX, VMS, RT, OS/2, COMMS, 4GL, PASCAL, C, . . . commercial or scientific experience . . . we shall be pleased to hear from you.

FOREST – 20 YEARS ON – STILL GOING STRONG

Forest ComputerStaff
Tel: (0734) 587272
11/12 Gun Street, Reading.

UNIX IN SURREY

COMMERCIAL SYSTEMS ***£15-23K***

Well-established small Business Systems Consultancy with high profile in state-of-art RDBMS & 4GL, now expanding, offers new opportunities (Programmers to Project Leader) in major new Development Program for Next Generation Products with plans for 5GL! If you've UNIX/C OR PASCAL OR 4GL experience & wish to develop your Business Software career - don't delay!

Call George Burrows 01-549 6441 ref CG 1206

REAL-TIME GRAPHICS ***£15-20K***

Very high-tech prestige Manufacturer with high profile in 32 bit Multiprocessor systems for GRAPHICS, IMAGE PROCESSING & 3-D MODELLING, plans to double R&D activities in '88. Truly dynamic state-of-art environment offering Bit Slice, RISC, Array Processing, UNIX/C, latest processor technology & more! Tremendous variety, USA travel, excellent career potential - in short, everything for Tomorrow's Real Time Professional.

Call James Hunt 01-549 6441 ref CJ 501

SALES SUPPORT ***£15-20K + Car***

Major new product launches in '88 + large installed world-wide customer base + market leader status = new Software Support opportunities with a major international mini-computer manufacturer. IF UNIX + Personality = YOU, THEN SUPPORT = People Contact, Problem Solving, Travel, Training, Varied Applications (Comms, GRAPHICS, A.I., D.T.P.).

Call Allan Sanders 01-549 6441 ref CA 678

For these and other UNIX/C opportunities (& not necessarily in Surrey!) call our UNIX TEAM today!

ab executive
HIGH TECHNOLOGY RECRUITMENT
59 Eden Street, Kingston-Upon-Thames, Surrey KT1 1BW. Tel: 01-549 6441 (24-hrs)

UNIX & 'C'

Salaries to £20,000

London & Home Counties

* Networking and Communications
* ISO/OSI
* Banking
* Compiler Writing
* Office Automation
* Financial Applications
* Graphics

We are currently recruiting for a number of companies seeking Applications Programmers. Software Engineers and Team Leaders to work in the above areas. Our clients, ranging from small, expanding companies to some of the larger organisations, offer candidates excellent training and the opportunity for career progression into some of the fastest growing markets in the computer industry.

If you offer fluency in 'C' and would like to find out more about the positions on offer, why not telephone one of our Consultants on the above number. Alternatively, send your CV and we will make contact at a time to suit you.

logistix FRES MEMBER
Logistix Recruitment Limited
Logistix House, 148 King Street,
London W6 0QU
Telephone: 01-741 8511
Facsimile: 01-563 0166

Promotional Reprints of Articles Featuring Your Product

Considering the fact that other people's independent opinions endorse the value of your product, .EXE would like to offer you the opportunity of ordering reprints of .EXE articles which feature your product or company, for you to include in your promotional mailings, sales packs and press releases.
The articles may be adapted especially for you, to include your company logo, more tables and diagrams, further information etc.

For more information about this ideal promotional opportunity, call Helena Adams on 01-994 6477 ext. 359.

Minimum quantity: 200 reprints.

Company	Page No	Circle No
Addison Wesley	57	20
Arthur Young	43	14
Ashling Microsystems	83	34
Brazier Computer Systems	85	36
California Software	59	21
Computer Bookshop	83	32
Creative Data Systems	69	25
Digital Research	25	8
Excelerator	35	11
Generics	61	22
Glockenspiel	65	28
Grey Matter	3	2
Hitachi	19	7
IBM	78,79	30
IXI	85	38
LBMS	55	19
Michael Jackson	51	17
Microcomputer Unit	37	12
Microsoft	13	4
Microtec Research	17,65	6,29
NEC	5	3
NEOW	41	13
OS/2 User Group	47	15
Prentice Hall	85	37
Prospero	53	18
QA Training	31	10
Ready Systems	15	5
RTS	65	27
Samna International	49	16
Scientific Software	83	31
Semantics	85	35
Software Construction	28,29	9
Sycero	IBC	40
System Science	61	23
Thomson	OBC	41
Unify	89	39
Vivaway	83	33
Xoren	72	26
Zortech	IFC	1

/shutdown

The House of Lords' ruling has already taken space at the beginning of this issue and in the middle, too. It's only right, then, that it gets a mention at the end. I was going to use this column to report what it's like sitting in the Peers' Gallery in the House of Lords. Unfortunately, one is not allowed to enter said gallery unless you have a coloured piece of material dangling from your shirt collar, so I went back home instead (I object to wearing the appendage in question, and have walked out of many a posh lunch rather than be judged by my clothing). Sermon mode ends. Back in the real world, and Honeywell Bull have found yet another way to make money in order to cover the cost of all those stamps (they send me at least 6 press releases a week, and they're all boring). This time, they're taking a feed from the on-screen computer that gives BBC TV viewers the cricket score, and sending it to a speech synthesizer for the benefit of telephone callers. The purpose of the exercise, says the press release, is to give callers what they want "at minimum cost". If this is the case, why charge 38p a minute for the call? There's always the old-fashioned 5p-for-8-minutes lines. Next, to Amstrad Towers, and that wonderful TV ad in which Alan Sugar explains to the Great British Businessperson exactly what's going to happen to their trading channels in 1992 (something to do with Europe, I understand). Standing in front of a row of 8 PC1512's, Alan (I object to the formality of surnames, too) talks of how wonderful it will be when, in 1992, he can make just one computer for the whole of Europe. Perhaps someone should tell him that he already does (remember those DIP switches). Perhaps someone else should tell him that not everyone in Europe speaks the same language, so all the DOS messages will still need translating. Talking of which, a wonderful tale has reached me regarding the translation of the MS-DOS error messages into French. As any good programmer knows, once you start using too many SET commands you get an "out of environment space" message. Not with a certain French PC clone, you don't. This one says "There is no room left in the garden". Well, it's better than "My hovercraft is full of eels", I suppose. While we're insulting Honeywell, let's really keep the lawyers busy and moan about Facit as well. According to their latest advertising blurb, the company now has a 400 dpi laser printer. "Aha", I thought, "perhaps we could use one with our copy of Ventura to preview the magazine pages". Called the sales office. "Sorry, we only do 6, 8 or 15 pages per minute printers". "What about a 400 dpi one?". "How many pages per minute is that, sir?". And I bet they'll blame the advertising agency if they don't sell any printers. Finally, the answer to the competition from March's issue, where we asked what a Wheelabrator does. Unfortunately, even a 2am slot on Channel 4 would be considered an unsuitable place to broadcast the winning entry, so you'll have to make do with the silver medalist instead. John McDermott, who works for Marconi, writes thus: A Wheelabrator is a machine like a big tumble drier. It is used for removing burrs and sharp edges from iron castings. When Thomas Stafford has a spare moment he collects new castings, puts them in his Wheelabrator with a big pile of iron filings and leaves the machine on overnight. Next morning, a nice shiny casting and the General has made another buck. Another use for a Wheelabrator, says the letter, is for adding wear and tear to forged bank notes. Most forgers are conscientious, apparently, and insist on using a Wheelabrator, and perhaps Seagate use them for testing the durability of their disk drives. I think we'd better leave it there.

PROGRAMMING IN dBASE? LET SYCERO dB DO IT FOR YOU!

dB *Overview*
Sycero dB is a complete and powerful program generator for dBase developers. It can be used to develop most types of application.

Sycero dB can develop both simple and complex suites of programs with greatly reduced development and debugging times. Generated programs have a common user interface and are automatically documented to a high standard.

d *efining Databases*
Databases are defined in a similar manner to dBase. For each field, enter a name, description, type, width and decimals. Sycero dB supports arrays on file and in memory. The field and file descriptions are used for documentation purposes. Sycero supports all dBase field types, including memo.

Up to 14 index keys can be defined per database. Keys may be simple (eg CODE), composite (eg SUPPLIER + CODE) or expressions (eg PRICE*QTY).

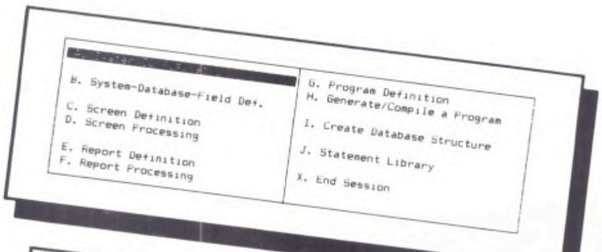
d *efining Screens*
Screens are defined by moving the cursor around the screen and painting words, lines, boxes and coloured areas. Memory variables and fields from up to 30 different databases can be included on each screen.

Up to 10 screens can be included in each program and screens can be called conditionally and overlayed on top of one another.

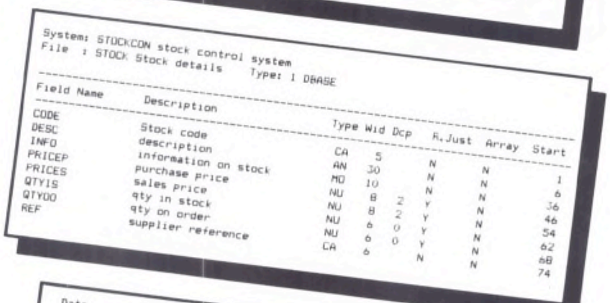
Screen processing is used for validation, calculation, help, error and warning messages, looking at databases and overriding the automatic program logic.

Unlike dBase, Sycero dB is not screen based. Each field is treated separately. Up to 300 lines of code may be added to the SAY and GET for each field, using dBase and/or the Sycero command language. This gives full control over the screen.

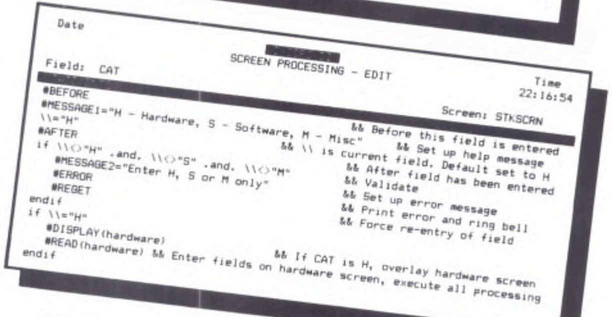
"Sycero dB — The dBase Generator"



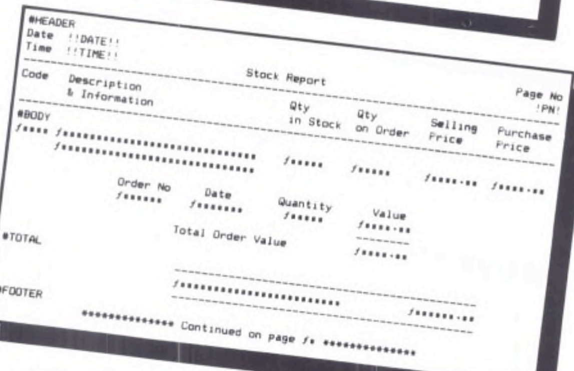
```
B. System-Database-Field Def.          G. Program Definition
                                       H. Generate/Compile a Program
C. Screen Definition
D. Screen Processing                   I. Create Database Structure

E. Report Definition                   J. Statement Library
F. Report Processing
                                       X. End Session
```



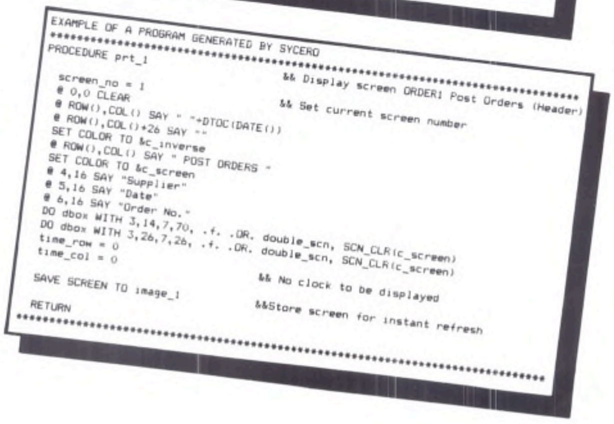
```
System: STOCKCON stock control system
File  : STOCK Stock details      Type: 1 DBASE
---------------------------------------------------------------------------
Field Name     Description
---------------------------------------------------------------------------
                                     Type Wid Dcp  R.Just  Array  Start
---------------------------------------------------------------------------
CODE           Stock code             CA    5        N       N       1
DESC           description            AN   30        N       N       6
INFO           information on stock   MO   10        N       N      36
PRICEP         purchase price         NU    8   2    Y       N      46
PRICES         sales price            NU    8   2    Y       N      54
QTYIS          qty in stock           NU    6   0    Y       N      62
QTYOO          qty on order           NU    6   0    Y       N      68
REF            supplier reference     CA    6        N       N      74
```



```
  Date
                          SCREEN PROCESSING - EDIT
Field:  CAT                                                     Time
                                                            22:16:54
#BEFORE                                                Screen: STKSCRN
#MESSAGE1="H - Hardware, S - Software, M - Misc"  && Before this field is entered
\\="H"                                 && Set up help message
#AFTER                       && \\ is current field. Default set to H
if \\<>"H" .and. \\<>"S" .and. \\<>"M"   && After field has been entered
   #MESSAGE2="Enter H, S or M only"    && Validate
   #ERROR                              && Set up error message
   #REGET                              && Print error and ring bell
endif                                  && Force re-entry of field
if \\="H"
   #DISPLAY(hardware)
   #READ(hardware) && Enter fields on hardware screen, execute all processing
endif                       && If CAT is H, overlay hardware screen
```



```
#HEADER
Date   !!DATE!!
Time   !!TIME!!
-------------------------------------------------------------------------
Code   Description                   Stock Report                Page No
       & Information                                               !PN!
-------------------------------------------------------------------------
#BODY                                 Qty       Qty      Selling  Purchase
                                      in Stock  on Order Price    Price
-------------------------------------------------------------------------
f#### f########################
      f########################       f#####    f#####   f####.## f####.##
           Order No    Date      Quantity    Value
           f######     f#######  f#####      f####.##
#TOTAL              Total Order Value        --------
                                             f####.##
                ----------------------------------------
#FOOTER         f######################
                ----------------------------------------  f######.##
        ############## Continued on page f# ##############
```



```
EXAMPLE OF A PROGRAM GENERATED BY SYCERO
*********************************************************************
PROCEDURE prt_1
                                 && Display screen ORDER1 Post Orders (Header)
 screen_no = 1
 @ 0,0 CLEAR                     && Set current screen number
 @ ROW(),COL() SAY " "+DTOC(DATE())
 @ ROW(),COL()+26 SAY ""
 SET COLOR TO &c_inverse
 @ ROW(),COL() SAY " POST ORDERS "
 SET COLOR TO &c_screen
 @ 4,16 SAY "Supplier"
 @ 5,16 SAY "Date"
 @ 6,16 SAY "Order No."
 DO dbox WITH 3,14,7,70, .f. .OR. double_scn, SCN_CLR(c_screen)
 DO dbox WITH 3,26,7,26, .f. .OR. double_scn, SCN_CLR(c_screen)
 time_row = 0
 time_col = 0
                                 && No clock to be displayed
 SAVE SCREEN TO image_1
                                 &&Store screen for instant refresh
 RETURN
*********************************************************************
```

d *efining Reports*
Sycero includes a powerful report generator. Report layouts are painted on the screen and can contain fields from up to 30 databases. Sycero supports automatic range checks, sort sequence selection, wild card matching, exception reporting, ten levels of sub-totalling and up to 10 secondary report loops per report. It is simple to generate lists, forms and invoice/statement type reports.

The backslash on the example report means list all of the orders for that stock item.

The ability to add up to 300 lines of dBase and Sycero command language to each field gives complete control over the report.

d *efining Programs*
Program definition assembles database, screens, reports and program logic into programs. Sycero supports six standard program types: File Maintenance, Enquiry, Posting, Report, Menu and Batch. dBase and the Sycero command language can be used to alter and extend these basic types.

Sycero dB generates dBase source code from these definitions and compiles it into .EXE programs using a dBase compiler such as Clipper. Generated programs can be modified before compilation. Several programs can be compiled into the same .EXE file and Sycero automatically produces the overlays.

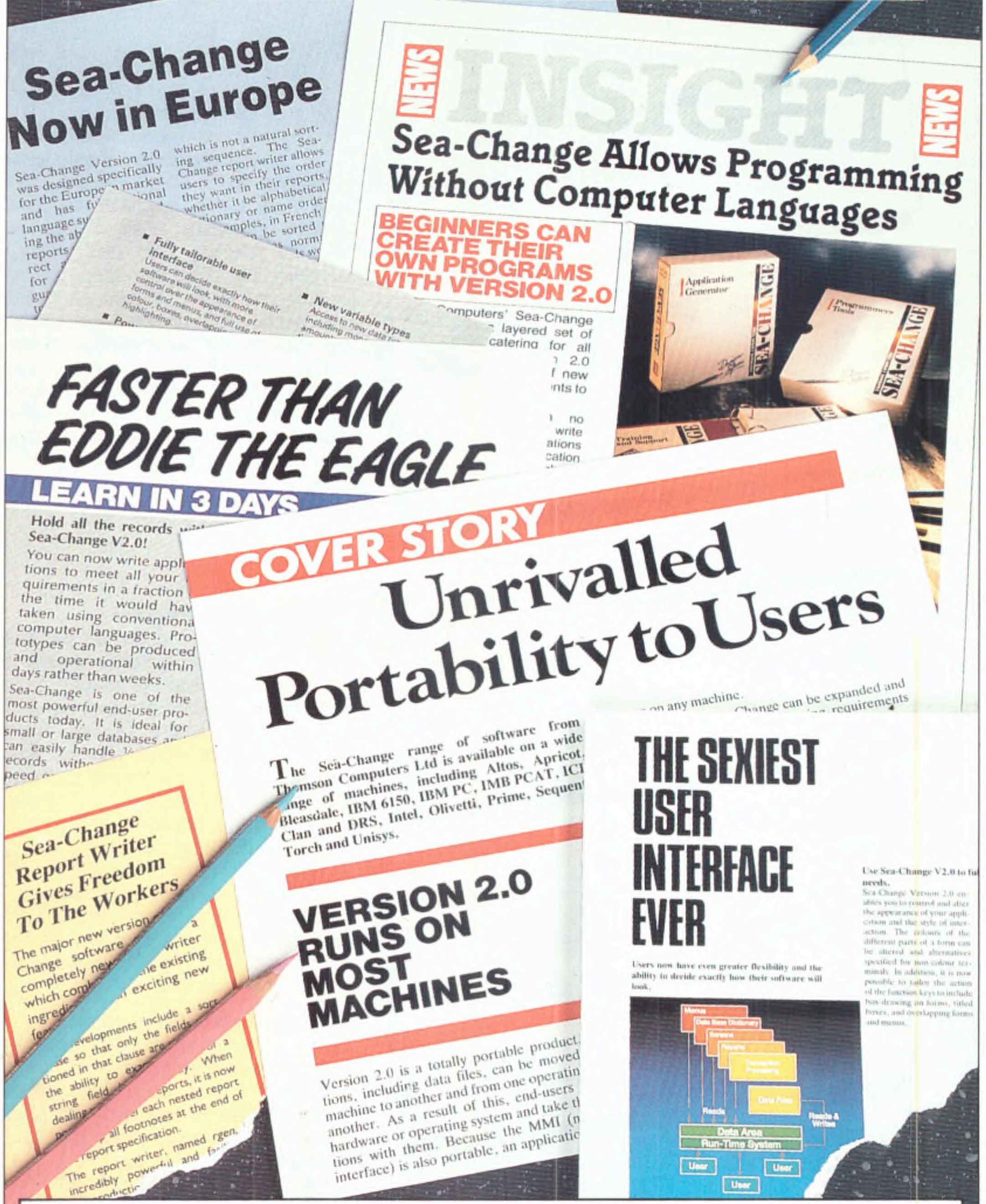
The Sycero Command Language
Sycero supports over 100 new commands to make the dBase language more powerful and easier to use. In addition, external command libraries such as the Tom Rettig library can be incorporated into Sycero definitions.

For further details call Jane at System C.

SYSTEM C LTD, 7 MILL STREET, MAIDSTONE, KENT ME15 6XW. Telephone: (0622) 55142. Fax: (0622) 691241

CIRCLE NO 40

THE LATEST EDITION



Sea-Change Now in Europe

Sea-Change Version 2.0 was designed specifically for the Europe n market and has fu ional language su ing the ab reports rect for gu

which is not a natural sorting sequence. The Sea-Change report writer allows users to specify the order they want in their reports, whether it be alphabetical ionary or name order mples, in French be sorted norm

- **Fully tailorable user interface** Users can decide exactly how their software will look, with more control over the appearance of forms and menus, and full use of colour, boxes, overlappi highlighting.
- **New variable types** Access to new data ty including mon amoun

NEWS INSIGHT NEWS

Sea-Change Allows Programming Without Computer Languages

BEGINNERS CAN CREATE THEIR OWN PROGRAMS WITH VERSION 2.0

omputers' Sea-Change layered set of catering for all 2.0 new nts to no write ations cation

FASTER THAN EDDIE THE EAGLE

LEARN IN 3 DAYS

Hold all the records wi Sea-Change V2.0!

You can now write appli tions to meet all your quirements in a fraction the time it would hav taken using conventiona computer languages. Prototypes can be produced and operational within days rather than weeks.

Sea-Change is one of the most powerful end-user products today. It is ideal for small or large databases a can easily handle ecords witho peed

COVER STORY

Unrivalled Portability to Users

The Sea-Change range of software from Thomson Computers Ltd is available on a wide range of machines, including Altos, Apricot, Bleasdale, IBM 6150, IBM PC, IMB PCAT, ICL, Clan and DRS, Intel, Olivetti, Prime, Sequent, Torch and Unisys.

on any machine. Change can be expanded and requirements

VERSION 2.0 RUNS ON MOST MACHINES

Version 2.0 is a totally portable product, tions, including data files, can be moved machine to another and from one operatin another. As a result of this, end-users hardware or operating system and take th tions with them. Because the MMI (m interface) is also portable, an applicatio

Sea-Change Report Writer Gives Freedom To The Workers

The major new version Change software completely new which com ingred fea

velopments include a sort se so that only the fields tioned in that clause are the ability to ex string field dealing When ports, it is now each nested report all footnotes at the end of report specification.

The report writer, named rgen, incredibly powerful and fa

THE SEXIEST USER INTERFACE EVER

Users now have even greater flexibility and the ability to decide exactly how their software will look.

Use Sea-Change V2.0 to fu needs.

Sea-Change Version 2.0 enables you to control and alter the appearance of your application and the style of interaction. The colours of the different parts of a form can be altered and alternatives specified for non-colour terminals. In addition, it is now possible to tailor the action of the function keys to include box-drawing on forms, titled boxes, and overlapping forms and menus.

SEA-CHANGE 2.0

SIMPLY AHEAD OF THE TIMES

For the full story simply fill in this coupon and return it to Thomson Computers. Today!

NAME & TITLE

COMPANY

ADDRESS

TEL:

THOMSON COMPUTERS 245

Marsters House, 43 Blossom Street, York YO2 2AQ. United Kingdom. Telephone: (0904) 611666.

THOMSON COMPUTERS

SEA-CHANGE