The International apple computer users' magazine

Windfall

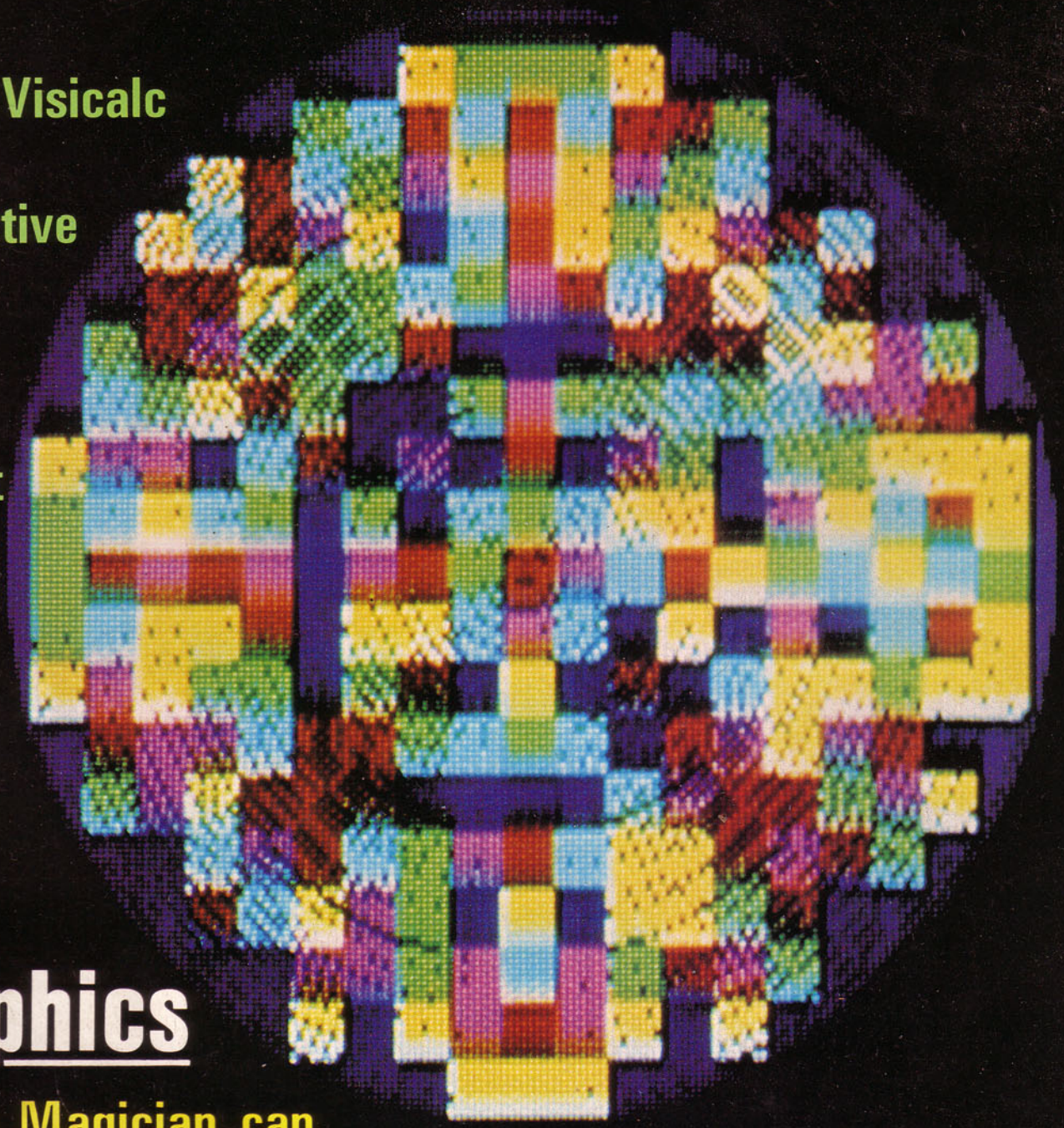
Vol. 3. No. 5 November 1983 £1

Visual Visicalc

Interactive video

New look at Logo

CP/M card review

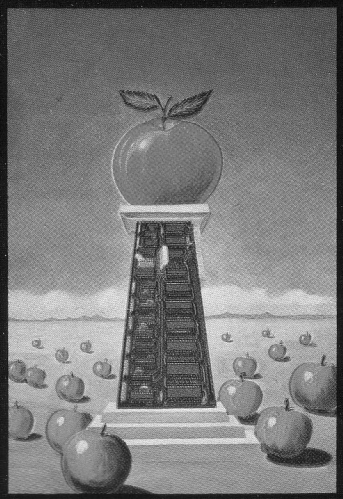


Graphics

How a Magician can transform your image

Come the resolution: Hi-res and Super-res

OUR CONTRIBUTION TO THE APPLE SCENE



CARDS

RAMCARDS

– with free software.

U-RAM16+ equivalent to Apple language card includes DOS mover. **£65.00**

SPECIAL BARGAIN **U-RAM32**+ versatile 32K card (includes DOS mover, VVE and VRD). **£50.00**

U-RAM64+e and **U-RAM128**+e big value, big memory cards including free DOS mover, disc emulation (VRD) and Visicalc expand software. (VVE).
64K **£180.00**
128K **£275.00**

NEW **VVE80 11e** Visicalc expand software for the 11e 80 column board and our U-RAM64 or U-RAM 128. **£30.00**

INTERFACES

NEW **U-PRINT 16**+e serial centronics, buffer and graphics printer interface. **£129.00**

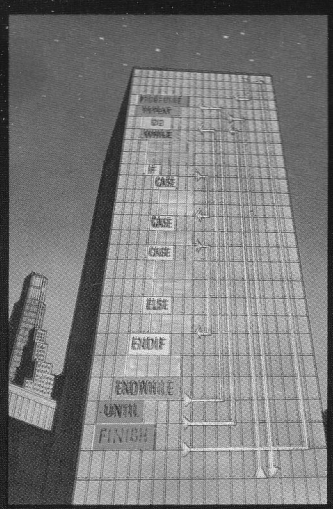
NEW **U-PRINT 64**+e version with 64K buffer. **£229.00**

U-PRINTCAB Centronics cable for U-PRINT. **£15.00**

U-S232+e /// serial interface. **£75.00**

U-PORT+e /// 8 serial interface on one card. **£195.00**

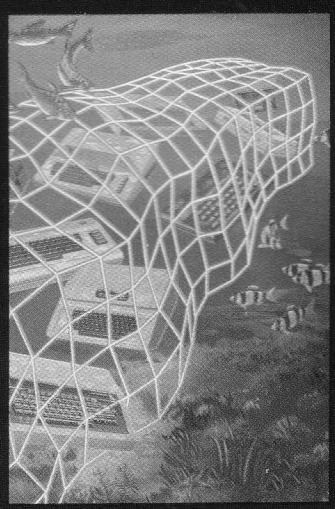
NEW **U-A/D**+e /// 8/16 channel, 12 bit A/D, software controlled gain, 32 digital I/O channel and timer! **£375.00**



STRUCTURED BASIC+e

The package that adds structured BASIC commands to Applesoft.
It allows structured programming without the complexities and fussiness of PASCAL. **£90.00**

'the only package I've seen this year I'd buy for my own collection'
Personal Computer News. Jun 83



U-NET+e

The most sophisticated, powerful and easy to use network system for the Apple II+ or 11e. Up to 29 users share up to six disc drives, one or two printers and a clock. Distance from host to satellite 100m with longer distances possible via modems. Detailed information on request.

Prices exclude VAT and delivery.
12 months warranty with all items.

CARDS

Interfaces continued –

U-DT+e /// digital I/O and timer card. **£105.00**

NEW **U-CENT**+e /// Centronics interface. **£69.00**

NEW **U-4DISC**+e disc controller (4 drives). **£85.00**

ENHANCEMENTS

U-Z80+e processor card for CP/M. **£95.00**

SPECIAL BARGAIN **U-TIM**+e precise timer. **£45.00**

U-TERM+ 80 col. display. **£150.00**

U-M68000+e 32 bit 68000 processor card. **£469.00**

NEW **U-TALK**+e speech synthesis. **£85.00**

AIDS

U-EXT+e /// slot extender. **£9.50**

U-PROT+e /// prototype card. **£12.00**

U-NSCRUMP+e /// software protection card. **POA**

MOTHERBOARD

The **U-COM2** is a fully compatible 8 slot, 64K RAM motherboard selling in increasing quantities to OEM's and System builders.
Big quantity discounts!
Details on request.

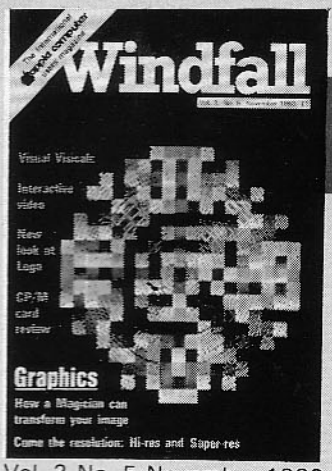
U-MICROCOMPUTERS

U-Microcomputers Ltd
Winstanley Industrial Estate,
Long Lane, Warrington,
Cheshire WA2 8PR
Telephone 0925 54117
Telex 629279 UMICRO G

USA Subsidiary –
U-Microcomputers Inc.,
300 Broad Street, Stamford,
Connecticut 06901, USA.
Telephone 203 359 4236
Telex 965999 O&S STD

KEY
+ compatible with II+
e compatible with IIe
/// Apple /// version available with SOS driver. Price on request

ORDER FROM YOUR DEALER NOW!



Vol. 3 No. 5 November 1983

Managing Editor
Derek Meakin

Production Editor
Peter Glover

Features Editor
David Creasey

Technical Editors
Cliff McKnight
Max Parrott

Advertisement Manager
John Riding

Advertising Sales
John Snowden
Peter Nowell

Marketing Manager
Linda Dobson

Editor-in-Chief of Database Publications
Peter F. Brameld

Tel: 061-456 8383 (Editorial)
061-456 8500 (Advertising)
Telex: 667664 SHARET G

Published by:
Database Publications Ltd, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.

Subscription rates for 12 issues, post free:

£12 – UK
£13 – Eire (IR £16)
£18 – Europe
£15 – USA (surface)
£25 – USA (airmail)
£15 – Rest of world (surface)
£30 – Rest of world (airmail)

Trade distribution in UK and Ireland by Wells, Gardner, Darton & Co Ltd, Faygate, Horsham, West Sussex RH12 4SU. Tel: Faygate 444.

Writing for Windfall: Articles and programs relating to the Apple are welcome. Articles should preferably be typed or computer-printed, using double spacing. Unsolicited manuscripts, discs, etc, should be accompanied by a self addressed stamped envelope, otherwise their return cannot be guaranteed. Unless otherwise agreed, material is accepted on an all rights basis.

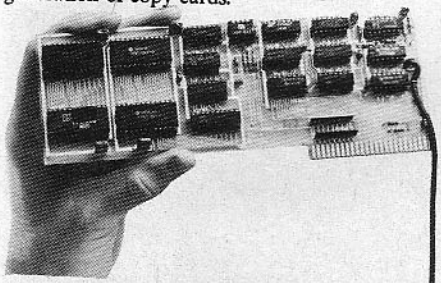
LISTING

Accidents will happen...

Relax, you've got the Snapshot Copykit

We've taken the worry out of computer use. A damaged disk no longer means weeks of waiting for a costly replacement, because you can now backup your important software with the Snapshot Copykit.

Once again, Dark Star Systems have led the way with the introduction of the new generation of copy cards.



IT'S NEVER OUT OF DATE – The Snapshot Copykit exactly duplicates every byte of memory-resident programs up to 128k.

Other copy cards disturb several bytes of memory. Sophisticated copy protection schemes can detect these changes and make the resulting backup unusable.

Because all its software is in RAM rather than inflexible ROM, the system enhancements we develop can be made available to you at nominal cost.

A full 8k of on-board memory will make the Snapshot Copykit particularly expandable for future use as a printer buffer, a multi-tasking system, a communications dispatcher and much more.

IT'S COMPATIBLE – The Snapshot Copykit is a board for all systems. It sits in any slot on the Apple IIe and II+ as well as the Basis 108, the Franklin Ace and other "look-alikes".

The Snapshot Copykit doesn't need a language/memory card and won't interfere with any of your other peripherals. So, you never need to remove it after installation.

IT'S USER-FRIENDLY – We have a well-deserved reputation for providing easy to use products. The Snapshot Copykit is no exception.

Other copy cards force you to use complex, unreliable procedures and repeated booting of "utility disks" to copy larger programs.

Old fashioned "nibble-copier" programs involve endless trial-and-error parameter changes before they will work.

The Snapshot Copykit will backup memory-resident programs up to 128k with one simple menu and one press of the trigger – in seconds!

IT'S POWERFUL – The Snapshot Copykit gives you back the power over your computer that copy-protected software took away.

You can interrupt a running program, copy it, list it, disassemble it, step and trace it, modify it and resume running it. Great for debugging and customizing!

The program is not disturbed in any way, even if you suspend it for a while to run another!

All backups made with the Snapshot Copykit are automatically BRUNnable files which you can transfer to hard disk. Its state-of-the-art compression facility lets you stack several backups on one floppy.

IT'S GUARANTEED – All our products are covered by a 90-day, no-quibble guarantee for defective parts.

We also support you with free technical advice – all owners of the Snapshot Copykit receive our consultancy hot-line number.

SYSTEM REQUIREMENTS

SNAPSHOT IIe VERSION:
Copies memory-resident programs up to 128k.
Apple II, II+, IIe, Basis 108, Franklin Ace or other lookalike with disk drive.

PRICE: £99.00 + VAT.

SNAPSHOT II VERSION:
Copies memory-resident programs up to 48k.
Apple II, II+ (RAMcard required – please specify brand when ordering), Basis 108, Franklin Ace or other lookalike with disk drive.

PRICE: £85.00 + VAT.

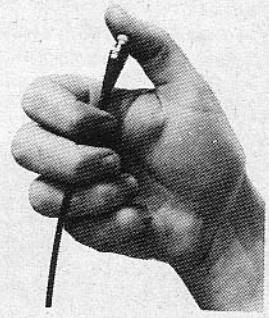
Terms: Payment with order – p&p included in purchase price. VISA & AMERICAN EXPRESS accepted.

Foreign orders: No VAT – add £2.00 postage in Europe, £7.00 elsewhere.

darkStar SYSTEMS

78 Robin Hood Way, GREENFORD,
Middlesex, UB6 7QW
Telephone: 01-900 0104
Telex: 8813271GECOMS G

The Snapshot Copykit is the "ultimate unlock system" for programmers, business users and hobbyists. Find out more by calling or writing for our info packet and the latest news on software updates and other new products.



The platform for anyone wishing to agree with, improve, disprove or generally discuss specific articles in *Windfall.*

FASTER WITH BLOCKWRITE

I WAS interested to read the technique of saving and loading Pascal hi-res pictures to disc, as explained by J.P. Lewis in the June issue of *Windfall.* writes **D. Hart.**

The saving and loading can be speeded up by using the Apple Pascal functions BLOCKWRITE and BLOCKREAD. These transfer physical blocks between disc and memory and are explained in the Apple Pascal Language Reference Manual.

Using these functions will reduce loading and saving times by about two seconds. The listings on the right are of the programs used by J.P. Lewis modified for BLOCKWRITE and BLOCKREAD.

```
PROGRAM MAKEART;

USES TURTLEGRAPHICS;

TYPE
  PICTURE = PACKED ARRAY[0..8191] OF CHAR;

VAR
  POINTER : RECORD
              CASE BOOLEAN OF
                TRUE  : (WHERE : INTEGER);
                FALSE : (PAGE  : ^PICTURE)
              END;

  F : FILE;
  CH : CHAR;
  BLOCKS : INTEGER;

PROCEDURE DOODLE;

  BEGIN
    MOVETO(0,0);
    PENCOLOR(WHITE);
    MOVETO(0,180);
    MOVETO(60,180);
    MOVETO(0,0)
  END;

BEGIN
  WRITE('PRESS ANY KEY TO START');
  READ(CH);
  INITTURTLE;
  DOODLE;
  REWRITE(F,'GALLERY');
  POINTER.WHERE:=8192;
  BLOCKS:=BLOCKWRITE(F,POINTER.PAGE^,16);
  CLOSE(F,LOCK)
END.
```

Speedier data-logging

IN advocating the use of the stack in his Think Tank article of September 1983 J.P. Lewis certainly puts his finger on one of the fastest methods of data-logging in an unadorned Apple, **writes Doug Shaw.**

The overall implication that using the stack resulted in a 40 per cent increase in sampling rate was, however, rather exaggerated.

Whereas PHA does indeed show a 40 per cent timing improvement over STA BUFFER,X the overall loop improvement is only 13½ per cent thereby increasing the sampling frequency from 55.5kHz to 62.5kHz. This is still a long way from the desired 100 kHz sampling frequency.

A further improvement in sampling rate to just over 83kHz might be implemented by triggering the ADC automatically every 12 mu.sec. and synchronising this with a toggle working outside the sampling loop.

I tend to think that a more predictable method would be to use the acclaimed accelerator in conjunction with the stack method of data logging.

Together with an ADC offering 2 mu.sec. conversion time this would give a theoretical sampling frequency of around 250 kHz.

Mr Lewis' final comment concerned an embargo on the use of JSRs and RTSs during data logging to the stack.

The reason for this is that a JSR stores the program counter on the stack, decrementing the stack pointer twice in doing so. Thus a stack overflow could occur when the number of samples to be stored on the stack exceeds 254.

For less than this number and where the sampling routine is to be called from different locations in the main program it might well be expedient to call the routine by JSR to establish the calling location, execute a do-it-yourself RTS and terminate the subroutine with an indirect JMP to the return address.

An outline of a suitable routine is shown below.

```
1    Sample   CLC              Simulated RTS
2             PLA              'Pop' LSB of return address
3             ADC#1            Add 1 to LSB
4             STA RETADR       Save to 'Return address'
5             PLA              'Pop' MSB of return address
6             ADC#0            Add carry – if set
7             STA RETADR+1
      Sampling loop goes here.
20            JMP (RETADR) End of loop
```

```
PROGRAM SHOWART;

USES TURTLEGRAPHICS;

TYPE
  PICTURE = PACKED ARRAY[0..8191] OF CHAR;

VAR
  POINTER : RECORD
              CASE BOOLEAN OF
                TRUE  : (WHERE : INTEGER);
                FALSE : (PAGE  : ^PICTURE)
              END;

  F : FILE;
  CH : CHAR;
  BLOCKS : INTEGER;

BEGIN
  WRITE('PRESS ANY KEY TO START');
  READ(CH);
  INITTURTLE;
  RESET(F,'GALLERY');
  POINTER.WHERE:=8192;
  BLOCKS:=BLOCKREAD(F,POINTER.PAGE^,16);
  CLOSE(F)
END.
```

Another approach to input validity checking

HERE is another approach to input validity checking in Basic programs, writes **Alan Dubost.** The routine is designed to ensure that only valid characters can be entered into a field in a record and that a warning beep is sounded if the character entered is invalid or if the operator attempts to go beyond the left and right limits of the screen display field.

It is a substitute for the INPUT statement and uses one array element to store each character entered via the keyboard. Before placing that character into the array it is tested for validity.

When the RETURN key is sensed then the elements (characters) of the array are built up to form P$ which, in this example, is 20 characters long and left justified.

The P$ is the field of valid data of the correct form and length for our file record layout.

```
17200 G$ = CHR$ (7): REM  beep t
      he speaker warning
17210 T = 1: REM  character count
      er
17220  IF T = 0 THEN T = 1: PRINT
      G$: REM  cannot go beyond le
      ft limit of field on screen
17230  IF T > 20 THEN T = T - 1: PRINT
      G$: REM  cannot go beyond ri
      ght limit of field on screen

17240  VTAB 6: HTAB (T + 19): GET
      DX$(T): PRINT DX$(T);: REM
      get character and place in a
      rray
17250  IF  ASC (DX$(T)) = 13 THEN
       FOR S = 1 TO T - 1:P$ =  +
       REM  carriage return key se
      nsed so P$ is built up from
      the valid array characters a
      nd left justified
17260  IF  ASC (DX$(T)) = 8 THEN
      T = T - 1: GOTO 17220: REM
      left arrow key so reduce T
17270  IF  ASC (DX$(T)) = 21 THEN
      T = T + 1: GOTO 17220: REM
      right arrow key so increase
      T,Now test for name field va
      lid characters lines 17280-1
      7310
17280  IF  ASC (DX$(T)) = 46 THEN
      17310: REM  full stop
17290  IF  ASC (DX$(T)) = 45 THEN
      17310: REM  hyphen
17300  IF  ASC (DX$(T)) < 65 THEN
       PRINT G$: GOTO 17220: REM
      less than alpha range
17310  IF  ASC (DX$(T)) > 90 THEN
       PRINT G$: GOTO 17220: REM
      greater than alpha range
17320  IF T > 20 THEN  PRINT G$:T
      = T - 1
17330 T = T + 1: GOTO 17240
17340  IF  LEN (P$) > 20 THEN P$ =
      LEFT$ (P$,20): REM  chop P$
       down to 20 characters
17350  IF  LEN (P$) < 20 THEN P$ =
      P$ + "": GOTO 17350: REM  bu
      ild up P$ and left justify
17360  RETURN
```

Date verifying program

AFTER a rather short battle with the Apple II I devised a suitable date verifying program that I feel is adequately secure in its particular field, writes **C. Geraghty of Durban.**

It can be incorporated into any business related program requiring an input for a specific date.

Contrary to the published listings of the Date Conversion and Fickle Fingerproofing programs by Geoff Stratton and J.P. Lewis respectively, I feel that my particular program is both far easier to understand and to input/enter although it may lack in some more specific, intricate, and precise spheres.

```
10  REM ********************
20  REM *                  *
30  REM *  DATE VERIFIER   *
40  REM *                  *
50  REM *   C. GERACHTY    *
60  REM *                  *
70  REM *    JULY 1983     *
80  REM *                  *
90  REM ********************
95  REM
100  HOME : NOTRACE
110  VTAB 10: INPUT "ENTER CURREN
     T DATE (DY/MN/YR).";DA$
120  IF  LEN (DA$) <  > 8 THEN  GOTO
      100
130  IF  MID$ (DA$,3,1) <  > "/" AND
       MID$ (DA$,3,1) <  > "." OR
       MID$ (DA$,6,1) <  > "/" AND
       MID$ (DA$,6,1) <  > "." THEN
       GOTO 100
140  IF  VAL ( MID$ (DA$,1,2)) >
     31 OR  VAL ( MID$ (DA$,1,2))
      < 1 THEN  GOTO 100
150  IF  VAL ( MID$ (DA$,4,2)) >
     12 OR  VAL ( MID$ (DA$,4,2))
      < 1 THEN  GOTO 100
160  IF  VAL ( MID$ (DA$,7,2)) >
     99 OR  VAL ( MID$ (DA$,7,2))
      < 70 THEN  GOTO 100
170  IF  VAL ( MID$ (DA$,4,2)) =
     0 AND  VAL ( MID$ (DA$,1,2))
      > 29 THEN  GOTO 100
180  READ OD
190  IF  VAL ( MID$ (DA$,4,2)) =
     OD AND  VAL ( MID$ (DA$,1,2)
     ) > 30 THEN  GOTO 100
200  DATA 4,6,9,11
210  RESTORE : CALL  - 936: VTAB
     10: HTAB 15: INVERSE : PRINT
     "VALID";: NORMAL : PRINT " D
     ATE."
220  VTAB 12: HTAB 15: PRINT "BRE
     AK WITH CTRL-C.": FOR I = 1 TO
     3000: NEXT I: GOTO 100
```

In space your Apple can't hear you scream!

Edited by CLIFF McKNIGHT

ANYBODY could have heard me scream in space when I formed my first impressions of Avalon Hill's new strategy game, The Alien. I was at the time, however, in ignorance of most of the game's good points, so don't let this put you off.

As you have probably realised, The Alien is based on the sci-fi horror movie "Alien". The game follows the same plot, with the panicked crew of a spaceship looking to kill or capture an escaped alien on board ship.

The alien poses greater and greater problems as time goes on, as it grows into bigger and more death-resistant forms. To make things worse, an unknown member of the crew is an android, determined to preserve the alien's life so that it may be studied in the future.

There are two significant additions to the Avalon Hill scenario. Firstly, the alien is able to clone, so before long you are faced with a whole bunch of the horrors.

Secondly, in the initial confusion several lab animals escaped. It is very important to recover them, since the ship's sensors can't tell them apart from the aliens. A neglectful player will soon find out that the aliens are very good at mixing in with the crowd.

At your disposal are seven crew members including two officers, two scientists and three engineers. Each class of crewman can do things that the others can't. Only the engineers, for example, can build weapons, and only the officers and scientists are likely to succeed in the computer room.

The stubborn computer will sometimes reveal useful information about the alien, such as the fact that . . . ah, but that would be telling.

If you feel sorry for the poor little aliens you can merely try to capture and secure them (this is not advisable when they get bigger, since they tend not to fit into the cages!). Electric prods, cages and sleep dart pistols are useful for this. Cages won't hold them for long, so they must be stored in a secure room or a lab cage.

For the more violent or vengeful player, lasers, gas canisters and flamethrowers are available. It is also possible to eject the alien into space via one of the airlocks. No method of destruction is infallible, and even a laser can fail against a fully developed alien.

Alien lovers and pacifists can do little but set the ship to self destruct and flee in the escape shuttles. Be careful though, aliens have an uncanny ability to sense when the self destruct mechanism is active!

Each turn of the game involves five phases. They are always called up in the same order, starting with the interrupt phase, followed by the non-crew movement phase, the first action phase, the crew movement phase and the second action phase.

All input during the game is via paddle(0) and is a little slow. If a key is hit during the play, the game will stop during the interrupt phase. The player may then select between quitting, continuing, saving the game, or restoring an old game.

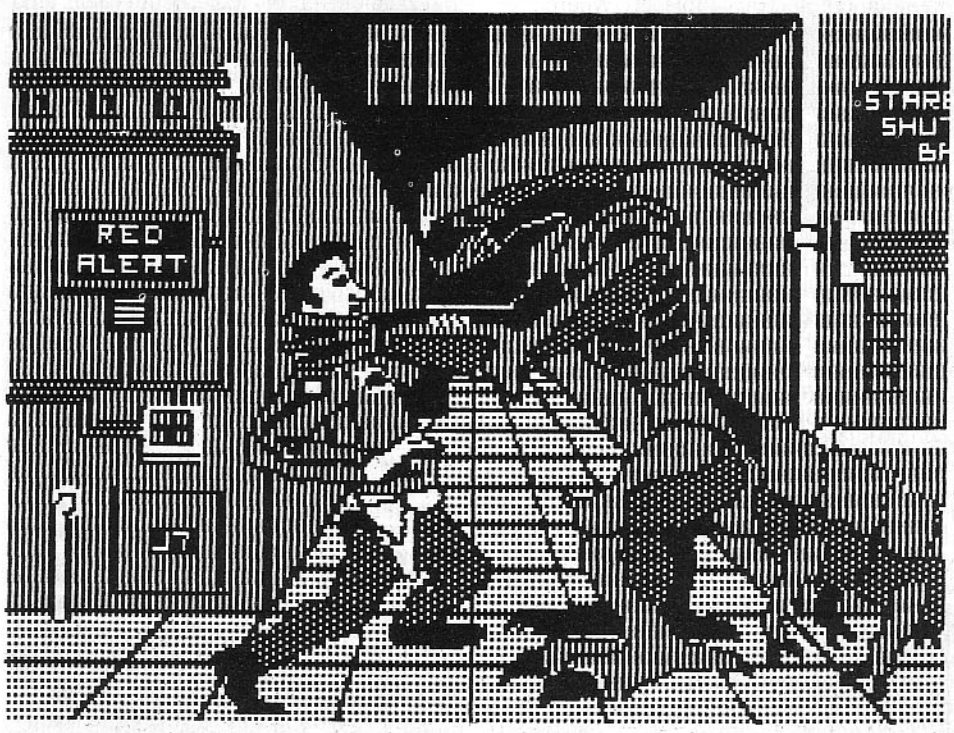
When an input is required, the player must turn the paddle back and forth searching for the option he desires – which is selected by the paddle button.

This form of input is very frustrating at first and requires some heavy paddle work. In the movement stage of the game the player has to look through a list of 30 room names to find the correct one. Most of these rooms are inaccessible from the player's position anyway!

During the action phases the computer switches to text mode and steps through each room occupied by crew members. The room contents are listed and the player may select to give the crew member(s) in the room one command each.

The commands available are ATTACK ALIEN, PROD ALIEN, PUT IN CAGE, CATCH ANIMAL, OPEN AIRLOCK, OPERATE COMPUTER, LOCK IN LABCAGE, INVENTORIZE (take a log of the animals caught), LEAVE SHIP and SELF DESTRUCT. There is also a NOTHING option if you selected the command mode by mistake.

During the crew movement phase each surviving member of the crew is allotted three action points. All, some or none of these may be used. Different actions take up a different number of points, but one action per point is usual. The options are MOVE, TAKE, DROP and CONSTRUCT (flamethrowers, etc.) All entail an obvious



The title page from Alien

CLIFF'S COLUMN

NOW, where was I? Oh, yes, I remember. It's all coming back to me now – must be the garlic. Sorry about the temporary lack of Cliff's Column last month – we suffered an editorial brown-out as a result of attempting to generate a multi-tasking environment. Well, it sounds good, doesn t it?

Last time I was talking about little tricks you may have discovered which weren't mentioned in the game instructions. For example, when I reviewed Bug Attack I complained about the movement keys. However I have now been informed that the two arrow keys also work perfectly well. Shame on me for not trying them, I'll slap myself on the wrist for punishment.

For a time Choplifter was extremely popular, so presumably many of you have played it. I remember trying to play with paddles before I got my excellent Kraft joystick. (Ignore that last plug if you're left-handed.) If you consigned your copy to the drawer because you were fed up with being shot down, you might like to try the following ways of cheating.

If you enter CTRL-L followed by a zero, only one tank attacks you. Replacing the zero with a 1 gives two tanks and jets. Using 2 gives two tanks and jets and two drones, and a 3 gives you two tanks and jets and two drones that fire.

Cannonball Blitz never really took off like Choplifter (groan, another slap on the wrist) but if you have a copy you might like to know how to survive a little longer. When you finish Level 1, hold down the SPACE bar and REPEAT key until the screen changes to the next level. You'll then find only two cannons.

Of course, it's also possible to cheat with adventure games. For example, The Prisoner is listable, but you'd need a lot of patience to plough through the listing.

Most adventure games are protected from view, but if you have a utility package like the CIA Files, you may be able to view sections of the code. However, this is even more tedious than studying a listing.

At least one major adventure game I know has not trapped RESET adequately, so that using it after you have been killed off allows you to continue past the obstacle.

Finally there are the commercial cheat packages like Wizplus from Datamost which allows you to build super-characters in Wizardry. Sir-Tech, the manufacturers of Wizardry, say that these products tend to interfere with the "subtle balance" of the game and that to use them "would be akin to playing chess with additional queens". For some of us though, it's the only way we'll ever get to see Knight of Diamonds.

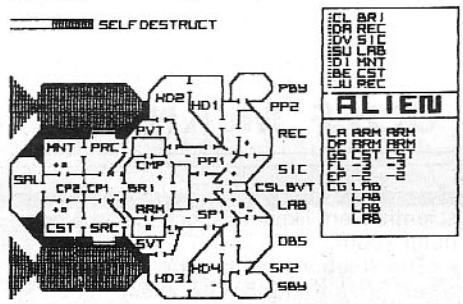
Some people play to win, so cheating the machine is perfectly OK. Others play for the sake of the game, and so would never dream of cheating. Others use euphemisms like "advanced playing techniques" to disguise the fact that they're cheating. Me, I'm into wrist slapping . . .

qualifying input.

A graphics display is used during the movement phase showing an annotated birds-eye view of the ship. Crew members are shown as little '+' signs and objects as little squares.

Aliens and animals are only shown if a crewmember is on the bridge to operate the sensors. They both share the same '–' sign, which adds problems to tracking the alien.

The possessions of crewmembers and other useful information are also shown on the screen, along with a self destruct countdown. The graphics in this phase, although carefully done, aren't very inspiring and a little daunting at first. The title page, however, is superb.



Map of the spaceship in Alien

The booklet provided is easy to understand and contains amusing and useful information about the animals, rooms and crewmembers. My biggest mistake was to rush into the game before reading the manual. Doing so deprives the player of a full knowledge of all the options available.

There are three things that the booklet doesn't mention. Firstly, CTRL-C may be used at any time to restart the game. Secondly, RESET can be used to quit the game (and enter into Applesoft, where the program may be listed). Lastly, scientists are only allowed to LOCK IN LABCAGE or INVENTORIZE once each turn (there are two action phases per turn). I think the last point might be a program error, not an omission.

There is no scoring as such, and the amount of enjoyment gained from the game is up to the player. It's easy to self destruct the ship and flee right at the beginning, but no fun.

I enjoy forming the crewmembers into strategical groups, waiting for the alien to clone, and then trying to use as many ways of getting rid of them as possible.

It is also much more fun to catch the aliens than to kill them, and the game congratulates you for doing so.

This is not a game for all-out arcade players, but adventure fans should get a fair amount of enjoyment out of it. If you buy it happy hunting, and remember – in space your Apple can't hear you scream!

Julian Brewer

Title: The Alien
Author: Hans von Alteren
Publisher: Avalon Hill
Requirements: Apple II or IIe with games paddles.

Colonial challenge

IF you have delusions of grandeur, there are several things you could do. One is to become Editor of *Windfall,* but if you fancy a bit more of a challenge, you could try playing New World, from Epyx.

In it you get an opportunity to play the role of the monarch of Spain, England or France in conquering and exploiting the assets of North and South America.

Depending on the vagaries of political and social life at home and abroad, you may get the chance to rob the natives of their gold and watch the coffers of the Old World fill up with doubloons.

As the manual explains after bringing your history up to scratch, the game is played in turns, each of which represents five years. Play starts in 1495 and ends in 1600, so each game involves 22 turns.

From one to three people can play. If you choose the solo option your friendly Apple will keep you company by cheating and winning! At least mine did.

A two player game involves Spain and France, with the monarch of Spain playing first. With a single player, you are monarch of Spain with apparently first play. However, a quick colony status check reveals that France has already slipped in and bagged the high productivity area of New Mexico.

Not only did the Apple get New Mexico, it also managed to transport more colonists there in one sneaky turn than I had available for at least seven!

While you can wheel and deal by shipping colonists (if you recruit any) and soldiers (if you can afford them) here and there, the basic strategy seems to be to load colonists and soldiers into areas of high productivity and watch the money roll in. In passing, you can transfer a colonist or two to areas of low productivity to dig for gold.

While mainly text-based, New World has some graphics to break the monotony. The colonists dig for gold very energetically – mine never found any – and your ship sails across the screen from east to west with a little flag-bearer to claim the land when you reach it. The maps for North or South America can also be called up, although they take a little time to arrive.

All this may sound quite interesting, particularly if you yearn for the chance to make a quick doubloon, but unfortunately the game has been so poorly implemented as to seriously detract from potential enjoyment.

The problem is that roughly half the inputs from the players use a GET format while the remainder use an INPUT one, the latter requiring a RETURN. Pressing RETURN by accident when it is not needed ends the game.

Before writing this review I spent more than two hours playing the game again to

try to give it the benefit of the doubt. I was determined to watch the GETs and RETURNs and at least find out what happened when you reach the year 1600.

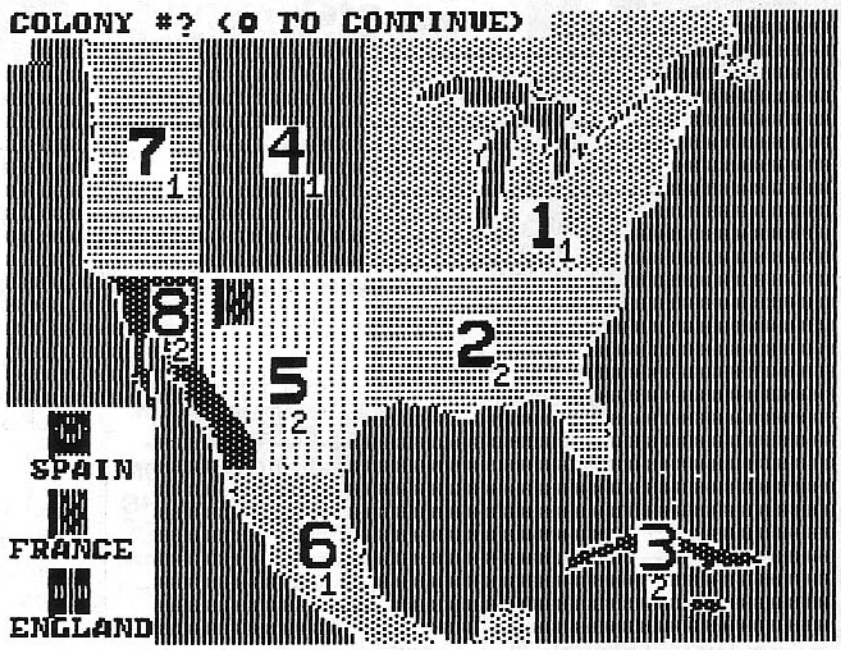
I'd reached 1590 and was just feeling as though I was getting somewhere when I hit RETURN by mistake in the heat of the moment.

I realised my mistake immediately, but there was no way back. Already I was being offered another game. I hadn't saved the game (well, I *was* losing) so all the time was wasted. Given that the manual lists four play-testers, I'm surprised that none of them discovered the same problem.

The Epyx boast is that theirs are "computer games thinkers play". I'd rather expend my limited brain capacity playing the game than remembering whether or not to press RETURN.

Denise McKnight

Title: New World
Authors: D.A. Decker and Steve Bryson
Publisher: Epyx
Requirements: Apple II



North America awaits explorers in New World

Crime time

'ULLO, 'ullo, 'ullo, what's all this 'ere. A bank robbery, eh? We'll soon put a stop to this little game . . . oh, it is a game, and from Penguin Software, too. This deserves further investigation.

Your job in Crime Wave is to patrol your patch in your squad car, trying to protect the numerous banks which are scattered about.

As you cruise around the streets, some of them one way, you encounter some other traffic and must use your "shield" to pass them. Failure to do so results in the usual one-way trip to the mortuary.

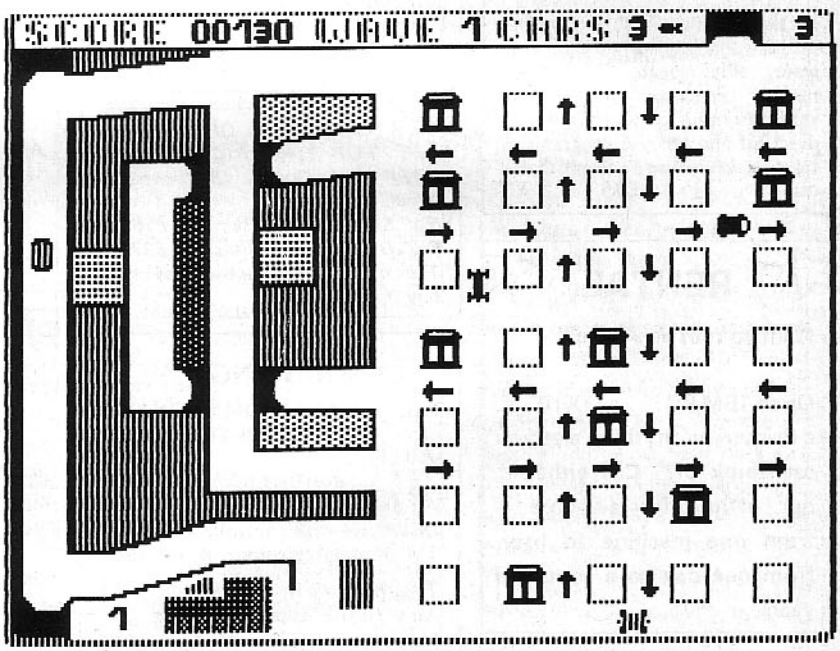
Some of the automotive citizens are not as innocent as they may seem. Eventually, they pull up outside the bank, which then starts to flash, indicating an attempted withdrawal in the absence of an account.

Conveniently, getaway cars become orange, thereby distinguishing them from the whiter-than-white innocent citizens.

The chase is now on, and you must catch the crook by crashing into him – real Starsky and Hutch stuff, this – and tow him to gaol.

If you can crash the crook's car while the robbery is in progress, the scene/screen changes. You now see the crook fleeing the bank and you must try to catch him in a cage. Failure to do so means that you must resort to the chase.

Successful robbers don't simply retire to South America and sell their story to the Daily Gossip. They use their ill-gotten gains to buy a Robot Rammer and then they come looking for you, so it pays to catch them quickly. You have a single bomb which you can lay in the path of a Robot Rammer, but you can only get another if you use it successfully.



The one-way streets in Crime Wave

Gaoling three robbers completes Level 1; another three are required for Level 2, and thereafter five are required per level. Points are awarded for how quickly you catch them and how fast you get them to gaol. Bonus points are awarded for the number of banks remaining unrobbed when you complete the level, and an extra car/life is also awarded at this time.

The manual doesn't mention it, but the screen menu lists an option to change the speed. The options are normal, quick, fast, faster and suicidal. The game defaults to fast, which is not too difficult to master. After this, normal seems hearse-like, but suicidal is aptly named.

There's plenty to keep you busy once you get into the game. Using the shield carefully, positioning the bombs and luring the robot rammer onto them, avoiding innocent citizens, trying to intercept the supply truck, catching robbers and coping with the one-way systems didn't leave me many spare brain cells.

Crime Wave is a good game with the added advantage that Penguin cut all their prices earlier in the year, so at least the price isn't daylight robbery, even if the game is!

Cliff McKnight

Title: Crime Wave
Author: Scott Schram
Publisher: Penguin Software
Requirements: None stated

Neat main frame style listings from your Apple

By J.P. LEWIS

ONE of the nice little features on mainframes is the way that the listings of programs that you request are sent to you with a huge banner page telling you who you are, what the program is called, when you spooled it off etc. The other feature of these listings is the way that they are tidily split into pages, where each page has the title printed at the top and a few blank lines top and bottom.

Although I haven't yet got around to doing the banner page, I have written a little Pascal program that does the paging and inserts the title (see Listing I, which is the program, and a demonstration of its own output).

The method is quite simple. The initial parts of the program merely open the input file (any text file), and the output file (probably '£6:' or 'PRINTER:'), and get the title line from the keyboard.

Note that there is no error trapping in the program, so it is the responsibility of the user to ensure that he enters legal filenames, and a title that is no more than 70 characters long. Failing in the former will cause an I/O ERROR, the latter will cause a STRING TOO LONG ERROR.

The main loop in the program prints a gap at the top of the current page, prints the title with page count and prints a few more blank lines.

Then, allowing for the number of lines already used, it starts counting its way down the page, reading and printing a line at a time until one of two things happens – either there are no more lines to read, in which case it throws a page clear and stops, or the point has come to leave a few blanks at the bottom of the page, in which case it throws a page, and goes back to repeat the process.

Although the program may be criticised on several points of style, one detail I would like to commend to you is the use of the CONST declarations.

The basic function of the program is to reformat the normal output of a Pascal listing. However, since all the important items, that is the ones which can change the resulting output are declared at the very top as constants, it takes virtually no time to correct the program if you change your formatting requirements.

To produce Listing I, run the programs with PAGELENGTH set to 38. This is a very important point to remember in writing Pascal programs – it is possible to isolate the key features of a task, so that a minimal amount of work can be spent in adapting one program to suit many situations.

```
program listing;
const
    TOPGAP=2;
    BOTTOMGAP=3;
    TITLEGAP=2;
    PAGELENGTH=66;
var
    infile,outfile:file of char;
    title,line,filename:string;
    pagecount,count:integer;

procedure skip(gap:integer);
begin
    for count:=1 to gap do
        writeln(outfile)
end;

procedure gettitle(var title:string);
begin
    write('Enter the program title ');
    readln(title);
    title:=concat('(*  ',title, '  --page-- ')
end;

procedure getname(whichfile:string; var filename:string);
begin
    write(whichfile,'Input a filename ');
    readln(filename)
end;
```

Listing I

```
begin
    page(output);
    getname('For input:   ',filename);
    reset(infile,filename);
    getname('For output:  ',filename);
    rewrite(outfile,filename);
    gettitle(title);
    page(output);
    pagecount:=0;
    repeat
        pagecount:=pagecount+1;
        skip(TOPGAP);
        writeln(outfile,title,pagecount,'  *)');
        skip(TITLEGAP);
        count:=TOPGAP+TITLEGAP+1;
        repeat
            readln(infile,line);
            writeln(outfile,line);
            count:=count+1
        until((eof(infile)) or (count=PAGELENGTH-BOTTOMGAP));
        page(outfile)
    until (eof(infile));
    close (outfile,lock)
end.
```

Listing II

Express route to learning is via the Apple

Apples have been chosen by American Express for its newly opened Computer Learning Centre at the company's European Headquarters in Brighton.

Three complete Apple learning stations have been installed, each with a Sony U-Matic videocassette player interfaced directly to the computer, and with the Apples networked to a single Symbfile hard disc unit.

An Apple is also used to tell Amex employees what's available in the Centre and how to use the equipment. A specially written catalogue system is operated by employees using a Gibson light pen.

Finally, to help its trainers to write courses for the centre, Amex is one of the first companies to install the Design of Learning package from IVL Learning Systems. Its experience with interactive video is described here.

By CLIVE SHEPHERD

FOR many years the training film has been a standard component for training courses within industry and commerce. The most successful ones have been glossily produced and entertaining, and provide a welcome break for (or from) the course instructor, particularly in the deadly after-lunch period.

However trainers are aware that, entertaining or not, such films do not seem to teach anybody very much. How much do YOU remember of the programmes you saw on TV two or three days ago?

Meanwhile, the more sophisticated training departments have been working for the last few years with computer-based systems, originally using large mainframes, but more recently with micros.

Quite a lot of progress has been made in this area, more because of good authoring software than anything else. However, it is still extremely difficult to produce convincing animated video and realistic audio from computers, even with the best graphics software, speech synthesisers and the lot.

Trainers just don't have the time necessary or the skills to generate pictures that look "something like" the original. Video (or photographs for that matter) does the job so much more easily.

Sooner or later someone had to get video together with computing to produce a more satisfactory solution. Unfortunately, the training, TV and computing industries are all infamous for their buzzwords, so we ended up with Interactive Video.

In simple terms, IV can be described as any technique whereby users have control over what they see on video, and when. This control is accomplished using random-access videocassette or videodisc players hooked up to a microprocessor controlled device – a unit built into the player, a handset or, in its most sophisticated form, a computer.

The video adds moving pictures and sound to the computer. The computer adds random accessing, selection, testing and management facilities to video.

As the Apple II has always been the most popular machine in the training and

INTERACTIVE VIDEO

Using the Gibson light pen

audio-visual industries, particularly in the USA, most early systems were developed for it. In fact, Apple still has a major hold on a technology which could well revolutionise learning and many other aspects of communication.

So what of American Express? Our interest in interactive video was inevitable – our training and video departments were already integrated, and using Apples for a wide variety of purposes.

The company had installed a worldwide network of random-access VHS players that were tailor-made for the job. Most importantly, we have the need for interactive video to train up a decentralised, geographically-spread workforce that operates in an environment where technology, products and organisation structure must, by necessity, change rapidly.

Interactive video constitutes the ideal medium for individual, self-paced instruction, that can be delivered without tutors, in multiple locations.

A number of interface cards are available to provide the necessary control over the videoplayer. The most common interface is the BCD450, from BCD Associates of Oklahoma, which will work with a variety of VHS, Betamax and U-Matic players.

The card switches between VCR and computer video, controls tape transport functions, and reads frame pulses from the VCR in order to determine exact tape positioning. Similar devices can be purchased to interface with videodisc players and, in fact, we have successfully operated a Phillips Laservision player under Apple control, for experimental purposes.

The only other requirement is a colour monitor and the usual disc drives and colour card.

When we first looked at IV for the Apple, the software available to enable trainers to create courses was minimal, and generally poor. Recently the situation has improved.

If your trainers are capable of actually coding programs, and require the flexibility that this provides, then Apple's own SuperPilot is probably the best bet. Built into the package is the facility to interface with popular cards such as the BCD, and the cost is, to say the least, reasonable.

For general purpose, self-instructional training however, we required something that would enable a non-programmer to put together courses quickly and with the minimum of hassle and frustration.

We wanted a menu-driven, highly prompted package at a reasonable price. In the end we commissioned a local consultancy, IVL Learning Systems, to produce the software for us. The end result, the IVL Authoring System, has since been enhanced and packaged for general commercial distribution.

Courseware is the term coined by training and computing people to avoid the confusion between the computer program and the training programme. Courseware is the finished lesson software.

Amex has produced a variety of Courseware to date:

- An "Introduction to Interactive Video" for managers and users.
- A management simulation called "Styles of Leadership".
- A number of training packages on operations that are specific to American Express.

Courses are not difficult to produce, given the right authoring software. Video, on the other hand, can be expensive if you do not have access to a suitable, ready-made tape. In addition, if you do produce special videos then your TV people must learn to think "interactive".

The programme should break down into short, self-contained scenes. Editing must allow for sensible sequencing of scenes so that users do not have to wait while the tape shuttles from one end to another.

All-in-all though, interactive video has proved itself to be popular with trainees, efficient in terms of learning, and, given a sensible choice of subject matter and treatment, a cost-effective solution.

Interactive video systems will be commonplace in medium size and large companies within two or three years, as training managers become less nervous and better informed about this new technology.

In the educational world, progress may be slower, bearing in mind the capital outlay required in hardware.

Some of the more exciting applications will come in point-of-sale displays, video juke-boxes, information retrieval systems and, in time, the home. When the home computer and the domestic VCR can be successfully integrated, the opportunities for home learning will be enormous.

At Amex we have already installed a Computer Learning Centre in Brighton, with three Apple systems, networked to a hard disc. To tell users what courses are available and how to operate the equipment, we have installed another Apple system operated by light pen. Only the screen is seen by the user.

Next year we will be aiming to produce our first videodisc. Although we have found tape to be a perfectly satisfactory medium, disc does offer the benefits of immediate access to scenes, improved picture and audio quality, and greater control over the picture speed and direction.

In addition, a disc can hold approximately 50,000 individual picture frames on each side. Remember, an Apple disc can hold only 14 binary graphic files!

My advice to you, if you are considering a venture into interactive video, is simple. As a *Windfall* reader, you probably already own an Apple. Get yourself a suitable VCR and interface card, and ideally an authoring language or system.

Keep the cost down by experimenting with any existing video programmes you have on the shelf. The financial outlay will be more than offset by the experience you will gain with the medium.

And if your research indicates a wide variety of potential applications within your organisation, you can proceed with confidence.

* *Clive Shepherd is Director of Training and Creative Services at American Express' European headquarters in Brighton.*

CP/M is one of the most widely used operating systems, particularly in the business area. Any improvement in its performance or ease of use will benefit a wide range of readers. The CP/M Card from Advanced Logic Systems offers a Z80 card together with the new CP/M Plus operating system and other utilities all in one package for £300.

Several commentators agree that it has the potential to be an excellent product. But as TOM DERWENT says in this article, some people have reported some early teething problems that might trouble an inexperienced user. An Apple dealer in Germany had similar difficulties. We publish his views here, together with a reply from the manufacturers.

A versatile CP/M Card – but take care

By TOM DERWENT

I WAS very pleasantly surprised when I fired up CP/M Plus. It contains all the usual CP/M utilities, some public domain software, such as CAT, but with a Digital Research copyright message and some new ones such as the symbolic instruction debugger, an upgraded version of DDT, a good Z80 macro assembler and a linker, among others.

The software comes on three discs, but only the first one contains a bootable system. The other two contain the utilities and the CBasic programming language.

Curiously, disc No 1 has a write-enable notch, and will not boot if it is covered up. The others do not have the notch and so are permanently write protected. It is important to copy these master discs, and file the originals in a safe place.

The manual is impressive, if not daunting. It is a two inch thick, cloth bound, book shaped ring binder containing three separate publications. The outer box can act as a storage container when placed on a bookshelf.

Luckily, it is not necessary to digest all of this material before using the CP/M Card. The manuals are the CP/M Card User's Guide, a 40 page offering by ALS; CP/M Plus (CP/M version 3) Operating System User's Guide ($\frac{1}{2}$in thick); and CBasic Language Reference Manual (slightly thinner), both from Digital Research.

The first 16 pages of the ALS book cover installation of the card and the fundamentals of the CP/M 3 operating system – concluding with a section relating some CP/M 3 commands to DOS equivalents – enough to get you started.

The rest of this book comprises a description of the utilities supplied (eight pages), a hardware description (four pages) for the technically minded, and an eight page glossary of terms which would be extremely useful for the novice.

The two manuals from Digital Research are much more in the reference manual line and as reference manuals go are quite good, with meaningful headings in the table of contents and lots of examples in red print, often showing exactly what would appear on the screen, plus indices.

However, one should realise that they are reference manuals and treat them accordingly. That is, they are excellent for looking up the required form of a command or CBasic statement, but you should not try to learn programming from them. It will probably be necessary for some users to buy one of the many CP/M operating system guides by independent authors.

Notwithstanding the above, the documentation provided should certainly prove sufficient for those users who merely wish to run pre-packaged programs and for enthusiasts who already have some computing knowledge.

It is essential after reading at least the first 16 pages of the ALS User's Guide to back up the discs using the relevant option on the start-up menu automatically presented when the system is booted.

This option takes you into the DSKCOPY utility which presents a further menu, including simple formatting, and copying of system tracks, as well as a whole disc copy option which will automatically format the destination disc.

However DSKCOPY always copies from drive A to B, so if you only have one drive you have a problem. You would have to obtain a copy program – either from public domain CP/M-80 software, buy a proprietary one, or else write one yourself.

In my view DSKCOPY is the most useful of the utilities provided. The others are:

- ADUMP – will give hexadecimal and Ascii dumps of files or sectors on a disc.
- MON 65 – permits programmers access to the Apple 6502 monitor.
- APPMAKER – creates discs containing the basic CP/M 3 system files, to which application packages may be added.
- HELLO – the program executed on startup to provide the initial "CP/M CARD" copyright notice and menu. It is not necessary to the system and one of its options is to remove the menu. In fact the HELLO.COM file may be removed and PROFILE.SUB modified or removed, to give additional free disc space.
- WSMAKER – a program which will automatically install Wordstar. This program is a victim of progress. I have successfully installed Wordstar 3.01P using this program, but the current one on sale is Wordstar 3.3, on which it will not work. In any case I do not fancy all the options chosen for me by this program and it cannot install specialist printers, only what MicroPro term "standard printer".

I recommend users to install their own, or have their local dealer install it for them. The UK distributors of the ALS CP/M Card, Scope Systems, on request provide a three page guide to be read in conjunction with the installation manual supplied with Wordstar. This does assume that the user will look up commands in the reference manual.

I have installed and used Wordstar using the ALS Smarterm II 80 column board, the Apple 80 column board, the Apple extended memory 80 column board, and a Vision-80 board, though I have not worked out how to get the inverse-video around the command option block as is done by WSMaker or WS 3.01P. Two other users have said they could install Wordstar – but the characters that should be in INVERSE are normal and vice versa.

The CP/M itself is far superior to CP/M 2.2 and, with the different processor, faster in operation than its older brother.

I have always been impressed with the very high standard of software that is available under CP/M, but the operating system has left a lot to be desired. CP/M Plus goes a long way to improving its image.

I spent a few happy hours playing with the many new commands and transient

programs that come with the package. I haven't listed them, or gone into any great detail as this would take a complete article on its own.

Instead I will pick a few facilities from which I can see an obvious benefit. Firstly, SUBMIT files. These are files which contain other CP/M 3 commands and hence can be used to create job streams or batch runs.

A special case of this is the PROFILE.SUB which is resident on system discs and is searched for and executed on startup if present. The HELLO program is called from PROFILE.SUB. This facility enables the operating environment (for example, the disc/file type search order using SETDEF, or the date/time set routine) to be set up automatically for an application disc. Further, the applications program could be called from the .SUB file – all without user action after the cold start.

The USER option allows files to be stamped with a user number from 0-16 and such files do not appear on other users catalog listings – could be very useful in a hard disc situation with many files. SHOW and CAT replace STAT command in giving information about disc contents, types, free space, etc. DIR remains, but is not as useful as CAT.

SET sets file, drive attributes, volume labels, password protection functions and date and time file stamping options. Program developers could use the date/time stamping to avoid losing track of which was the latest version of a file, as could other users who have time dependent files such as accounts files.

There are many other commands in CP/M Plus which have enhanced features – I have not even mentioned those to do with the assembly language programming facilities. Possibly this aspect could be dealt with in a separate article. Suffice it to say that this is CP/M Plus, not a variant of CP/M version 2.

I accomplished the installation within half an hour, including reading time – the card will go in any slot and still boot up (I have tried it in slots 1 to 7 on an Apple IIe, though when in slot 6, I had to move the disc controller to slot 7). Note that for an Apple II, a 16k RAM expansion card is necessary. ALS recommend slots 4 or 7, to fit in with other conventional slot selections.

For the inexperienced user it is possible that CP/M may be a bit much to swallow all at once, but it is possible to start small and build one's knowledge. The small computer system cannot devote the same amount of resources to the user interface as larger systems can.

Most CP/M 2.2 applications programs should work immediately under CP/M 3 – some problems may be encountered with those which have been written to intimately fit or interact with a previous operating system. Best procedure is to try before you buy at your local dealer.

The lack of hard disc support at the present time is disappointing. However the BIOS section has to be modified for every new hardware item and individual hardware manufacturers should produce their own drivers within a few months.

I am told that one networking firm has developed their own BIOS and can no take advantage of the 32 megabyte fi sizes possible under CP/M 3.

The ALS manual mentions a GSX-8 graphics extension programmer's guid This is not available because DRI ha changed their approach to graphics – least for Apple CP/M Plus. Graphics will supplied as an extension to CBasic at son time in the future. As CP/M is largely business operating system, perhaps that not too important. Assembly langua games should still work, but much faste

The CBasic supplied in the package not the native code compiler version, b the one which translates source code in intermediate code which is executed by run-time interpreter.

Overall, what is provided is somethi for everyone – a 6 mHz Z-80B process for scientific and technical numb crunching, a sophisticated operati system for enthusiasts and programme facilities for high and low level progra ming, and a cheap CP/M machine businessmen.

An Apple owner with a CP/M Ca would have access to the two large existing program collections in the wor Apple DOS and CP/M, as well as the futu developments which the advanc features of CP/M Plus will make possibl

However, as with any new item I wou advise prospective customers who upgrading their systems to ensure that t card works with their present system a software before they buy it. This can o be done by trying it, whether at t dealers', or at your own place of busines

Windfall has received the following letter from a dealer in Germany, Karl-Heinz Weiss of Wilhelmshaven.

I am a dealer and like to sell products, but with this card I am still cautious. The card has many good features, the hardware is all right, but the finished product was not finished or satisfactory at the time of writing (August 20) in terms of software.

The GSX-80 part is still missing although it was announced to be ready for shipment in the autumn. You cannot use any printer interface that works like the Grappler card, for example Microbuffer or Microtek's dumpling 64.

You cannot use drives other than the Apple disc drives, for example 8 inch with a controller like VISTA A800, unless you write all the necessary code yourself.

Several RAM cards like Saturn's or Vergecourt's Ramex cannot be installed with the CP/M Card for use as a RAM disc. If you have a normal Apple keyboard and a Videx Videoterm board or similar brands, you encounter trouble when you use the CP/M card with Wordstar or Datastar.

The WSMaker utility on the disc supplied by ALS does configure Wordstar, but only for use with ALS' own 80 column board – Smartterm.

However, if you use some kind of keyboard with separate keys for CONTROL and left square bracket [you will have no trouble at all.

You cannot use the MBasic supplied with the original Softcard and put it to use with the CP/M card. Instead you must take the Lifeboat Basic-80. At the moment the CP/M card has no utility like CONFIGIO to change character definitions.

Another important point is that the difference between free space with the Softcard's MBasic on the one hand and the Basic-80 and the CP/M Plus Card on the other is slightly more than 11 K1<.

● This reply is from Bob Ackerman, director of sales, Advanced Logic Systems.

GSX-80, as of this writing, has not been delivered by Digital Research, and therefore has not been delivered to any CP/M Card customers.

After delivery of a number of cards, we found that there were some shortcomings in the printer driver. Revision B software has now been released, solves the problem fully, and various customers indicate full satisfaction when using the wide variety of cards in the market.

The current release software does indeed address only the Apple controller. We are unable to define the full set of controller drivers that will satisfy everyone, so have been providing CP/M Cards to the controller developers. We expect that there will be a number of alternate controllers supported shortly by the manufacturers of the boards.

RAM discs fit into the same category as other controllers, and require drivers, dependent on hardware. We cannot select the card that will satisfy everyone, so intend to support the various card manufacturers.

The comment made about WSMaker is correct with respect to Revision A only Revision B replaces that program recognising the Apple IIe text card Videoterm and its copies, and also Smarterm boards.

MBasic is a version of Basic-80 designed by Microsoft for specific appli cation to the softcard operating system only. It does work on the CP/M Card, bu may not return to CCP properly.

The softcard is the only CP/M imple mentation that we know of that provides program like CONFIGIO, written to wor with MBasic to convert certain keyboard characters on the Apple II.

ALS's CP/M 2.2 and CP/M Plus take th more standard approach of convertin none of the keyboard input characters.

I am aware that there are Z-80 card sold without software, requiring eithe purchase of CP/M and the generation of BIOS or the copying of someone else software. One case is expensive an inefficient. The other is illegal in my countr and unethical in most.

Regarding free space, the transien program area of the CP/M Card is 60.5 we believe the largest currently available Supercalc signs on showing 35k of mode space on the CP/M Card, and only 25k o the CP/M 2.2 implementations.

We regret that Mr Weiss had difficult with the card, and intend to continue t upgrade our products and support.

Document debugging disappoints

By T.N. THOMPSON

HAVING read the advert for Disc-o-Doc from M.D. Software, I waited for it with anticipation. I'd already had many hours of satisfied service from both Bag of Tricks and The Inspector, and was expecting great things. What a shock when it arrived.

The disc costs $100 US (about £60 at present exchange rates) and the following facilities are advertised for the disc:

☐ Change DOS commands.
☐ Undelete disc files.
☐ Change programs while on disc.
☐ Remove DOS from a disc.
☐ Look into copy protected discs with a Nibble editor.
☐ Automatic error correction.
☐ User friendly and requires no knowledge of programming.

For anybody who has used either The Inspector or Bag of Tricks, it is obvious that Disc-o-Doc is offering a product similar in function. Without a good knowledge of the inner workings of DOS these disc utilities can be fatal to your discs, and of little value to the user *(Beneath Apple DOS, from Quality Software, is required reading).*

The disc boots on both 13 and 16 sector machines. When booted, a very pretty animated copyright message is displayed which, after a single key stroke, is replaced by a warning message: *"This program is designed to be used only by persons thoroughly familiar with the use of the Apple II disc drive and only after complete and careful reading of the documentation!!!".*

Disc-o-Doc is used to read one disc track at a time, either at sector/byte level, or at the nibble level using a nibble editor. The nibble level means simply dumping the contents of a track into memory without any error checking or translation to usable bytes. This nibble dump can access even copy protected discs, which means you can find out how they were copy protected.

Once read, a portion of the track is displayed on the screen for examination. Unlike Bag of Tricks and The Inspector, which both display the full sector on screen at once, only 50 bytes are displayed in three separate columns of 20 each. As a result, it takes four and a bit screens just to view one sector.

The program provides facilities to change anything on screen and then dump it back to disc. It is these screen alterations that allow DOS commands, error messages and programs to be changed on disc, as well as undeleting files and removing DOS.

Unlike Bag of Tricks, which is a joy to use, none of this is automatic, and sometimes requires a lot of brainwork.

Disc-o-Doc provides an easy method of changing the prologue or epilogue nibbles which DOS looks for on the disc as well as dispensing with the checksum error checking. This is the only nice, near automatic feature of the program. The other operations have to be performed manually and some can become quite longwinded.

The package comes with a 45 page manual which skims over DOS and its operation. It goes into some depth on certain aspects of the program, but on others it appears shallow.

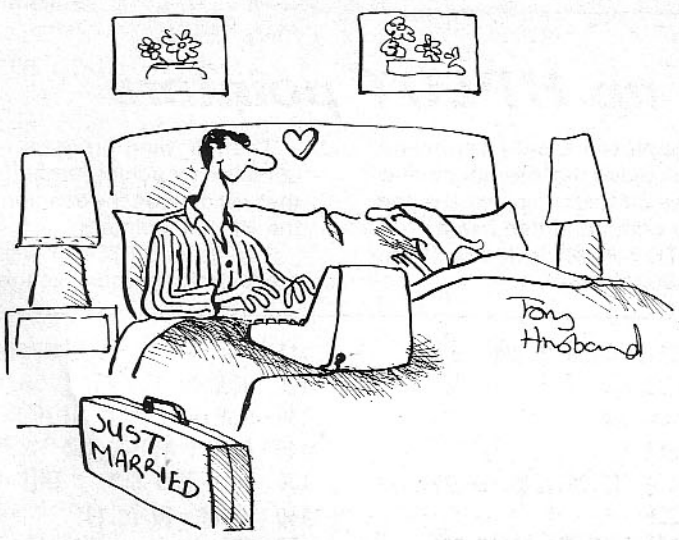
As an example, when Disc-o-Doc reads a track at sector/byte level all normal error checking is performed. If an error is flagged the program intercepts it and carries on. When all readable sectors on the track have been read the screen displays a list of sectors, along with a code, detailing the first error encountered for each sector, if any. These codes must then be compared to a list in the manual.

Most of this appears well documented, but on the first copy protected disc I read Disc-o-Doc produced three error codes that were not listed in the manual. To make matters worse, the manual is such that the meaning of these undocumented error codes cannot be deduced.

The more I used the system, the more errors I encountered in the documentation. This system is expensive, and I would have thought that documentation debugging would have been completed. Apparently not. The only nice thing I found with it was the ease of altering the expected prologue and epilogue nibbles.

All in all, I was very disappointed with this product. Although marketed as a means of disc access, it became apparent the program was intended as a means of assisting in copying copy protected software. Taken in this light, but for the missing error codes it would be a perfect partner to the bit copiers.

If you want disc access on a wider, better documented, scale, you can buy Beneath Apple DOS, Bag of Tricks AND The Inspector for the same price, getting all the facilities of Disc-o-Doc and a lot more besides.



"Wow, a personal computer. What a great wedding present..."

Almost, but not quite . . .

The suggestion that Allan Dubost (July Appletips) makes that POKE 44033,16 will move the catalog to track 16 is not entirely correct. It only moves the volume table of contents to track 16, sector 0, which is almost as good as moving the catalog itself.

Unfortunately there is a slight problem with moving the vtoc to another track, that is that its new position is not marked in the vtoc, so saving a program can potentially overwrite it with catastrophic affects.

This may not happen for quite a while because of the way DOS saves programs but the more the disc is used, the more likely that DOS will decide to use that sector.

My solution to this problem is to move the vtoc to track 2, to one of the sectors unused by DOS. This can be done by POKE 44033,2 and then poking 45069 with a value between 5 and 15.

Malcolm Whapshott

Keeping tabs

Modifying and debugging programs is made simpler if it is easy to find the various subroutines. The following tricks make it easier to see what's going on in your software.

Using linefeeds (CTRL-J) as the first and last character in the text of your REMs will make the text of the remark stand out in the listings.

Some printers will even put the remark at the left margin along with the line numbers.

You can use colons to indent FOR-NEXT loops. It makes them stand out and lets you know if you've closed them all before going on to the next routine.

```
10  REM
THIS STANDS OUT
20  FOR J = 1 TO 10
30 : FOR K = 1 TO 10
40 :: PRINT J * K
50 : NEXT K
60  NEXT J
```

CONTROL S catch

Sometimes if you use the CONTROL-S feature to examine a Basic program listing, you will get a SYNTAX ERROR with the next typed line.

This is probably because you accidentally typed an extra CONTROL-S after the program was done listing.

This extra CONTROL-S is put into the input buffer like any other characters and, since no Basic command starts with a CONTROL-S, you get a SYNTAX ERROR.

Typing the left arrow key until the Apple generates a new prompt line will insure that there is no CONTROL-S in the buffer.

S1FF8 saves space

One sector can be saved on the disc for each hi-res screen that you save if you do so with a length of S1FF8 instead of $2000.

The reason is that the starting address and length of the binary file are stored as the first four bytes of the file.

Saving a file with a length of $2000 actually stores $2004 bytes and requires another sector. The syntax will now be:

```
10 D$ =  CHR$ (4)
100  PRINT D$;"BSAVE HIRES-1,A$20
     00,L$1FF8"
200  PRINT D$;"BSAVE HIRES-2,A$40
     00,L$1FF8"
```

In the July 1982 edition of Windfall an Appletip suggested using hex to enable the drive to remain running while cleaning the head.

As I did not really understand this I have devised the following method using Basic, bearing in mind that the problem only arises when attempting to clean the head on drive 2.

Save the program on disc in drive 1:

```
10  ONERR  GOTO 30
20 D$ =  CHR$ (4)
30  PRINT D$"CATALOG,D2"
```

If not already there load this into memory. Place cleaning discs, suitably saturated, in each drive and run.

Wait 30 seconds and gently raise the gate to drive 2. Remove disc. Switch off the computer.

Pause, then switch on computer for 30 seconds. Open gate on drive 1 and remove disc. Switch off . . . hey presto!

William G. Watson

Set up HPLOT pointers

The Applesoft DRAW command doesn't leave the internal pointer for the last plotted point where the HPLOT TO command can use it.

HPLOT TO X,Y will draw a line from a random point to X,Y.

The following program from Apple contains a machine language program that will decode the position and set up the HPLOT pointers.

Then HPLOT TO X,Y will draw from the last plotted point of the shape.

```
10  FOR J = 768 TO 780
20  READ A
30  POKE J,A
40  NEXT J
50  DATA  32,203,245,166,224,164,
      225
60  DATA  165,226,76,17,244
100  SHLOAD
110  HGR
120  HCOLOR= 3
130  SCALE= 1
140  DRAW 1 AT 100,100
150  CALL 768: REM    THIS IS IT!
160  HPLOT  TO 10,10
170  END
```

PASCAL START UP SAVER

If you are using a Pascal-based program that auto boots, the Pascal system on booting runs a program named SYSTEM.STARTUP. If this program crashes out and simply displays the Pascal command line, typing X, for EXECUTE, and then SYSTEM.STARTUP will not work.

The P-code system automatically adds .CODE to the end of whatever you have typed, if it is not already there. This can be defeated by ending the SYSTEM.STARTUP with a full stop (SYSTEM.STARTUP.)

This may, in some programs, preserve any data that you have already typed in.

T.N. Thompson

Square way to tackle small symbols

The Apple's high resolution graphics plot is complex, but interesting. Consider Page 1, which starts at 8192 (hex $ 2000). After 40 squares it jumps down eight lines and so on. See Page 21 of the reference manual.

Now each square is eight bytes deep, and so a byte into 8192 puts your information into the top line of 8192's square.

If you wish to enter a byte into the second line of this square, it has to be in 8192 + 1024 = 9216, or hex $ 2400, and so on. Page 21 also explains this, in a vague sort of way.

Each of these lines (within the big squares) are filled by bytes from 0 to 127 (for example, 127 is all 1s and produces a full white line. Eight just produces a dot in the centre of the line).

It is therefore possible to compose any shape by attacking each square in turn and POKE in the bytes as planned from a graph. This would be tedious if you wish to do a big picture, but easy and useful for small symbols occupying, say, just four squares.

The following program enables you to do this. The second part of the program, from line 370, enables you to move all eight bytes from one square to another (with deletion, if you delete line 460).

If you want a clean sheet, type HGR before running the program, which avoids HGR so as not to wipe any BLOADED plots.

RUN the program, enter **8378** when asked for the store, and RETURN. Then enter, for example, **65, 34, 20, 8, 8, 20, 34, 65,** and RETURN.

This will produce a cross in the middle of Page 1.

Each square is 7 across by 8 deep, so get out your graph paper, the bits run from left to right – 1, 2, 4, 8, 16, 32, 64, – add these bits for each byte, and start plotting.

If you have a result which you wish to save on disc, then type **BSAVE HGR, A $ 2000, L $ 2000.**

A later **BLOAD HGR** will put this back into P1 for further work.

K. Archer

```
10  REM  HGRPOKE SEE P21 OF REF M
    ANUAL
20  TEXT : HOME : PRINT
30  PRINT "$$  HGRPOKE BY K.ARCHE
    R  $$"
40  PRINT : PRINT : PRINT "DELETE
     LINE 460 IF ERASURES REQD"
50  PRINT : PRINT : PRINT "THIS P
    ROG WILL ENTER A SHAPE"
60  PRINT " INTO ANY STORE INC P1
     OR P2 GR"
70  PRINT "(OR MOVE ONE, SEE LATE
    R * )"
80  PRINT : PRINT : PRINT "P1 STA
    RTS AT 8192 ($2000)"
90  PRINT : PRINT "P2 STARTS AT 1
    6384($4000)": PRINT
100  PRINT "  PRESS ANY BUT M,& R
    ETURN TO "
110  PRINT "   CONTINUE WITH NEW
    POKES"
120  INPUT "* TO MOVE A SHAPE PRE
    SS M *";M$
130  REM  GOTO HGR WITHOUT CLEARI
     NG PAGE 1 MEMORY
140  POKE  - 16297,0: POKE  - 163
     01,0: POKE  - 16304,0
150  HOME : VTAB 21
160  IF M$ =  CHR$ (77) THEN 370
170  INPUT "ENTER STORE,(0 TO END
     )? ";Q
180  IF Q = 0 THEN  TEXT : END
190  INPUT "ENTER 8 BYTES ";A,B,C
     ,D,E,F,G,H
200 K = 1024
210 Z = Q - 1024: GOSUB 230
220  GOTO 50
230  FOR X = 1 TO 8
240  IF X = 1 THEN Y = A
250  IF X = 2 THEN Y = B
260  IF X = 3 THEN Y = C
270  IF X = 4 THEN Y = D
280  IF X = 5 THEN Y = E
290  IF X = 6 THEN Y = F
300  IF X = 7 THEN Y = G
310  IF X = 8 THEN Y = H
320  POKE Z + X * K,Y
330  NEXT X: RETURN
340  REM  *             *
350  REM  * HGR SHIFTER *
360  REM  *             *
370  PRINT : PRINT " ** HGR SHIFT
     ER,0 TO END **"
380  INPUT "ENTER ORIG.LOCATION
     ";A
390  IF A = 0 THEN  VTAB 23: END
400  IF A < 8192 OR A > 15395 THEN
      GOTO 380
410  INPUT "ENTER FINAL LOCATION
     ";B
420  IF B < 8192 OR B > 15395 THEN
      GOTO 410
430  REM  READ 8 BYTES
440  FOR X = 0 TO 8192 STEP 1024
450  POKE (B + X), PEEK (A + X)
460  GOTO 480
470  POKE (A + X),0
480  NEXT X
490  VTAB 23
500  PRINT "  **** "A" MOVED TO "
     B" ****"
510  GOTO 380
```

Super-res graphics can double your dots

GRAPHICS are an important facet of any Apple usage. Games programs, for example, are often judged as much on their visual impact as by their skill and entertainment value.

Business, scientific and engineering applications become effective problem-solving tools often because of their graphics capabilities and potential.

In this issue we look at two graphics programming techniques, hi-res text and super-res graphics, and review a remarkable graphics package, The Graphics Magician.

Nick Levy also looks at graphical output in his Visicalc column.

Last month we introduced the world of Computer Aided Graphics with a review of the Digisolv Graphics System.

It is a measure of the speed in which things are moving in this field that there is already a new improved version of this system on the market.

Also worth noting is the fact that Apple's Lisa can also be purchased as a Computer Aided Design tool – using its built-in LisaDraw software package.

THE Apple's standard high-resolution graphics mode can display dots in a matrix 280 dots wide and 192 dots high, or so the Apple II Reference Manual states. This article describes how, with just a little thought and only one or two extra lines of coding, it is possible to increase this resolution to 560 dots wide by 192 dots high.

There is a price to pay, as the "extra" dots are limited to black and white, but the big plus is that you can program for this extra capability entirely using Basic and mix hi-res as well without much problem.

To explain where the extra dots come from it is necessary to look at how the Apple maps the bits in the high-resolution graphics page on to the dots on the screen.

The standard screen is divided into 40 columns, or character positions, each of which consist of an array 7 dot positions

By B.A. BAKER

wide by 8 dot positions high. Thus the 40 character positions per line also represent 7 × 40 = 280 dot positions.

Each group of 7 dots is associated with one byte of memory. If a bit is off (equal to 0), its corresponding dot will be black. If a bit is on (equal to 1), its corresponding dot will be a colour depending on the position of that bit within the byte and the setting of the most significant undisplayed (eighth) bit. (See Table 1.)

It follows that, apart from black or white, column 0 (or any even numbered column) can appear only violet or blue and column 1 (or any odd numbered column) can appear green or red.

If this is not clear then refer to page 19 of the Apple II Reference Manual for a more detailed explanation.

The next step is to look at how, and where, the Apple displays the four different colours (except black and white) on the screen. It is the way in which the Apple does this that provides the key to creating 560 distinguishable dot positions across the screen.

As we have already seen, column 0 can appear either violet or blue, but the important point is that the violet and blue dots are not displayed at exactly the same point on the screen. In fact, they appear shifted either half position left or half position right. (Table II.)

This can best be illustrated by running Program I and observing the shift in the alternator violet and blue dots at (0,0) against the alternative blue and violet dots at (0,1).

At this point it is worth noting that the two different values of HCOLOR which produce white (3 and 7) are also plotted at these slightly different points on the screen.

This can be demonstrated by re-running the above program after changing line 20 to read: C = 3 and replacing 8 – C by 10 – C in lines 60 and 80.

Having demonstrated that it is possible to plot at 560 dot positions across the screen, we can now look at how we can do this from a Basic program. The secret lies in selecting the appropriate value for HCOLOR before plotting. For example, suppose we wish to plot two points on the screen and their precise positions, before rounding, are (48.2,0) and (53.7,0). The normal HPLOT routines will round these

MSB	7	6	5	4	3	2	1
0	V	G	V	G	V	G	V

MSB	7	6	5	4	3	2	1	Bit
1	B	R	B	R	B	R	B	Colour

Table I

0	1	2	3	4	5	Column
V B G R V B G R V B G R						Dot position and colour

Table II

48	49	50	51	52	53	54	Column
■					■		Dot position if HCOLOR = 3
	■					■	Dot position if HCOLOR = 7

Table III

48	49	50	51	52	53	54	Column
■					■		Dot position, HCOLOR variable

Table IV

```
10 HGR
20 C = 2
30 FOR N = 1 TO 10
40 HCOLOR= C
50 HPLOT 0,0
60 HCOLOR= 8 - C
70 HPLOT 0,1
80 C = 8 - C
90 FOR M = 1 TO 1000: NEXT
100 NEXT
```

Program I

values down and they will be plotted as in Table III.

However, if we examine the fractional part before plotting and select the value for HCOLOR according to the rule: "If the fractional part is less than $\frac{1}{2}$ then set HCOLOR to 3 otherwise set HCOLOR to 7" then these two points will be plotted as in Table IV.

The extra resolution achieved by the technique is clearly visible. Program II shows a normal circle drawing routine and the two extra Basic statements required to implement this technique to increase the resolution to 560 × 192.

A useful application is for graph drawing, but note, of course, that HPLOT a,b to c,d cannot be used as the individual points forming the line have to be calculated and adjusted before plotting.

```
10  HGR
20 R = 64:P = 70:Q = 70:PI4 = 0.7
    854
30 S = 1 / 60
40  FOR A = 0 TO PI4 STEP S
50 X = R *  COS (A):Y = R *  SIN
    (A)
60 P1 = P + X:P2 = P - X:Q1 = Q +
    Y:Q2 = Q - Y
70 C = 7
90  HCOLOR= C
100  HPLOT P1,Q1: HPLOT P1,Q2
110  HCOLOR= 10 - C
120  HPLOT P2,Q1: HPLOT P2,Q2
130 P3 = P + Y:P4 = P - Y:Q3 = Q +
    X:Q4 = Q - X
140 C = 7
160  HCOLOR= C
170  HPLOT P3,Q3: HPLOT P4,Q4
180  HCOLOR= 10 - C
190  HPLOT P4,Q3: HPLOT P4,Q4
200  NEXT

80  IF P1 -  INT (P1) < 0.5 THEN
    C = 3
150  IF P3 -  INT (P3) < 0.5 THEN
    C = 3
```

Program II

A spell-binding system for magical graphics

By CLIFF McKNIGHT

AS regular readers of *Windfall* will know, I've reviewed several hi-res adventure games. One which I thought had particularly good graphics and fast colour-fills was Transylvania from Penguin Software (see *Windfall,* January 1983). This was created using Penguin's own graphics package, The Graphics Magician.

I wanted to write an adventure game and had got stuck at the graphics stage when I encountered The Graphics Magician. I'd looked at Graforth from Insoft (reviewed in *Windfall,* February 1983) and the well-known Apple Pilot. Both looked like excellent packages, and in my next incarnation I might have time to read the manuals!

I'd also looked at Artist Designer from Apple Special Delivery but had a bit of trouble running it on some Apples — notably the one I use most at home. Even when it ran, it was not at all user-friendly and I couldn't produce a decent (or even an indecent) picture.

In contrast, The Graphics Magician is a breath of fresh air. Despite the fact that it is a graphics *and* animation package, the manual is only 32 pages long. As the length of the manual suggests, the system is very easy to use.

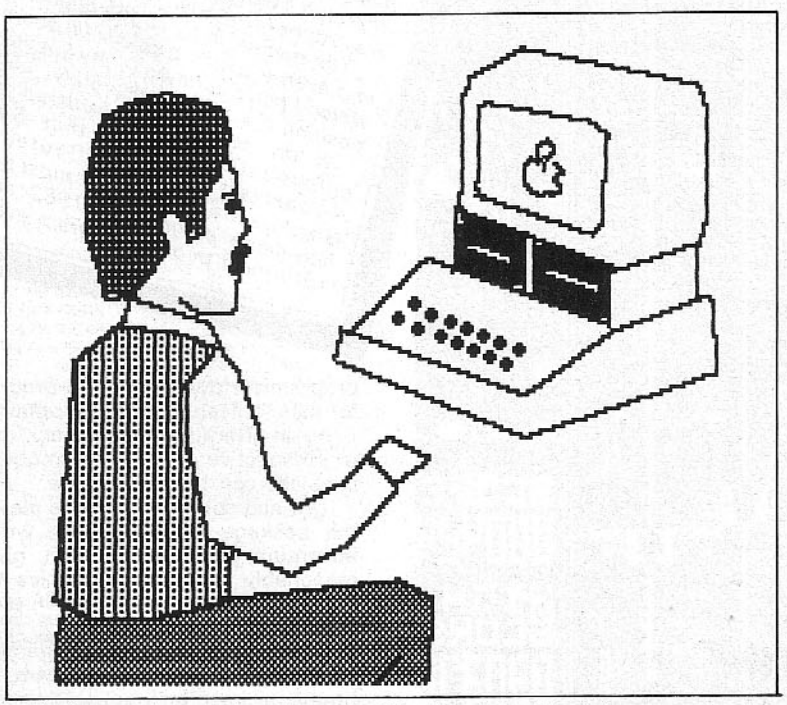
With my usual impatience, I had a quick look at the manual and booted the system and was drawing my first picture almost immediately. The example picture accompanying this review was drawn (by my artistic wife) in about 15 minutes.

Also included with the manual is a 12-page programming tutorial. This lists some brief routines which show you how the various creations can be incorporated into your own programs. I've tried several of these and they've all worked perfectly. The worst I could say is that I spotted an error in a REM line, which obviously doesn't affect execution.

If you've ever dumped a hi-res screen to disc, you'll have noticed that it occupies 34 sectors. However, The Graphics Magician uses a method of storage which reduces this to about three sectors. Instead of the screen image being stored, the moves necessary to create the image are stored.

Drawing is achieved with either paddles or joystick and the various controls are entered on the keyboard. In addition to line drawing, in which the colour of the line can be specified, there are also eight brushes, two of which give an air-brush effect. Colour filling is easy, with a choice of 108 colours.

Although cursor movement is reasonably easy, if you need fine control there is a sort of "zoom" facility which gives you the full range of joystick movement over a small area. This is useful for



Author and computer drawn by The Graphics Magician

GRAPHICS

filling in details, such as facial features, where only a small area is involved.

When you've drawn your masterpiece, a single keystroke will show you how it will be reconstructed at machine language speed. Hence, even if it's taken you ages, you can see how fast it will appear in your program. You can also edit or delete bits too, of course.

Although my main interest was in the picture graphics routines, I have tried my hand at animation too. There are four animation routines included, each with its own advantages and disadvantages. There are also three editors for animation: a shape editor, a path editor and an animation editor.

The shape editor lets you draw the shape you want to move around the screen, the path editor lets you specify the route to be taken and the animation editor lets you put the two together.

It was remarkably easy to draw a shape and no problem to get it whizzing around the screen. I don't pretend it would be child's play to compile a complete game, with crashes and so forth. However, all the ingredients are there and they're easy to use, which is half the battle.

Also included in the package is a binary file transfer utility which is useful for finding the starting address and length of binary files. Another bonus is the "animated alphabet" which is ready to use.

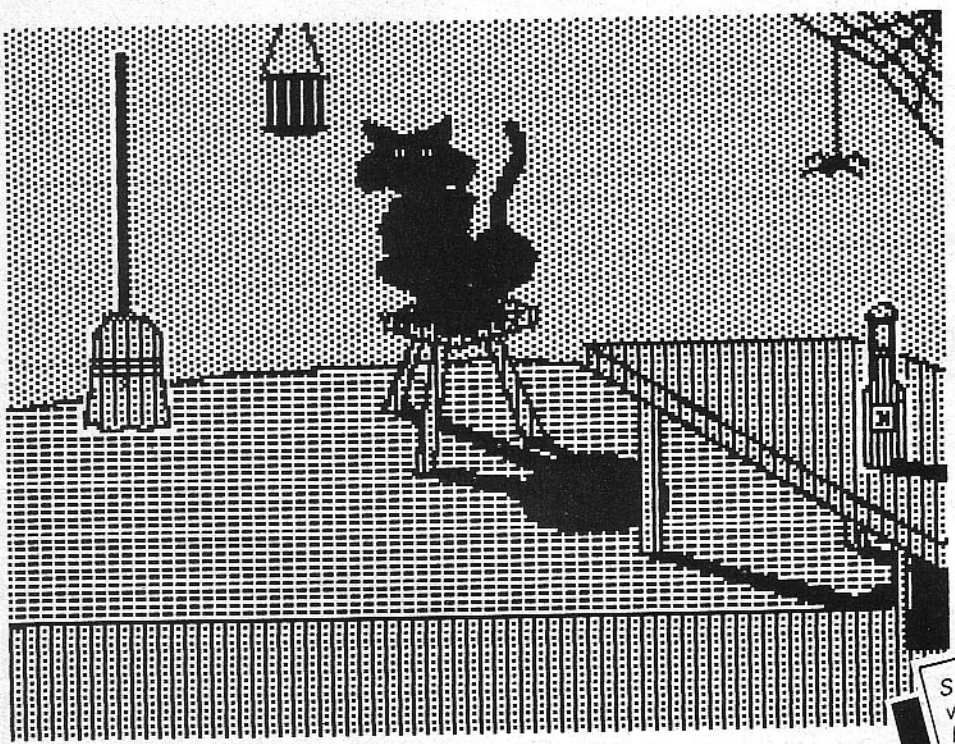
It is a measure of the power of The Graphics Magician that Penguin Software use it in making their own games. Transylvania is a testament to the graphics capability, and Pie Man and Spy's Demise, both excellent games, were created with the help of the package.

It is not only Penguin games that have benefited from the package. Police Artist, a new game from Sir-Tech (to be reviewed in *Windfall* soon), was also created using The Graphics Magician.

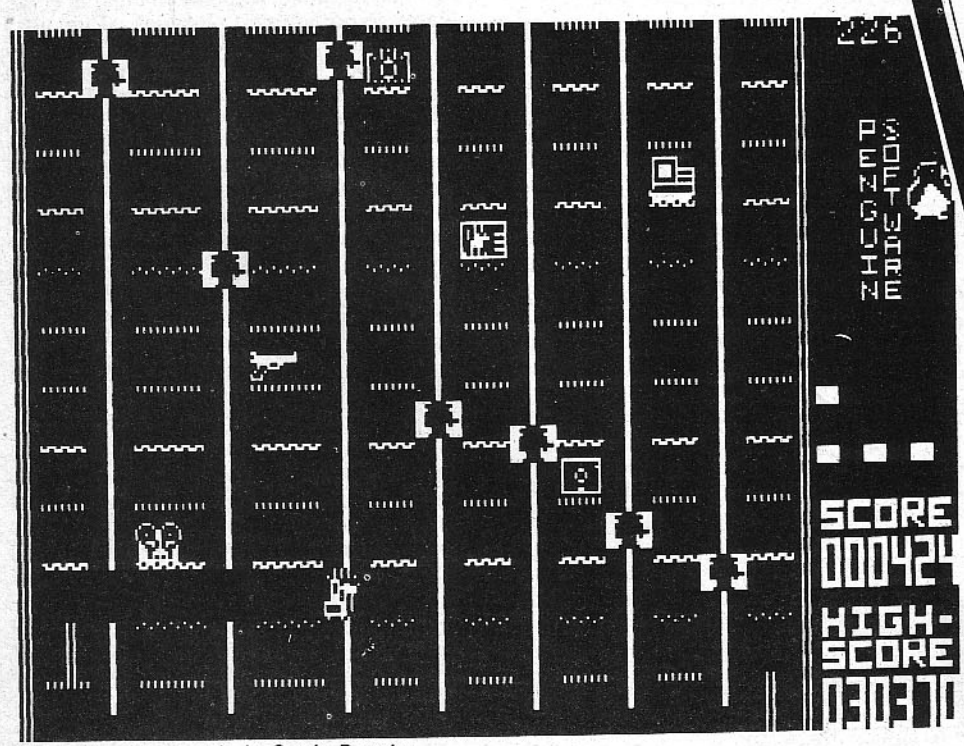
Penguin are one of the more progressive companies in the market. None of their applications software is protected, and any of the routines in The Graphics Magician can be used in your own programs. However, if the programs are for sale, a licence must be obtained from Penguin. The information pack they send to prospective authors is a model of how business can be friendly too.

I've had such a great time playing with the package that I haven't written the adventure game yet. In fact, given how reasonably priced The Graphics Magician is, I think I would choose it in preference to many games!

If you fancy your hand at game-writing and don't know where to start, this may be the package for you . . . I'll look forward to reviewing them in *Windfall soon.*



The lucky black cat from Transylvania



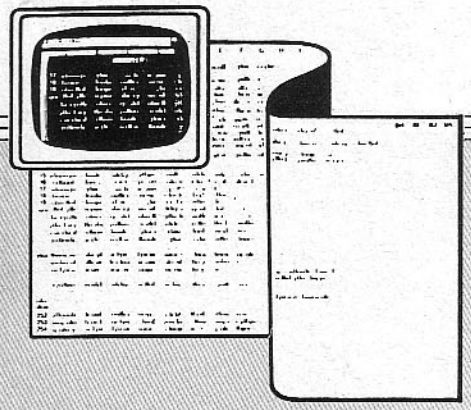
Dodging the guards in Spy's Demise

SINCE this review was written a new version of The Graphics Magician has been released. As you might expect, it's even better than before.

All the routines are faster, and a great many refinements have been made, although I wouldn't have thought there was room for many.

The manual is twice the original size and even explains a colour-fill anomaly which I had put down to my inability to position the cursor accurately.

If my review is considered over-enthusiastic, consider that Softalk magazine's 1983 poll rated The Graphics Magician as the most popular non-game program of 1982 and the 19th most popular program ever.



Put it in Apple pie chart order

By NICK LEVY
Principal,
Interface Management

WITH graphics under the spotlight in this month's *Windfall* we shall look now at how to produce good looking graphs from any Visicalc models.

Let's assume that you have a model with key figures which you want to plot and present in the form of bar charts and pie charts.

In order to draw graphs from any Visicalc model it must first be saved as a DIF file (DIF stands for Data Interchange File). This is easy to accomplish, and it only involves typing a short sequence of key strokes which start with /S#S.

A couple of examples will follow, but in the meantime why not look up the reference section of the VC manual starting from page 3-65, where there is a short thesis on DIF?

For demonstration purposes I shall use:

(a) The model shown as Exhibit I which will provide the data to be plotted.

(b) The Visitrend + Visiplot program for plotting lines, barcharts, piecharts, etc. If you use the Visiplot program without Visitrend the instructions which follow still apply.

(c) A program called Combined Enhanced Graphics Software for printing the graphs after they have been saved on a disc. I have to use this program because my old Paper Tiger printer does not have the hardware parts that turn it into a graphic printer, so I use this very versatile piece of software instead.

Start by copying Exhibit I (with column width set to /GC7) and save it as a /SS file under whatever name you choose. At this stage do not save it as a DIF file because the Visiplot program does not accept for plotting purposes just any file saved in DIF.

Ideally, any VC file to be used for plotting should consist of a single column of labels (text) followed, on the right, by one or more columns of values. Alternatively, your model could consist of a single row of labels (text) with one or more rows of values underneath, which, when saved as a DIF file, could in the saving process be turned around to reappear as a single column of labels followed by columns of values.

So before saving Exhibit I as a DIF file the layout of that model has to be changed and rearranged to make it look like Exhibit II.

Note that the changes affect only the columns and rows of text, labels and underscoring. They do not affect any of the rows or columns containing values.

Having created Exhibit II do save it as a /SS file (on the same spreadsheet as Exhibit I) but again do not save it as a DIF file.

Note that:

(a) Exhibit II contains one column of labels (I18 to I22) and one row of labels (J17 to K17). Bear in mind that when we

→

	A	B	C	D	E	F	G
1		PETROL SALES IN '000 OF GALLONS (IMP).					
2		==========					
3							
4		YEAR -)	1979	1980	1981	1982	1983
5		----------					
6		1ST QUARTER	678	751	837	711	791
7		2ND QUARTER	642	707	807	792	850
8		3RD QUARTER	686	764	850	887	935
9		4TH QUARTER	711	792	954	980	900
10		----------					
11		TOTAL:	2717	3014	3448	3370	3476
12		==========					

Exhibit I

	I	J	K	L	M	N	
17		1979	1980	1981	1982	1983	(- NOTE: THIS ROW CONSISTS
18	1ST QU	678	751	837	711	791	OF LABELS (TEXT) ONLY,
19	2ND QU	642	707	807	792	850	NOT OF NUMERALS!
20	3RD QU	686	764	850	887	935	
21	4TH QU	711	792	954	980	900	
22	TOTAL	2717	3014	3448	3370	3476	

Exhibit II

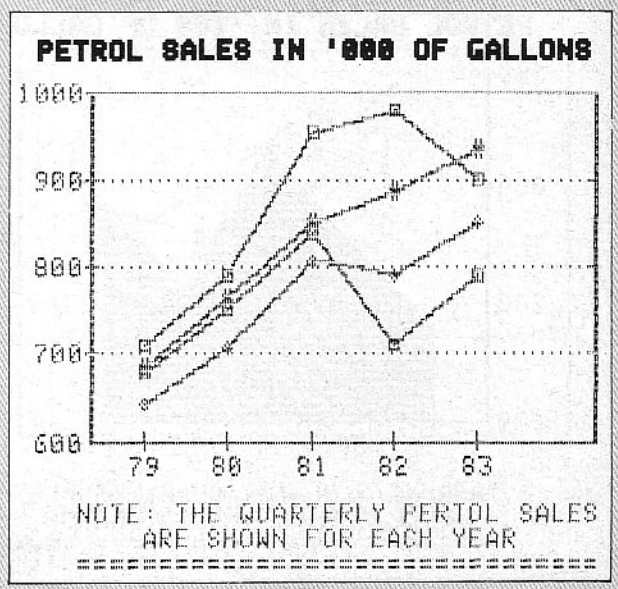


Exhibit III

create a DIF file from Exhibit II either column I or row 17 will be included in any one DIF file. As a general rule, if a DIF file is to be used for graph plotting purposes, never include in the same file a top row of labels as well as the column of labels which usually appear on the left side of most models. You should only have in your DIF file one or the other, not both together.

(b) Although columns I18 to I22 (Exhibit II) display "1ST QU", "2ND QU" etc, the cell content according to the entry line (the very top line on the VC screen) reads "1ST QUARTER", "2ND QUARTER" etc. In other words, the text of columns A and B in Exhibit I have been combined into a single column in Exhibit II.

The text in column I may be only partially exposed on the monitor, but the Visiplot program is designed to reach and read the parts you cannot see on your screen.

(c) Exhibit II does not contain any underscoring, nor any empty rows or empty columns or blank cells (except where it is intended to enter a value of nil).

(d) Before creating a DIF file make sure that all the columns in your model are of the same width. This is automatic if your model was produced with the Visicalc program, but if you use VC Advanced Version or Magicalc, the model which you created could contain columns of different widths, and DIF files created from such models would not be acceptable to Visiplot.

As you will appreciate, a knowledge of how to replicate, how to delete row and columns, and how to enter numbers as labels, is required in order to convert Exhibit I into Exhibit II. As mentioned above, both exhibits should then be saved, on the same spreadsheet with /SS.

We are now ready to create two DIF files for plotting graphs from the data appearing in Exhibit II. Two are needed because in the first instance we shall use row 17 as our horizontal axis, and in the second instance we shall use column I as the horizontal axis (in other words we shall see what our model looks like when turned sideways – see Exhibit VII).

Using Exhibit II, start by placing the cursor on I18 and type /S#S (note that we are not going to include row 17 in this DIF file). In response to VC's prompt for a file name type ABC DIF RETURN or any other name followed (optionally) by the suffix DIF and press the RETURN key.

Next respond to VC's request for the lower right hand corner of the model by typing N22 R (the final R stands for row). That's all there is to saving DIF files.

With the DIF file which you have just created the Visiplot program could be made to create the kind of graphs shown as Exhibits III, IV, V and VI, as well as other types of graphical presentations as described in the Visiplot manual.

Using the Visiplot program to draw graphics from DIF files is a different topic. This month we are concentrating primarily on how to prepare a Visicalc file to use with Visiplot. If you know how to prepare VC data files for conversion into the particular kind of DIF file which can be read and handled by Visiplot, then you are more than

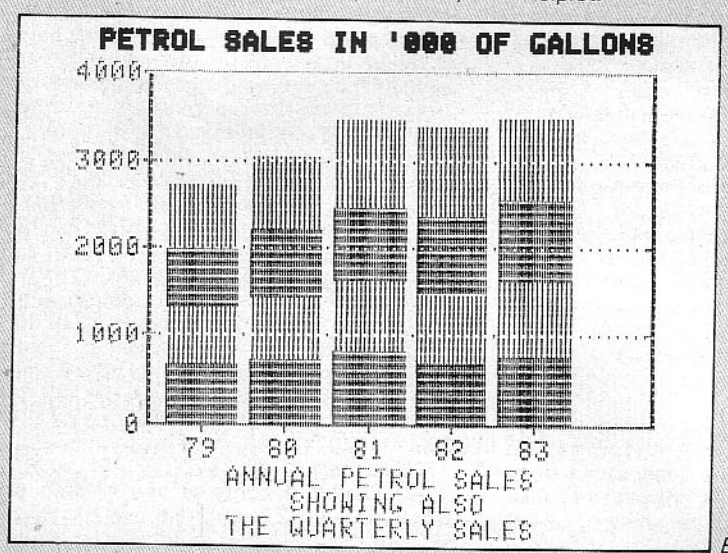


Exhibit IV

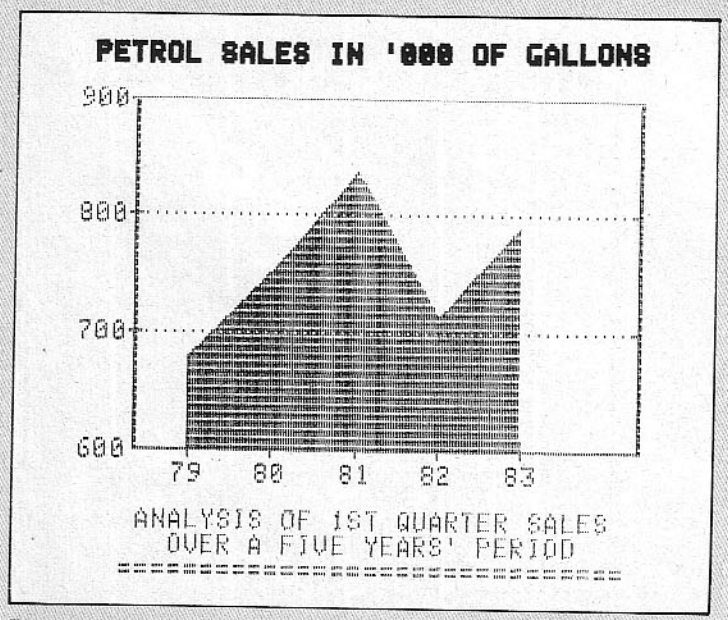


Exhibit V

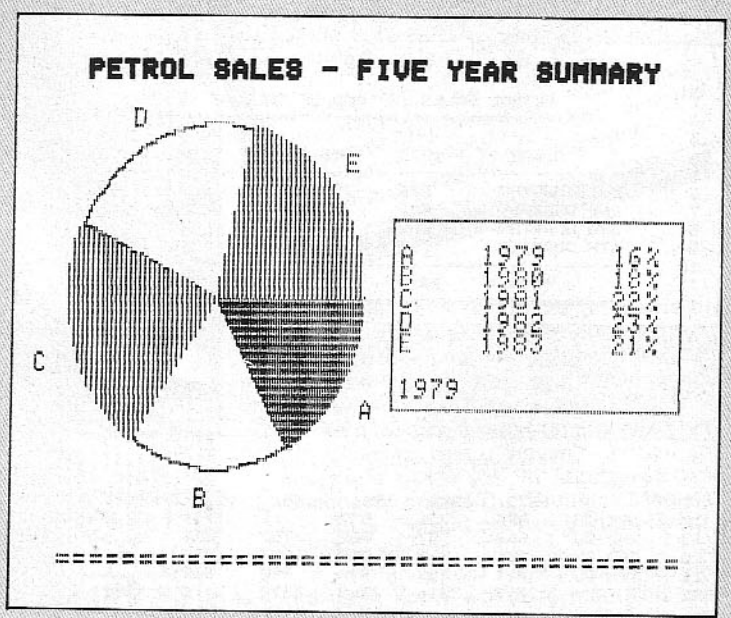


Exhibit VI

1979	678	642	686	711
1980	751	707	764	792
1981	837	807	850	954
1982	711	792	887	980
1983	791	850	935	900

PART OF EXHIBIT II TURNED ON ITS SIDE

Exhibit VII

two thirds on the way of being able to use Visiplot to produce very good-looking graphs.

Incidentally, do not try to draw any conclusions from the graphs accompanying this article. Any such inferences will inevitably be nonsensical because they were prepared only in order to demonstrate the variety of graphs that can be produced when you know how to convert Visicalc files into DIF files that can be read by the Visiplot program. The figures are entirely fictitious.

So far our model and graphs have shown the changes that occurred from year to year on a horizontal axis, and the quarterly changes on a vertical axis.

If you want to show the quarterly changes on a horizontal axis, you must first use Exhibit II to produce a new DIF file.

This time put the cursor on J17 (note that we are not going to include column I in our file) and type /S#S followed by a file name of your choosing, add the optional suffix DIF and press the RETURN key.

Next, in response to Visicalc's prompt asking for the bottom end of the model, enter N21 followed by C for column. Make sure that you have NOT entered N22 as the bottom right hand corner of the model to be saved. The reason for excluding row 22 is that the Visiplot program would have treated it as the fifth quarter in any year.

In addition, any diagramatic presentation of the figures in row 22 would dwarf the diagramatic presentation of the figures in the other rows because of the relative magnitude of row 22 compared to the other figures in the model.

You have just created a DIF file which looks like Exhibit VII. As you can see, what used to be columns of data in Exhibit II now appear as rows of data in Exhibit VII. With this exhibit saved as a DIF file you can now use Visiplot to produce the graphs shown as Exhibit VIII and IX.

Finally, you might be interested to know that when writing this article I was simultaneously using four Apple computers, four programs, two printers and one data disc which was shared by the four programs.

I started by loading Visicalc on one computer, prepared Exhibits I and II, printed them and created the two DIF files mentioned in the article as well as a couple of additional DIF files to experiment with.

I then moved to another computer, loaded Applewriter and started writing the article, saving it as I went along on the same disc which contained the VC files.

Next I loaded the Visitrend-Visiplot program on a third computer and began work on creating some of the graphics to accompany this article, as well as toying with some others.

The first attempts did not produce the desired results so I was able to go back quickly to the computer with the Visicalc program, make necessary corrections, reprint Exhibit II, and return to the computer with the Visiplot program.

Working with Visitrend-Visiplot is a rather slow, grinding process and if I had to use a single computer switching from Visicalc to Visiplot then returning to Visicalc to make some corrections and back to loading Visiplot, not to mention the other program which I had to use (Applewriter and the Enhanced Graphics) I would probably not have started such a project in the first place!

I was now ready and inspired to write a few more paragraphs using Applewriter, but at the back of my mind I was worried about how the graphics would look when printed.

So I set up the fourth computer, which had its own printer, booted it with the Enhanced Graphics program and started printing a few of the graphical exhibits which had been produced and saved with Visiplot.

I did not like the quality of the printing but, at least, I had peace of mind knowing that the graphics I saved on the disc were printable and I could reproduce them later on a better quality printer – if I got access to one.

Back to the computer running Applewriter to write a few more paragraphs, then to the computer running Visicalc (to produce and print Exhibit VII), then to the computer with Visiplot . . . and so on.

What motivated me in doing all this work was not the financial reward. My reward will come when and if I find out that this article has helped readers to create graphics from their Visicalc files.

My motivation stems from the feeling that I get when using such programs as Visicalc, Visiplot, Applewriter and Enhanced Graphics, that I create things as if by magic.

Regardless of the quality of the end-product, creating this article was certainly great fun. I wonder whether I would have had the same enjoyment if I had used Lisa, where the electronic spreadsheet, word processing and graphics are all integrated into the hardware, and you don't have to jump from computer to computer.

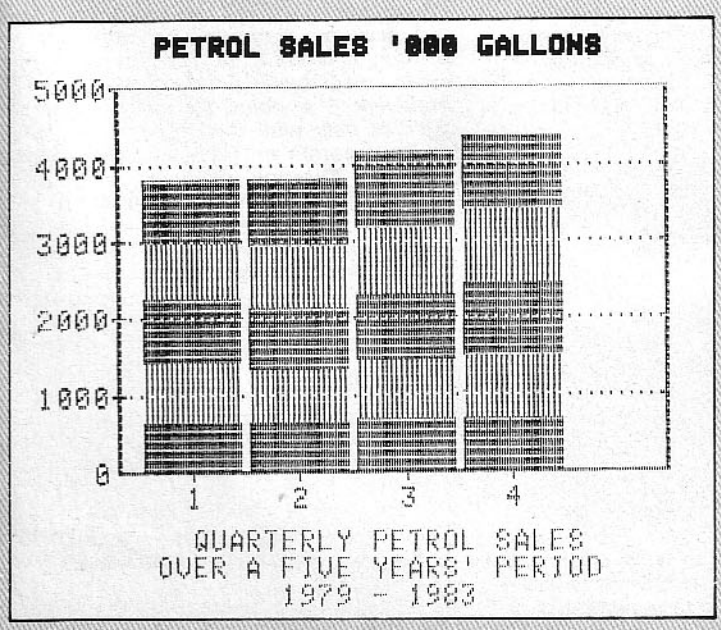


Exhibit VIII

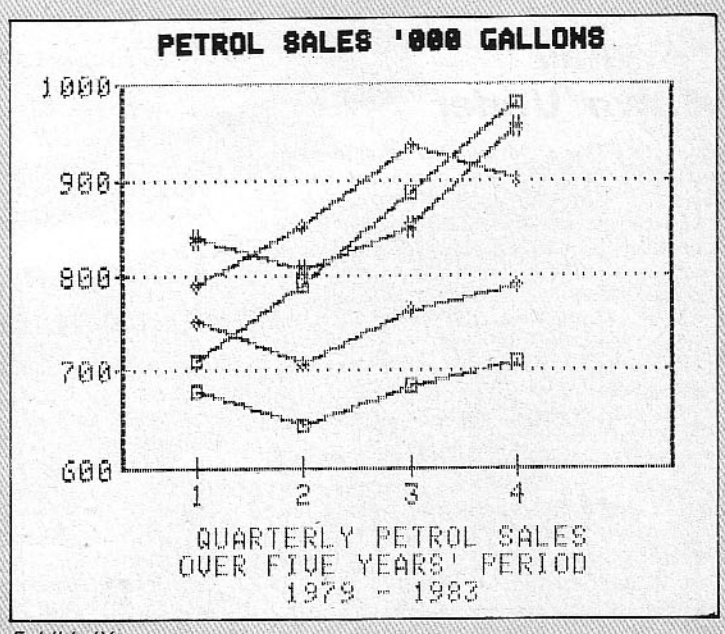


Exhibit IX

An ICE card for the hardware hacker

WE looked last month in general terms at the requirements for developing control applications on an Apple II and transferring the resulting programs onto a "target" single board computer (SBC).

It was seen that in-circuit emulation (ICE) allows the full integration of the software and hardware testing. Although ICE has been available on commercial microprocessor development systems since around 1976, there appears to have been little effort to extend this to the bottom end of the microcomputer market.

One exception to this was a low-cost ICE card produced by Acorn Computers for use with their development system. However, the facilities it afforded were somewhat limited, and it didn't really gain widespread acceptance. Although it is possible to attach this card to the Apple, it was apparent that there was a real need for a purpose-built ICE/Debug card for the Apple. This need has been satisfied by Rovino of Bath, with their ICE II card.

At under £200, with full software support, it is a very desirable acquisition for anyone who uses the Apple for developing control applications eventually intended to be implemented on SBCs. In other words, it is for the "hardware hacker".

Prospective customers would be well advised that a thorough knowledge of 6502 assembly language programming and hardware principles are necessary to be able to exploit the full potential of this board.

However, in the right hands it is a very powerful tool indeed – in fact Rovino themselves describe it as a being a kind of "microcomputer multimeter".

ICE II consists of a standard height Apple peripheral card about 19cm long with a 60cm ribbon cable attached at the keyboard end. This terminates in a 40 way header which plugs into the target board in the socket normally occupied by its 6502 microprocessor.

The only other hardware supplied is a small jumper cable, intended for linking the Apple's 6502 SYNC pin – pin 7 – to the appropriate terminal post on the ICE II card.

This is not strictly necessary for all applications, but is generally useful if extensive debugging is to be carried out, as it allows one to distinguish between opcode fetches and other bus activity.

Four other terminal posts provided on the card are labelled GND, +5v, SCOPE and DEC. The +5v simply makes available the Apple's supply for powering the target SBC, provided its power consumption (together with other peripheral cards) isn't such that the 2.5 amp total capability of the Apple power supply is exceeded.

Under normal circumstances the SBC will have its own supply and its ground will be commoned with that of the Apple via pins 1 and 21 of the header.

The DECode post is tied to the combined trigger line of the ICE II card and will go to logic 0 for 500 nS when the address set on the card matches the current address (the address match capabilities will be dealt with later).

This can be useful for several purposes, including triggering a logic analyser or oscilloscope. However, for the latter purpose, an extended trigger pulse is available on the SCOPE post.

The most powerful debug facilities of the ICE II card are determined by the state of 48 DIL switches located along the top edge of the board – their functions are described in Figure I. These allow the board to trigger when the current address of the executing program on the Apple falls within the range of addresses determined by the Address Match and Address Qualifier switches.

For example, if single stepping is required within the address range $0200 to $023F, the Address Match switches would be set to $0200 and the Address Qualifier switches to $FFC0.

Address match alone is not sufficient for the board to trigger – the appropriate Machine Cycle Qualification and Enable (MCQ) switches must be set. These allow the user to further restrict the triggering to, for example, only read or write operations or opcode fetches.

A $\overline{\text{BDEN}}$ switch is provided to enable the board (that is the address comparator), and this is indicated by a red LED illuminating.

When the board triggers there is a momentary flash of the green LED, and the effect of the trigger is then determined by the state of the Output Function switches – either a RESET, an IRQ or NMI may be received by the Apple when this occurs, and the appropriate software must be provided to service them.

In some cases the software may be written by the user, but mostly it would be provided on the system software disc supplied by Rovino.

When the system disc is booted we are presented with Rovino's logo, then a main menu of five items:

- Installation of the ICE and debug monitors.
- An ICE (target board) RAM tester.
- An Apple RAM tester.
- Loading or configuration of a printer spooler.
- A disc disassembler.

The RAM testers are run in an identical fashion – the start and end addresses are entered, and a rather thorough RAM test starts.

The only indication that anything is happening is a tiny flashing dot above the first "R" of the RAM tester title. The tester takes about 17 sec for 1k of memory – its completion is signalled by an asterisk printed on the screen, whereupon the test immediately recommences.

To stop the test CTRL-C is entered, but the current test cycle is completed first, which could be a slight annoyance. Another CTRL-C returns us back to the RAM test prompt, or pressing any other key restarts testing.

If an error is detected ERR is printed

followed by the memory address, the expected and the actual contents.

Systematic examination of these can often show patterns in either the address or data that may reveal the source of the problem, for example, stuck or shorted address or data lines.

Notes were provided on how to configure a spooler program for particular printer cards but, since the author's Apple already has a 32k Microbuffer II installed, its own buffer had to be disabled before running the spooler program.

Although it worked satisfactorily, it is not clear to what extent Rovino intends to support this particular option. I would have preferred, say, a mini-assembler instead. If someone really wants a spooler, they would probably prefer to buy a purpose-built card, since the cost of these is dropping as memory costs reduce steadily. Also there is then no reduction in Apple memory available for programming.

The "disc" disassembler allows machine code from either the Apple or ICE (target) memory to be disassembled into a named text file (complete with line numbers) on disc.

This file may then be edited, using, for example, the editor provided with the Applesoft Toolkit, and reassembled.

This facility was found to be extremely useful. For example, when used in conjunction with the Debug and ICE monitors, it was possible to disassemble a suspect program in EPROM from the target, alter it to run in the Apple's RAM (but still using the target's I/O devices), fully debug it and finally prepare a corrected EPROM.

Having dealt with the menu items in a somewhat arbitrary order, we now turn our attention to the first item, which installs the debug monitor for developing programs in the Apple, and the ICE monitor for controlling access of the facilities of the target via the ICE cable and header.

After selecting the first menu item and entering the slot number for the ICE card, the screen lists the addresses of the four interpreters which will allow access to the target board memory map when running a program on the Apple.

At this stage this information could probably be omitted, and would be better included only in the manual. Pressing any key completes the setting up of the monitors, and they may be entered at any stage from Basic by CALL 0, or from machine code by 0G. This causes the "DEBUG>" prompt to be displayed.

To enter the ICE monitor one simply types "I", and the debug can be re-entered by typing "N".

The debug monitor, loosely speaking, includes all the facilities of the Apple machine code monitor and provides several important new ones. Examining and changing memory locations, moving and comparing areas of memory, 8 bit arithmetic, output redirection to any I/O slot, disassembly, register display and alteration, and program execution are exactly as described in Chapter 3 of the Apple II Reference Manual.

Although it was mentioned earlier that it might have been useful to have a mini-assembler, as on the earlier Apples, it is

Function	Switch	Description
Address Match	A0 A1 A2 A3 A13 A14 A15	*These tell the ICE II card what initial address to look for. When match occurs between current Apple address and this address (plus those resulting from the Address Qualifier switches), together with the appropriate MCQ switches, the card produces a trigger pulse (indicated by the green LED) which may cause IRQ, NMI or RESET, as determined by the Output Function switches.*
Address Qualifier	X0 X1 X2 X3 X13 X14 X15	*These determine which bits set by the Address Match are to be included in the address comparison. If address qualifier bit=1, corresponding address bit is included. If qualifier bit=0, the address bit becomes a "don't care".*
Output Functions	iIRQ iNMI	*Logic 1 enables IRQ and NMI from the target (via the ICE header) to route to Apple for servicing.*
	iRES RAI OV	*1 reset on 1 target 0 resets Apple* / *1 resets 0 target 1 board* / *1 resets 1 Apple 1 Apple & target*
	aRES	*Apple receives a RESET when address match triggers.*
	aIRQ	*Apple receives an IRQ when address match triggers.*
	aNMI	*Apple receives an NMI when address match triggers.*
Machine cycle qualification and enable (MCQ)	RDY	*Connects RDY on 6502 to ICE header to extend read cycle.*
	EN∅2 ∅2	*Enable ∅2: enables state to ∅2 to be included into address match (e.g. set ∅2=1 for valid address on bus).*
	EN$R\overline{W}$ $R\overline{W}$	*Enable RW: enables state of RW to be included into address match (e.g. set RW=1 for memory read operation).*
	ENS SYNC	*Enable SYNC: if SYNC jumpered to 6502, this may be included into address match (SYNC=1 for opcode fetch).*
	$\overline{BDEN}$	*Board Enable: logic 0 enables address comparator, and red LED illuminates.*
	1 ⟷ 0	

Figure I: Debug function switches

By W.G. ALLEN

accepted that the level of software development involved in the present context would almost certainly justify a full editor/assembler.

However, two facilities that are no longer available as standard are STEP and TRACE. These are so vital for program development that Rovino has provided a comprehensive single-step and trace facility, linked to the function switches described in Figure I.

The board is first disabled, then the H (Halt) command is entered. Two options, T/S (Trace/Single-step) are given to define what to do after the first program interrupt (that is when an address match occurs).

Suppose S is entered – the display responds with NMI – STEP/TRACE, showing that the NMI vector has been set up. We now set the Address Match and Address Qualifier switches to cover the required memory range, the Output Functions switches to $80 (that is the Apple receives an NMI when the address match occurs), and the MCQ switches to $66 (for halting on an opcode fetch and enabling the board).

The program is now executed using the G command, and the display will show the instruction just executed, the resulting state of all the registers (including the individual flags of the status register), and the next instruction about to be executed.

Single stepping is obtained by successively pressing the S key, and would continue as long as the program remained within the address match range. Outside this range the program executes at full speed, so it is easy to set the switches so that, say, a section of program executes at full speed but a certain subroutine single-steps, or vice versa.

In fact, at any stage the switches can be reset to new values and, of course, the register values can be changed.

The T (Trace) facility works in an exactly similar manner, and effectively provides a continuous single-step function. However, one very nice feature is that the speed of the trace can be regulated by pressing the number keys 0 to 9, resulting in about 13 to 0.6 instructions per second, respectively.

When initially entering the H command, if the Address Qualifier switches are set to $FFFF (match on a single address), the program obviously sees the address set on the Address Match switches as a breakpoint. The switches could then be altered to allow single-stepping or tracing as before.

Since these facilities are hardware-based, it can be very instructive to trace through a Basic program, especially when the output is directed to the printer for a hard copy.

An Access Address facility is provided which allows the ICE II card to monitor the address of any instruction which accesses a particular memory location.

The Address Match and Address Qualifier switches are set as for a breakpoint – for example, when set at $02F8 and $FFFF respectively, we would be monitoring access of memory location $02F8.

The Output Functions switches would be set at $80 to again access the appropriate routines via an Apple NMI and, with the MCQ switches set at $0E we would be matching only on a write operation.

Of course, we could equally well set MCQ=$1E for a read-only access, or $06 for all accesses of the chosen location. The debug command Y is entered to point to the access routine, then the program is executed using the command G.

The screen might then show, for example, 02F8 ACCESSED FROM (–1 INST) 0849, together with a dump of all the registers and a C/S prompt for continuing or stopping program execution.

The reason why the instruction after the accessing one is displayed is connected with the pipelined architecture of the 6502, but this causes little difficulty in practice since, at worst, one has to disassemble a small region of code to identify the actual instruction.

The debug command Q behaves in an exactly similar manner, except that a continuous printout of accesses is given (without the need for repeatedly pressing the C key to continue execution) until CTRL-C is entered.

Another feature, similar to the above, is a Call Address facility. This is used to find the address from where a particular subroutine is called (the address of the latter being entered on the Address Match Switches as before). It is accessed by the debug commands X and P, and this time the MCQ switches are set at $66 (for an opcode fetch).

We no longer have the problem caused by the pipelining, since all JSR instructions are three bytes long, so the software can calculate the calling address without ambiguity. X again presents us with the C/S prompt, and P lists the calling address continuously until CTRL-C is entered.

A final facility of the debug monitor is the data window. This allows 1, 2, 4, 8 . . . 128 consecutive bytes of memory to be "windowed", that is, their contents displayed continuously during program execution.

Certain memory locations, such as the stack and those used by the window routine itself ($40 – $44) should not be displayed, and the main restriction is that the initial address displayed should have its last 3 bits =0. That is, the last digit of the hexadecimal address should be either 8 or 0.

After setting the appropriate initial window address on the Address Match switches, and the Address Qualifier switches at $FFFF (for windowing 1 location) . . . $FF80 (for 128 locations), the Output Functions switches are this time set at $40 (since the window routines are accessed via the Apple's IRQ).

Finally, the MCQ switches are set at $06 for any access, and the command W initiates the function.

The only problem encountered with the data window was that the ever-present pipelining necessitated the insertion of a dummy NOP instruction between two adjacent accesses of a windowed location.

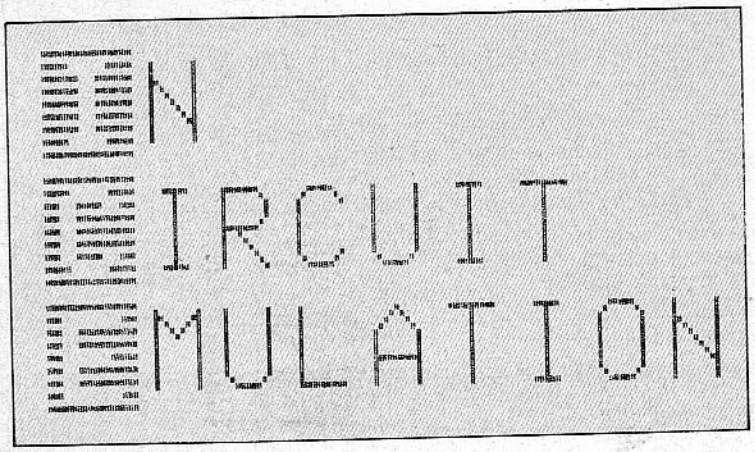
This is most likely to be required when initialising, for example, a list of zero page variables – it is a minor problem, and easily overcome.

Again, Basic provides a very interesting demonstration for the Data Window – after setting up a window for zero page locations $80 to $FF, we can return to Basic by entering 3DOG and examine Apple's use of these locations when running a simple program.

So far, the features of the ICE II have been described in the context of debugging programs running on the Apple and generally speaking, can be obtained from other hardware/software packages on the market.

Where Rovino's card really comes into its own is when its facilities for in-circuit emulation are combined with those for debugging.

Having booted the system disc and entered the debug monitor, the IC monitor is obtained by entering I, and th



new prompt ICE> is displayed, indicating that any commands now entered will influence the 64k memory map of the target board rather than the host Apple.

The ICE monitor commands available include those for examining, changing and disassembling memory locations, with an identical format to the debug monitor. For example, DEBUG>8000L would disassemble 20 instructions from address $8000 on the Apple, and ICE>8000L does the same thing for the target board memory.

Of course disassembly from target to disc can be carried out using the fifth item of the main menu. Another command identical in both monitors is for output redirection – 1 CTRL-P can be used to direct the screen output to a printer.

Whereas the debug monitor provides facilities for moving blocks of memory within the Apple memory map, the corresponding facility for the ICE monitor is to transfer blocks of memory from the Apple to the target board, and vice versa.

This is done using the ICE commands M and I, respectively. For example, 800<F800.F84DI moves the contents of the target's locations $F800 to $F84D to the Apple's $800 to $84D.

In addition to the facilities offered by the ICE monitor, the in-circuit emulation can, as expected, also be invoked from an application program running on the Apple. This is done by using four "interpreters" – subroutines which redirect memory access from the Apple's memory map to the target's. These are INTER1=$9240, INTER2=$9430. INTER3=$9500 and INTER 4=$95AO.

INTER1 is inserted immediately before an instruction which is intended to access a target board memory location using Absolute, Absolute Indexed, Zero Page, or Zero Page Indexed addressing modes – it also contains extensive error checking facilities, which obviously has the effect of slowing down the execution time of the interpreted instruction.

On the other hand, INTER4 is only intended for use with instructions using Absolute addressing, and therefore has no error checking. This speeds up the access considerably, and it is estimated that the delay introduced by this interpreter is about 120 microseconds.

It is possible to further speed up the interpretation process by writing one's own routines, but one should bear in mind that the delay only occurs for a very limited number of instructions – all the rest execute at full speed.

No mention was made above of the problems associated with Indirect addressing. It should be apparent that, although the instruction itself is in the Apple memory map, one may wish to utilise zero-page locations in either the Apple or the target (see Figure III of Part 1 of this article).

Coupled with the fact that the effective address accessed by the indexed addressing may also be required in either the Apple or the target memory map, it is necessary to provide interpreters to handle all eventualities – this is why the emulation of indirect addressing may require one of three interpreters (INTER1, 2, or 3).

The manual deals very thoroughly with this aspect and, in any case, the use of the wrong interpreter usually causes an error message on the Apple screen.

The best way to illustrate the application of the in-circuit emulation is to consider a simple application, and the ubiquitous traffic light sequencer is ideal.

Let us suppose that the lights are connected to a typical I/O port, first of all on the Apple, then ultimately on a target SBC – I used an Acorn controller card for this purpose.

The emulation increasingly utilises more facilities of the target board as each stage of development is completed, as illustrated in Part 1 in last month's *Windfall*.

The assembly language listing for the program is given in Figure II. Note that the program has not been written with elegance or efficiency in mind – for example, the time delay routine was written to include a zero page location solely to illustrate the use of INTER1.

As it stands, the program in Figure I runs entirely on the Apple. The next stage

→

```
0800            1  ;PROGRAM "TRAFFIC1"
0800            2  ;
0800            3  ;SIMPLE TRAFFIC LIGHT SEQUENCER
0800            4  ;(APPLE USING VIA CARD IN SLOT 5)
0800            5  ;(17 JULY 1983)
0800            6  ;
0800            7  CNTR    EPZ $00              ;NO. OF SECONDS DELAY REQUIRED
0800            8  PT1PRB  EQU $C500            ;PERIPHERAL REGISTER B
0800            9  PT1DRB  EQU $C502            ;DIRECTION REGISTER B
0800           10  ;
0800           11         ORG $800
0800           12  ;
0800           13  ;SET UP I/O PORT FOR OUTPUTS ON LINES 1,2,3
0800           14  ;(PB0=RED, PB1=AMBER, PB2=GREEN)
0800           15  ;
0800 A907      16         LDA #%00000111
0802 8D02C5    17         STA PT1DRB
0805           18  ;
0805 A901      19  START  LDA #%00000001
0807 8D00C5    20         STA PT1PRB           ;SWITCH ON RED
080A A90A      21         LDA #$0A
080C 8500      22         STA CNTR             ;DELAY 10 SEC
080E 203808    23         JSR DELAY
0811           24  ;
0811 A903      25         LDA #%00000011
0813 8D00C5    26         STA PT1PRB           ;SWITCH ON RED & AMBER
0816 A901      27         LDA #$01
0818 8500      28         STA CNTR             ;DELAY 1 SEC
081A 203808    29         JSR DELAY
081D           30  ;
081D A904      31         LDA #%00000100
081F 8D00C5    32         STA PT1PRB           ;SWITCH ON GREEN
0822 A914      33         LDA #$14
0824 8500      34         STA CNTR             ;DELAY 20 SEC
0826 203808    35         JSR DELAY
0829           36  ;
0829 A902      37         LDA #%00000010
082B 8D00C5    38         STA PT1PRB           ;SWITCH ON AMBER
082E A901      39         LDA #$01
0830 8500      40         STA CNTR             ;DELAY 1 SEC
0832 203808    41         JSR DELAY
0835           42  ;
0835 4C0508    43         JMP START            ;KEEP REPEATING SEQUENCE
0838           44  ;
0838           45  ;SUBROUTINE DELAY
0838 0600      46  DELAY  ASL CNTR
083A 0600      47         ASL CNTR
083C A2C3      48  COUNT  LDX #$C3
083E A0FF      49  AGAIN  LDY #$FF
0840 88        50  MORE   DEY
0841 D0FD      51         BNE MORE
0843 CA        52         DEX
0844 D0F8      53         BNE AGAIN
0846 C600      54         DEC CNTR
0848 D0F2      55         BNE COUNT
084A 60        56         RTS
084B           57  ;
               58         END
```

Figure II: Example program

was to connect the ICE II card to the target SBC via the emulator cable and header and utilise the target's zero-page RAM. This was done by simply inserting JSR INTER1 just before accesses to the zero page variable CNTR on lines 22, 28, 34, 40, 46, 47 and 54 (having, of course, defined the interpreter by an equate, INTER1 EQU $9240, at the beginning of the program).

Next the I/O of the target board was utilised instead of the Apple's. This involved first of all changing the addresses on lines 8 and 9 – PT1PRB EQU $0921, and PT1DRB EQU $0923. Then JSR INTER4 was inserted just before accesses to these new I/O locations – on lines 17, 20, 26, 32 and 38 of the original program.

Finally, when the program is ready for transfer completely to the target SBC, all the JSR INTER1, JSR INTER4 instructions are removed.

The program's origin is changed by altering line 11 to ORG $F800 to suit the location of the SBC's eprom, the instruction OBJ $800 (say) is added to define the location of the object program after assembly, the stack is initialised by adding the instructions LDX#$FF and TXS, and finally the reset vector is defined by inserting 00 and F8 into memory locations $FFFC and $FFFD, respectively.

When assembled, the object program is transferred to an eprom using a suitable programmer, and this eprom and a 6502 microprocessor are fitted to the SBC. The program now runs on the target SBC as soon as the reset button is pressed.

It's as simple as that!

Well, at great length, the process of prototype development has been described and the facilities of Rovino's ICE II card have been examined in detail.

But what did I *really* think of the card and its support software and documentation? I have nothing but praise for the whole concept, and the quality of the board.

As far as the support software is concerned, this was quite comprehensive, but could have been a little more user-friendly. For example, being presented with flashing instructions to "READ THE MANUAL" caused me some irritation.

Also, as stated earlier, I felt that a description of the ICE interpreters on the screen only caused confusion, as the correct choice of interpreter is best decided with the aid of the diagrams and thorough explanation provided in the manual.

Rovino might also like to heed my comments regarding the Apple and ICE RAM testers. These criticisms are, however, very minor, and easily rectified.

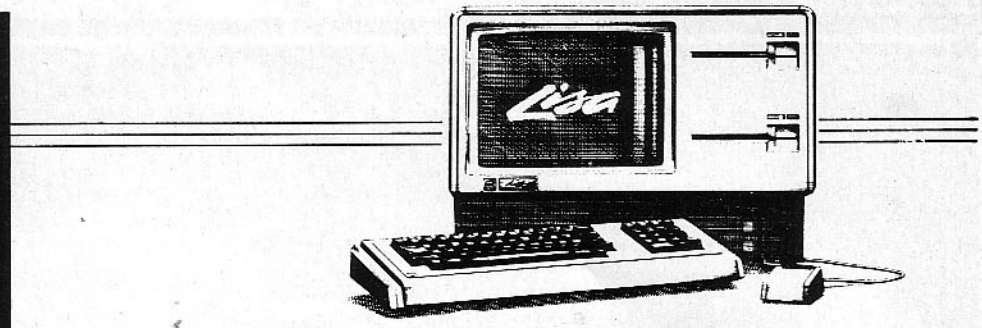
Realistically, I received a hot-from-the-press version of the software disc, and a preliminary version of the instruction manual – such is the novelty of this product. Rovino should therefore be given the opportunity to react to these kind of criticisms from their customers.

Having had the use of the ICE II for several weeks, it was used with a vengeance on a couple of prototyping projects with which I am currently engaged, and I found it absolutely invaluable. It actually allowed me to locate and correct a fault on a controller card that had eluded me for several months. What higher recommendation than that?

The only problem is that I have now become totally dependent on the ICE II, and since I couldn't bear to part company with it, I've sent my cheque off to Rovino. I hope *Windfall* doesn't ask me to do any more reviews – I just can't afford it!

UNDER an agreement signed in April Apple Computer Inc. is to adapt the Lisa computer to interface with Cullinet Database Information, which it believes is the best method for connecting a Lisa with data on IBM mainframes.

Following the news that Apple and Cullinet Software have announced the joint development programme to provide a fully integrated solution to business information systems needs, Vic Morris, Cullinet's UK managing director, told *Windfall* how he felt it would help business managers.



Mainframe link-up for Lisa

By VIC MORRIS

THE Cullinet Information Database facility will provide the necessary link to allow managers or end users to have simple access to the information needed to run their business.

It consists of a series of files of summarised management information which may be created from either production data or external sources. These files reside on the mainframe and may be manipulated by all standard relational operators. This data can be easily accessed from Lisa using a variety of non-procedural, menu-driven tools.

With the database, it is now possible to link business managers and decision-makers in any easy-to-use, computerised information network; access summary level information compiled from production data, external sources, and other networked systems; manipulate this data using the graphics on Lisa; create files from a personal computer that can be stored in the information database and later retrieved for personal use or broadcast to other users.

The system provides the essential link necessary to access data from the production database and/or external sources, summarise it and make it accessible to end users from their personal computers.

In addition to serving as a central information resource with a relational file orientation, the information database also collects all of an organisation's personal computers into a single network.

Once documents, messages or graphs have been stored in the information database, they can then be broadcast directly to selected users, greatly facilitating the implementation of electronic mail.

The database may be automatically supplied from a number of sources – the production database on any external or public database (for example, Econometrics, Data Resources, Dow Jones, Bureau of the Census etc) and other networked systems.

Managers and other end users can use their personal computers to access, retrieve, locally store, and then manipulate this database information. The files created as a result of local processing and analysis can be stored back in the information database for distribution to other users or, given the proper security clearances and procedures, they can be used to update the production database.

A good example of its use can be seen from this example. Data from the production general ledger is used to create corporate expense statements. The department head, working from his Lisa, retrieves the expenses related to his department.

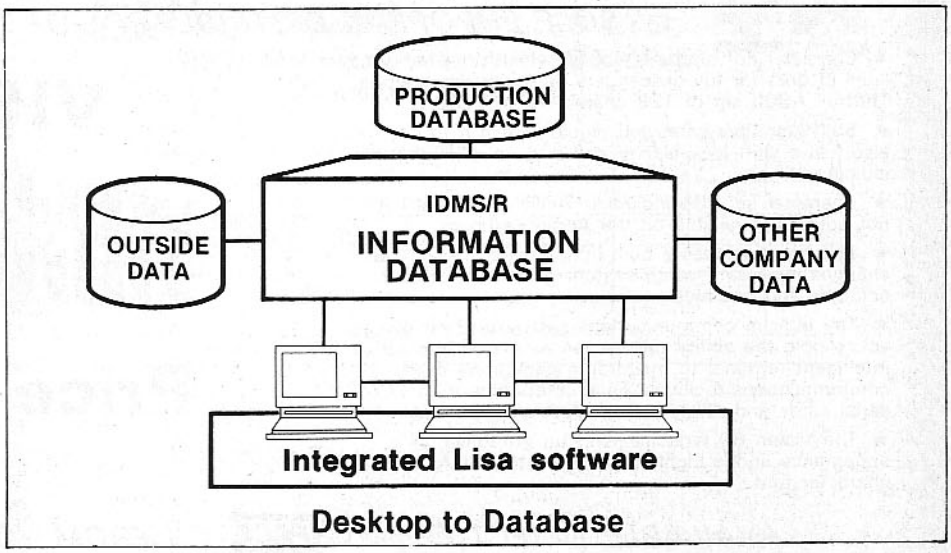
He uses Lisa-Calc to determine trends and project budgets. This may involve access of external data on the information database, such as projected interest rates on the consumer price index.

Graphs showing growth of expenses are sent via electronic mail to the appropriate management personnel. A new table showing next year's projected budgets is stored back on the information database, and the budget information along with that of other departments is used to update the production database for use by corporate financial systems.

The facility has been carefully designed for use by the non-computer manager. All the operations are menu-driven. The system prompts the user with step-by-step directions, and a "Help" key function is supported to provide on-the-spot assistance.

The software typically resides and runs on an organisation's mainframe or on a separate system such as an IBM 4300, dedicated to information database operations. Multiple information databases at widely spread geographical locations can be networked to transfer information between sites, with each system working transparently to the other.

The system has been designed to run on an IBM 360/370, 30XX or 43XX or compatible system under IBM operating systems OS MFT, OS MVT, OS/VS1, OS/VS2 (SVS), OS/VS2 (MVS), DOS/VS, or DOS/VSE.



How the three newly-announced Cullinet products work together

THIS utility program supplies one of the "missing" sets of commands from Apple Basic. With its use text can be written at any point on the hi-res pages using the usual Basic PRINT statement constructions. The text may be written horizontally, vertically or rotated, and may be normal or in inverse.

It will be especially useful to those readers who plot graphs or need hi-res demonstrations such as described in last month's Applecart program, CAT not CAL.

Finally, simple animations can easily be carried out from Basic on the hi-res pages, and an example is given.

GIVE YOUR HI-RES TEXT

ONE utility missing from Applesoft Basic is the ability to put text easily on the hi-res pages at any position. I used to accomplish this using shape tables, but it is slow and difficult, usually involving string manipulation subroutines.

After I had realised how to calculate the base address and required offset for any pixel on the graphics screens (see *Windfall,* January 1983, page 22) it was an easy step to putting text on at any position.

I decided to create the characters as bit patterns based on an eight deep by seven wide array for two reasons. Firstly it was easy because the Apple's text is created thus, enabling just seven bits of a hi-res byte to be significant. Secondly, it would be easy to create reasonable looking lower case characters with descenders.

To this end I decided that the "bottom" row of any upper case character would be full of "nulls". These would allow separation of text between lines and give room for the descenders. Using bit patterns rather than shape table-like structures also has the advantage that manipulations of the text can easily be carried out since a "constant" array structure has to be processed. The big disappointment is that the on-board character generator cannot be accessed from software, and so valuable memory has to be used to store the character set.

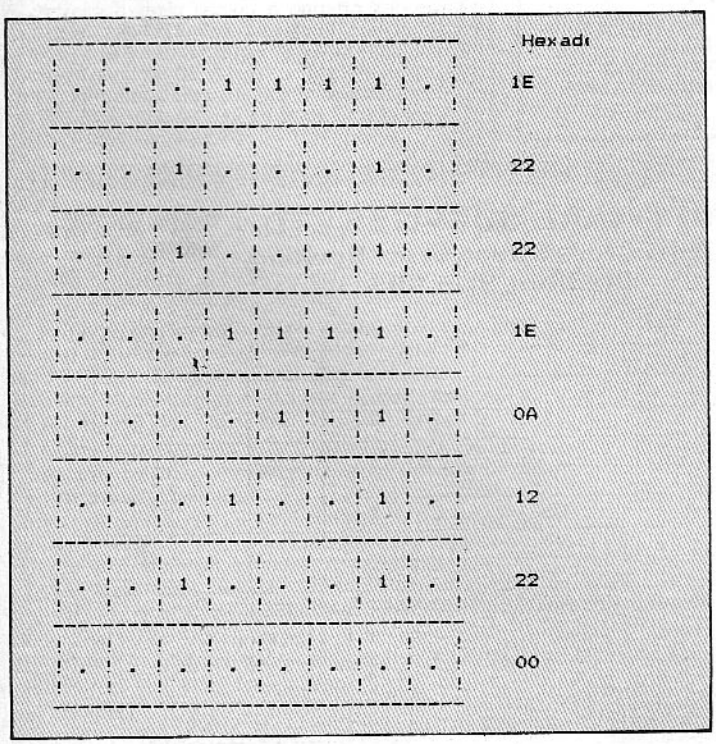


Figure I

My first task therefore was to design the characters. The hi-res screen pixels are controlled by the seven least significant bits of a byte in the correct area of memory. The most significant bit partially controls the colour and as far as text goes was not important to me. The controlling bits are arranged from left to right on the screen so that if set the pixel is lit and if reset the pixel is not lit.

Traditionally numbers are arranged with the more significant digits to the left of the less significant, and so to design the set I simply draw the mirror image of the characters on an eight by eight array, leaving the bottom row and side columns empty. This is shown by considering the letter R. My array was designed as shown in Figure I.

I arranged my bit patterns in Ascii order, with the top row of each pattern at the lower memory location. This enabled me to access any character from its Ascii code by subtracting 32 ($20) and multiplying the result by 8 to give the position in the table of patterns of the top row of the character.

By simply adding the address of the table start and using indirectly addressed indexing I could access any row of the required pattern. For example, the Ascii code of A is $41. Subtracting $20 gives $21 and multiplying this by 8 (three shifts to the left) gives $108. With the program assembled to sit under DOS in a 48K machine the start of the text table is at $9400 and so the top row of letter A is situated at $9508.

I wanted to EOR the patterns on to the hi-res bytes so that animation could be accomplished and I also wanted to be able to stack vertically the text and to rotate it through 90 degrees so that graph axes could be easily labelled as in Figure II, which was created by the example at the end of this article.

I decided therefore to move the byte of each bit pattern into a temporary storage area so that this could be manipulated before EORing with the

A WHIRL

Stringing along the hi-res page with MAX PARROTT

screen. In particular I wanted to be able to put INVERSE text on the screen and to be able to start the text at any screen position, which would necessitate shifting the pattern between bytes.

It was easy to put the patterns into a store for horizontal and vertical text by a straight copy, but a little more difficult to "rotate" the patterns into the same storage area for the rotated text. Finally I decided to copy the pattern into yet another temporary storage area and to manipulate this into the required area of memory. I decided to use the top of page 2 for these temporary areas in order to save memory higher up.

My final problem was to decide on the syntax of the commands. It seemed easiest to use the ubiquitous ampersand for the main commands and to follow this with standard tokenisable Basic commands. I decided to signal whether upper or lower case was to be printed by embedding control characters within the Basic string.

After perusing page 121 of the manual I decided that to print text at screen position x,y (the lower left hand corner of the first character) the commands should be

& PRINT AT x,y, ...

where the dots indicate any legitimate PRINT type statement.

To make all following text appear in inverse the command would be

& INVERSE,

To make the text stack vertically downwards the command would be

& VLIN,

To make the following text rotated the command would be

& ROT=,

Each of these commands would be cleared by the issue of

& NORMAL.

It was clear by this point that quite a lot of memory would be consumed by the program and so I decided to assemble it to sit under DOS and to have two forms, one with lower case ability and one without. It is the latter I have presented here, as generally it is more useful.

In this form the program is BRUNned at $9235, whereupon it sets the ampersand vectors, and sets HIMEM to protect its main routines. The first part of the program is not protected and so if you type in the hexadecimal dump, SAVE it first. The length is $3CB.

I then only had to connect these program parts together so that the Basic line would be properly parsed and error messages given if necessary to complete the program. I used as many Applesoft routines as I could (I am eternally grateful to John Crossley and "The Apple Orchard," Fall 1980, for the entry points of these) for parsing.

The first token on the line of Basic is checked for 'NORMAL', 'VLIN', 'ROT=', or 'INVERSE' and if it is there the appropriate flags are set or reset. If none of these are present SYNCHK (line 94) checks to see if 'PRINT' is present and if it is a check is made for 'AT'. HFNS (line 97) retrieves the starting co-ordinates and the X and Y directions are dealt with as previously described. The presence of a comma is checked and the rest of the line is parsed.

I used FRMEVL to evaluate the formula found. This flags whether the result is a string or a number, VALTYP (line 140) is checked to see which. If it is a number I used FOUT to create a string equal to the

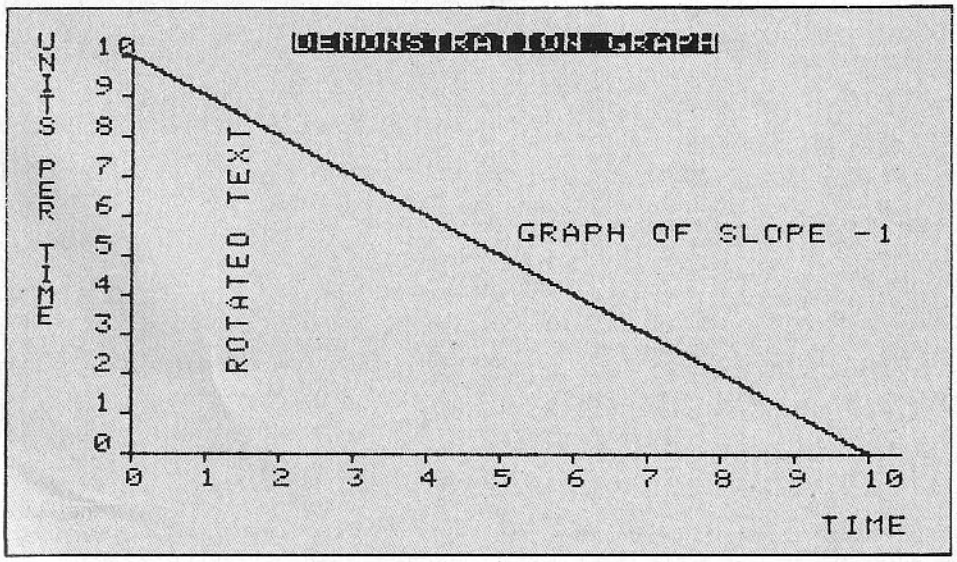


Figure II

GRAPHICS

Floating Point Accumulator at $100-$110 (it leaves a pointer to it in Y,A). I then set DSCTMP+1 to point at it.

If the formula is a string, FRMEVL and the routines it falls into leave a pointer to it in FACMO. I moved this to DSCTMP+1 and JSRed to FREEFAC to free up the temporary string descriptor which had been created.

The string is then dissected, character by character, making checks for zero bytes and double quotes which signify the ends of strings. Any printable character is transformed into the index to the table, the address is looked up and the transfers made.

Its use is best illustrated by two short Basic programs. The first draws a pretend graph and labels the axes and line etc. The second performs a simple piece of animation. Note that writing a string to screen a second time removes it. Both Basic programs assume that the program is resident on disc under the name GRAPHICS TEXT.

```
10  PRINT  CHR$ (4)"BRUN GRAPHICS TEXT"
20  HGR2 : HCOLOR= 3
30  HPLOT 30,10 TO 30,160: HPLOT 30,160 TO 270,160
40  FOR I = 160 TO 10 STEP  - 15: HPLOT 27,I TO 30,I: &  PRINT  AT
18,I,(160-I)/15: NEXT
50  FOR I = 30 TO 260 STEP 23: HPLOT I,160 TO I,163: &  PRINT  AT I -
2,173,(I-30)/23: NEXT
60  &  VLIN : &  PRINT  AT 0,8,"UNITS PER TIME"
70  &  ROT= : &  PRINT  AT 60,130,"ROTATED TEXT"
80  &  NORMAL : &  PRINT  AT 245,191,"TIME"
90  HPLOT 30,10 TO 260,160
100  &  PRINT  AT 150,80,"GRAPH OF SLOPE -1"
110  &  INVERSE : &  PRINT  AT 80,8,"DEMONSTRATION GRAPH"
120  &  NORMAL
130  GET T$: TEXT
```

Program I

```
10 PRINT CHR$(4)"BRUN GRAPHICS TEXT"
20 HGR : HCOLOR= 3:A$ = "HELP"
30 FOR I = 0 TO 279 STEP 7: HPLOT I,0 TO I,161: NEXT
40 FOR I = 0 TO 279 STEP 2
50 & PRINT AT I,50,A$: REM WRITE A$
60 FOR J = 1 TO 20: NEXT
70 & PRINT AT I,50,A$: REM REMOVE A$
80 NEXT
90 GET T$: TEXT
```

Program II

```
0000              1  * Copywrite (C) M.J. Parrott 1983.
0000              2  *
0003              3  Y        EPZ 3
0004              4  BASE     EPZ 4
0006              5  XRES     EPZ 6
0007              6  QUOT     EPZ 7
0008              7  YTEMP    EPZ 8
0009              8  TEMP     EPZ 9
0011              9  VALTYP   EPZ $11
005E             10  INDEX    EPZ $5E
0073             11  HIMEM    EPZ $73
009D             12  DSCTMP   EPZ $9D
00A0             13  FACMO    EPZ $A0
00B1             14  CHRGET   EPZ $B1
00B7             15  CHRGOT   EPZ $B7
00B8             16  TXTPTR   EPZ $B8
00E6             17  HPAG     EPZ $E6
0800             18  *
0800             19  * TOKENS
0800             20  *
008F             21  VLIN     EPZ 143
0098             22  ROT      EPZ 152
009D             23  NORMAL   EPZ 157
009E             24  INVERSE  EPZ 158
00BA             25  PRNTOK   EPZ 186
00C5             26  AT       EPZ 197
0800             27  *
0800             28  * SUBROUTINES ETC.
0800             29  *
02F0             30  TSTORE   EQU $2F0
02F8             31  STORE    EQU $2F8
02EF             32  XSTORE   EQU $2EF
02EE             33  YSTORE   EQU $2EE
03F5             34  AMPER    EQU $3F5
D412             35  ERROR    EQU $D412
D995             36  UPDATE   EQU $D995
DD7B             37  FRMEVL   EQU $DD7B
DEBE             38  CHKCOM   EQU $DEBE
DEC0             39  SYNCHK   EQU $DEC0
E3ED             40  STRLT2   EQU $E3ED
E600             41  FREEFAC  EQU $E600
ED34             42  FOUT     EQU $ED34
F6B9             43  HFNS     EQU $F6B9
0800             44  *
0800             45  *SET UP AMPERSAND
0800             46  *
9235             47           ORG $9235
9235             48           OBJ $1000
9235 A9 4C       49           LDA #$4C          ;JMP COMMAND
9237 8D F5 03    50           STA AMPER
923A A9 49       51           LDA #START
923C 8D F6 03    52           STA AMPER+1
923F 85 73       53           STA HIMEM
9241 A9 92       54           LDA /START
9243 8D F7 03    55           STA AMPER+2
9246 85 74       56           STA HIMEM+1
9248 60          57           RTS               ;AND BACK TO BASIC
9249             58  START:
9249 C9 9D       59           CMP #NORMAL       ;IS FIRST CHAR THE
924B D0 0D       60           BNE ROTIT         ;NORMAL TOKEN?
924D A9 00       61           LDA #0
924F 8D 86 92    62           STA IFLAG
9252 8D 88 92    63           STA VFLAG
9255 8D 87 92    64           STA RFLAG
9258 F0 29       65           BEQ RETURN
925A             66  ROTIT:
925A C9 8F       67           CMP #VLIN
925C D0 0C       68           BNE ROTT
925E A9 00       69           LDA #0
9260 8D 87 92    70           STA RFLAG
9263 A9 FF       71           LDA #$FF
9265 8D 88 92    72           STA VFLAG
9268 D0 19       73           BNE RETURN
926A             74  ROTT:
926A C9 98       75           CMP #ROT
926C D0 0C       76           BNE INV
926E A9 FF       77           LDA #$FF
9270 8D 87 92    78           STA RFLAG
9273 A9 00       79           LDA #0
9275 8D 88 92    80           STA VFLAG         ;ZERO IT IN CASE
9278 F0 09       81           BEQ RETURN
927A             82  INV:
927A C9 9E       83           CMP #INVERSE
927C D0 0B       84           BNE PRINT
927E A9 7F       85           LDA #$7F
9280 8D 86 92    86           STA IFLAG
9283             87  RETURN:
9283 4C 95 D9    88           JMP UPDATE        ;MOVE TXTPTR AND RETURN
9286             89  IFLAG    DFS 1,0
9287             90  RFLAG    DFS 1,0
9288             91  VFLAG    DFS 1,0
9289             92  PRINT:
9289 A9 BA       93           LDA #PRNTOK       ;CHECK FOR PRINT
928B 20 C0 DE    94           JSR SYNCHK        ;TOKEN
928E A9 C5       95           LDA #AT           ;AND FOR AT TOKEN
9290 20 C0 DE    96           JSR SYNCHK
9293 20 B9 F6    97           JSR HFNS          ;GET PLOTTING COORDS
9296 85 03       98           STA Y             ;STORE THE Y COORD
9298 86 04       99           STX BASE          ;AND LSB OF X COORD
929A            100  *
929A            101  * CALC STARTING COORDS FOR STRING
929A            102  *
929A 98         103           TYA               ;TAKE THE MSB OF X COORD
929B A0 07      104           LDY #7            ;AND DIVIDE BY 7 TO CALC
929D 84 06      105           STY XRES          ;BYTE FOR PLOTTING OFFSET
929F 38         106           SEC
92A0 E5 06      107           SBC XRES
92A2 08         108  LOOP     PHP
92A3 26 07      109           ROL QUOT
92A5 06 04      110           ASL BASE
92A7 2A         111           ROL
92A8 28         112           PLP
92A9 90 05      113           BCC ADD
92AB E5 06      114           SBC XRES
92AD 4C B2 92   115           JMP NEXT
92B0 65 06      116  ADD      ADC XRES
92B2 88         117  NEXT     DEY
92B3 10 ED      118           BPL LOOP
92B5 B0 03      119           BCS LAST
92B7 65 06      120           ADC XRES
92B9 18         121           CLC
92BA 26 07      122  LAST     ROL QUOT          ;ON EXIT QUOT HAS #BYTE
92BC 85 06      123           STA XRES          ;& XRES HAS #BIT
92BE A5 07      124           LDA QUOT          ;CHECK TO SEE IF ON SCREEN
92C0 10 05      125           BPL CHECK
92C2 A2 35      126  ERR      LDX #53           ;'ILLEGAL QUANTITY'
92C4 4C 12 D4   127           JMP ERROR
92C7 C9 28      128  CHECK    CMP #40           ;IF OVER 39 ERROR
92C9 B0 F7      129           BCS ERR
92CB A5 03      130           LDA Y             ;GET THE Y COORD
92CD C9 C0      131           CMP #192          ;IS IT ON SCREEN?
92CF B0 F1      132           BCS ERR
```

GRAPHICS

```
92D1 E6 03      133          INC Y              ;USED LATER
92D3            134  *
92D3            135  * PARSE REST OF LINE
92D3            136  *
92D3 20 BE DE   137          JSR CHKCOM         ;IS IT A COMMA NEXT?
92D6            138  PRINTIT:
92D6 20 7B DD   139          JSR FRMEVL         ;EVALUATE WHAT'S THERE
92D9 24 11      140          BIT VALTYP         ;STRING OR NUMBER?
92DB 30 0A      141          BMI STR
92DD 20 34 ED   142          JSR FOUT           ;CREATE STRING FROM FAC
92E0 85 9E      143          STA DSCTMP+1       ;SET POINTER TO IT
92E2 84 9F      144          STY DSCTMP+2
92E4 4C F4 92   145          JMP STRING1        ;NOW PRINT IT
92E7            146  STR:
92E7 A0 02      147          LDY #2             ;NEED TO SET DSCTMP
92E9 B1 A0      148  TRANS    LDA (FACMO),Y      ;FROM FACMO,LO
92EB 99 9D 00   149          STA DSCTMP,Y
92EE 88         150          DEY
92EF D0 F8      151          BNE TRANS          ;DON'T NEED LENGTH
92F1 20 00 E6   152          JSR FREEFAC        ;FREE TEMP DESCRIPTOR
92F4            153  STRING1:
92F4 20 0A 93   154          JSR WRITE1         ;PUT IT ON SCREEN
92F7            155  WHTNXT1:
92F7 20 B7 00   156          JSR CHRGOT         ;CHECK THE CHAR
92FA            157  WHTNXT:
92FA F0 0D      158          BEQ NOMORE
92FC C9 2C      159          CMP #$2C           ;IS IT A COMMA?
92FE F0 04      160          BEQ MVIT
9300 C9 3B      161          CMP #$3B           ;IS IT A ;
9302 D0 D2      162          BNE PRINTIT
9304            163  MVIT:
9304 20 B1 00   164          JSR CHRGET         ;GET NEXT CHAR
9307 D0 F1      165          BNE WHTNXT
9309            166  NOMORE:
9309 60         167          RTS                ;BACK TO BASIC
930A            168  WRITE1:
930A A0 00      169          LDY #0             ;INIT CHAR COUNTER
930C            170  PRINT2:
930C A5 03      171          LDA Y              ;GET COORD FOR BOTTOM
930E 85 0B      172          STA YTEMP
9310 A9 00      173          LDA #0             ;CALC INDEX OF CHAR
9312 85 5F      174          STA INDEX+1
9314 8C EE 02   175          STY YSTORE         ;STORE CHAR COUNTER
9317 B1 9E      176          LDA (DSCTMP+1),Y   ;GET ASCII OF CHAR
9319 D0 01      177          BNE YES1
931B            178  NO1:
931B 60         179          RTS                ;BACK TO CALLER
931C            180  YES1:
931C C9 22      181          CMP #$22           ;STOP ON A QUOTE
931E F0 FB      182          BEQ NO1
9320 38         183          SEC
9321 E9 20      184          SBC #$20           ;PUT IN RANGE 0 TO $5F
9323 85 5E      185          STA INDEX
9325 A2 02      186          LDX #2             ;TO MULTIPLY BY 8
9327            187  MULT:
9327 06 5E      188          ASL INDEX          ;LEAVING RESULT IN INDEX
9329 26 5F      189          ROL INDEX+1
932B CA         190          DEX
932C 10 F9      191          BPL MULT
932E 18         192          CLC
932F A5 5E      193          LDA INDEX          ;ADD THE ADDRESS OF TABLE
9331 69 00      194          ADC #TEXT          ;START
9333 85 5E      195          STA INDEX
9335 A5 5F      196          LDA INDEX+1
9337 69 94      197          ADC /TEXT
9339 85 5F      198          STA INDEX+1
933B A0 07      199          LDY #7             ;COUNTER FOR COLUMNS
933D A2 07      200          LDX #7             ;AND ROWS
933F AD B7 92   201          LDA RFLAG          ;WHAT'S ROT?
9342 F0 1E      202          BEQ HORIZ
9344            203  PUTIN:
9344 B1 5E      204          LDA (INDEX),Y      ;USE TEMP STORAGE
9346 99 F0 02   205          STA TSTORE,Y
9349 88         206          DEY
934A 10 F8      207          BPL PUTIN
934C A2 07      208          LDX #7             ;COUNTER FOR BYTES
934E A0 07      209  BYTES    LDY #7             ;COUNTER FOR BITS
9350 B9 F0 02   210  BITS     LDA TSTORE,Y       ;GET PATTERN
9353 4A         211          LSR
9354 99 F0 02   212          STA TSTORE,Y
9357 3E F8 02   213          ROL STORE,X
935A 88         214          DEY
935B 10 F3      215          BPL BITS
935D CA         216          DEX
935E 10 EE      217          BPL BYTES
9360 30 08      218          BMI WRITE
9362            219  HORIZ:
9362 B1 5E      220          LDA (INDEX),Y      ;TRANSFER BIT PATTERN
9364 99 F8 02   221          STA STORE,Y        ;TO THE STORE AREA
9367 88         222          DEY
9368 10 F8      223          BPL HORIZ
936A            224  WRITE:
936A A2 07      225          LDX #7
936C            226  LOOP1:
936C A4 08      227          LDY YTEMP          ;GET THE Y COORD FOR THE
PATTERN
936E 88         228          DEY                ;MOVE UP ONE ROW
936F 84 08      229          STY YTEMP          ;SAVE IT
9371 98         230          TYA
9372 20 CB 93   231          JSR YCALC          ;CALC NEW ADDRESS BASE
9375 A9 00      232          LDA #0             ;INIT TEMP TO
9377 85 09      233          STA TEMP           ;ZEROS
9379 AD B6 92   234          LDA IFLAG          ;INVERSE WANTED?
937C 5D F8 02   235          EOR STORE,X        ;SUPERIMPOSE PATTERN
937F 8E EF 02   236          STX XSTORE         ;SAVE ROW COUNTER
9382 A6 06      237          LDX XRES           ;WITH #BITS REQUIRED
9384 F0 07      238          BEQ NO             ;FOR THIS BYTE OF HIRES
9386            239  SHIFT:
9386 0A         240          ASL                ;MOVE PATTERN AS REQUIRED
9387 26 09      241          ROL TEMP           ;INTO TEMP AS WELL AS A
9389 CA         242          DEX
938A 10 FA      243          BPL SHIFT          ;ONCE MORE FOR BIT 7
938C 4A         244          LSR                ;AND SHIFT ACC BACK TO CLEAR 7
938D            245  NO:
938D A4 07      246          LDY QUOT           ;GET THE OFFSET FOR THE BYTE
938F 51 04      247          EOR (BASE),Y       ;ADJUST BYTE
9391 91 04      248          STA (BASE),Y       ;AND PUT IT IN
9393 C8         249          INY
9394 C0 28      250          CPY #40
9396 90 01      251          BCC GOON
9398            252  END:
9398 60         253          RTS                ;BACK TO BASIC
9399            254  GOON:
9399 A5 09      255          LDA TEMP           ;GET PATTERN
939B 51 04      256          EOR (BASE),Y       ;FORCE THE PATTERN
939D 91 04      257          STA (BASE),Y       ;AND PUT ON SCREEN
939F AE EF 02   258          LDX XSTORE         ;RESTORE ROW COUNTER
93A2 CA         259          DEX                ;MOVE UP ONE ROW
93A3 10 C7      260          BPL LOOP1          ;FINISH 8 ROWS
93A5 AD B7 92   261          LDA RFLAG          ;WHAT'S ROT?
93A8 F0 07      262          BEQ VRT
93AA A5 08      263          LDA YTEMP
93AC 85 03      264          STA Y
93AE 4C C4 93   265          JMP NEXT1
93B1            266  VRT:
93B1 AD B8 92   267          LDA VFLAG
93B4 F0 09      268          BEQ HORIZ1
93B6 18         269          CLC
93B7 A5 03      270          LDA Y
93B9 69 08      271          ADC #8
93BB 85 03      272          STA Y
93BD D0 05      273          BNE NEXT1
93BF            274  HORIZ1:
93BF A4 07      275          LDY QUOT           ;GET OFFSET FOR 1ST BYTE
93C1 C8         276          INY                ;SAFE AS CHECKED BEFORE
93C2 84 07      277          STY QUOT
93C4            278  NEXT1:
93C4 AC EE 02   279          LDY YSTORE         ;GET STRING COUNTER
93C7 C8         280          INY
93C8 4C 0C 93   281          JMP PRINT2         ;NO! SO BACK FOR NEXT CHAR
93CB            282  YCALC:
93CB C9 C0      283          CMP #192           ;IS IT ON SCREEN?
93CD 90 03      284          BCC YES
93CF 68         285          PLA                ;DISCARD A WORD
93D0 68         286          PLA
93D1 60         287          RTS                ;BACK TO BASIC
93D2            288  YES:
93D2 29 30      289          AND #$30           ;Y COORD ALREADY IN A
93D4 4A         290          LSR
93D5 4A         291          LSR
93D6 4A         292          LSR
93D7 4A         293          LSR
93D8 05 E6      294          ORA HPAG           ;WHICH PAGE?
93DA 85 05      295          STA BASE+1
93DC A5 08      296          LDA YTEMP
93DE 29 07      297          AND #$7
93E0 0A         298          ASL
93E1 0A         299          ASL
93E2 65 05      300          ADC BASE+1
93E4 85 05      301          STA BASE+1
93E6 A5 08      302          LDA YTEMP
93E8 29 C0      303          AND #$C0
93EA 4A         304          LSR
93EB 85 04      305          STA BASE
93ED 4A         306          LSR
93EE 4A         307          LSR
93EF 05 04      308          ORA BASE
93F1 85 04      309          STA BASE
93F3 A5 08      310          LDA YTEMP
93F5 29 08      311          AND #$8
93F7 F0 06      312          BEQ DONE
93F9 A9 80      313          LDA #$80
93FB 65 04      314          ADC BASE
93FD 85 04      315          STA BASE
93FF            316  DONE:
93FF 60         317          RTS
9400            318  TEXT:
9400 00 00 00   319          HEX 0000000000000000 ;space
```

GRAPHICS

```
9403 00 00 00
9406 00 00
9408 08 08 08   320        HEX 0808080808000800 ;!
940B 08 08 00
940E 08 00
9410 14 14 14   321        HEX 1414140000000000 ;"
9413 00 00 00
9416 00 00
9418 14 14 3E   322        HEX 14143E143E141400 ;#
941B 14 3E 14
941E 14 00
9420 08 3C 0A   323        HEX 083C0A1C281E0800 ;$
9423 1C 28 1E
9426 08 00
9428 06 26 10   324        HEX 0626100804323000 ;%
942B 08 04 32
942E 30 00
9430 04 0A 0A   325        HEX 040A0A042A122C00 ;&
9433 04 2A 12
9436 2C 00
9438 08 08 00   326        HEX 0808000000000000 ;'
943B 00 00 00
943E 00 00
9440 08 04 02   327        HEX 0804020202040800 ;(
9443 02 02 04
9446 08 00
9448 08 10 20   328        HEX 0810202020100800 ;)
944B 20 20 10
944E 08 00
9450 08 2A 1C   329        HEX 082A1C081C2A0800 ;*
9453 08 1C 2A
9456 08 00
9458 00 08 08   330        HEX 0008083E08080000 ;+
945B 3E 08 08
945E 00 00
9460 00 00 00   331        HEX 0000000008080400 ;,
9463 00 08 08
9466 04 00
9468 00 00 00   332        HEX 0000003E00000000 ;-
946B 3E 00 00
946E 00 00
9470 00 00 00   333        HEX 0000000000000800 ;.
9473 00 00 00
9476 08 00
9478 00 20 10   334        HEX 0020100804020000 ;/
947B 08 04 02
947E 00 00
9480 1C 22 32   335        HEX 1C22322A26221C00 ;0
9483 2A 26 22
9486 1C 00
9488 08 0C 08   336        HEX 080C0808080081C00 ;1
948B 08 08 08
948E 1C 00
9490 1C 22 20   337        HEX 1C22201804023E00 ;2
9493 18 04 02
9496 3E 00
9498 3E 20 10   338        HEX 3E20101820221C00 ;3
949B 18 20 22
949E 1C 00
94A0 10 18 14   339        HEX 101814123E101000 ;4
94A3 12 3E 10
94A6 10 00
94A8 3E 02 1E   340        HEX 3E021E2020221C00 ;5
94AB 20 20 22
94AE 1C 00
94B0 38 04 02   341        HEX 3804021E22221C00 ;6
94B3 1E 22 22
94B6 1C 00
94B8 3E 20 10   342        HEX 3E20100804040400 ;7
94BB 08 04 04
94BE 04 00
94C0 1C 22 22   343        HEX 1C22221C22221C00 ;8
94C3 1C 22 22
94C6 1C 00
94C8 1C 22 22   344        HEX 1C22223C20100E00 ;9
94CB 3C 20 10
94CE 0E 00
94D0 00 00 08   345        HEX 0000080008000000 ;:
94D3 00 08 00
94D6 00 00
94D8 00 00 08   346        HEX 0000080008080400 ;;
94DB 00 08 08
94DE 04 00
94E0 10 08 04   347        HEX 1008040204081000 ;<
94E3 02 04 08
94E6 10 00
94E8 00 00 3E   348        HEX 00003E003E000000 ;=
94EB 00 3E 00
94EE 00 00
94F0 04 08 10   349        HEX 0408102010080400 ;>
94F3 20 10 08
94F6 04 00
94F8 1C 22 10   350        HEX 1C22100808000800 ;?
94FB 08 08 00
94FE 08 00
9500 1C 22 2A   351        HEX 1C222A2A1A023C00 ;@
9503 2A 1A 02
9506 3C 00
9508 08 14 22   352        HEX 0814222223E222200 ;A
950B 22 3E 22
950E 22 00
9510 1E 22 22   353        HEX 1E22221E22221E00 ;B
9513 1E 22 22
9516 1E 00
9518 1C 22 02   354        HEX 1C220202022210C00 ;C
951B 02 02 22
951E 1C 00
9520 1E 22 22   355        HEX 1E222222222221E00 ;D
9523 22 22 22
9526 1E 00
9528 3E 02 02   356        HEX 3E02021E02023E00 ;E
952B 1E 02 02
952E 3E 00
9530 3E 02 02   357        HEX 3E02021E02020200 ;F
9533 1E 02 02
9536 02 00
9538 3C 02 02   358        HEX 3C020202322230C00 ;G
953B 02 32 22
953E 3C 00
9540 22 22 22   359        HEX 2222223E22222200 ;H
9543 3E 22 22
9546 22 00
9548 1C 08 08   360        HEX 1C0808080808 1C00 ;I
954B 08 08 08
954E 1C 00
9550 20 20 20   361        HEX 2020202020221C00 ;J
9553 20 20 22
9556 1C 00
9558 22 12 0A   362        HEX 22120A060A122200 ;K
955B 06 0A 12
955E 22 00
9560 02 02 02   363        HEX 02020202023E00 ;L
9563 02 02 02
9566 3E 00
9568 22 36 2A   364        HEX 22362A2222222200 ;M
956B 22 22 22
956E 22 00
9570 22 22 26   365        HEX 2222262A32222200 ;N
9573 2A 32 22
9576 22 00
9578 1C 22 22   366        HEX 1C222222222221C00 ;O
957B 22 22 22
957E 1C 00
9580 1E 22 22   367        HEX 1E22221E02020200 ;P
9583 1E 02 02
9586 02 00
9588 1C 22 22   368        HEX 1C2222222A225C00 ;Q
958B 22 2A 22
958E 5C 00
9590 1E 22 22   369        HEX 1E22221E0A122200 ;R
9593 1E 0A 12
9596 22 00
9598 1C 22 02   370        HEX 1C22021C20221C00 ;S
959B 1C 20 22
959E 1C 00
95A0 3E 08 08   371        HEX 3E0808080808 0800 ;T
95A3 08 08 08
95A6 08 00
95A8 22 22 22   372        HEX 2222222222221C00 ;U
95AB 22 22 22
95AE 1C 00
95B0 22 22 22   373        HEX 2222222222140800 ;V
95B3 22 22 14
95B6 08 00
95B8 22 22 22   374        HEX 2222222A2A362200 ;W
95BB 2A 2A 36
95BE 22 00
95C0 22 22 14   375        HEX 2222140814222200 ;X
95C3 08 14 22
95C6 22 00
95C8 22 22 14   376        HEX 2222140808080800 ;Y
95CB 08 08 08
95CE 08 00
95D0 3E 20 10   377        HEX 3E20100804023E00 ;Z
95D3 08 04 02
95D6 3E 00
95D8 3E 06 06   378        HEX 3E06060606063E00 ;[
95DB 06 06 06
95DE 3E 00
95E0 00 02 04   379        HEX 0002040810200000 ;backslash
95E3 08 10 20
95E6 00 00
95E8 3E 30 30   380        HEX 3E30303030303E00 ;]
95EB 30 30 30
95EE 3E 00
95F0 00 08 14   381        HEX 0008142200000000 ;^
95F3 22 00 00
95F6 00 00
95F8 00 00 00   382        HEX 0000000000003E00 ;_
95FB 00 00 00
95FE 3E 00
9600            383        END
```

GRAPHICS

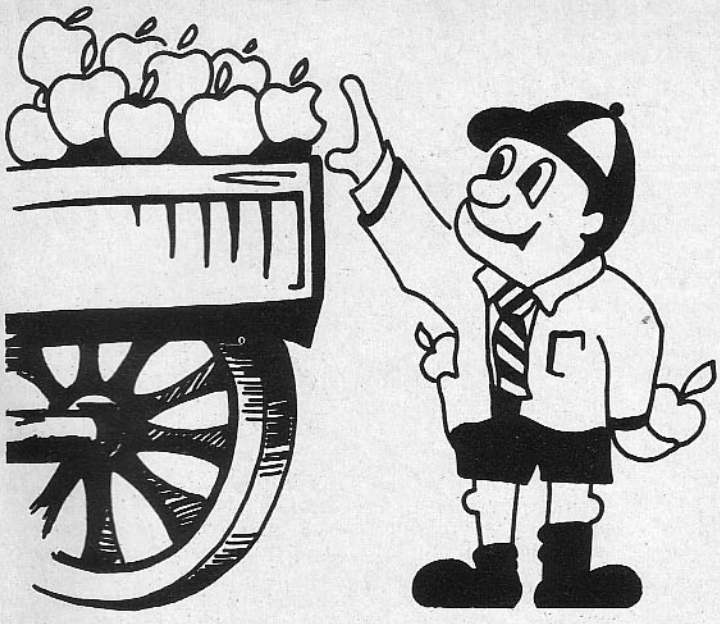
HEXADECIMAL DUMP OF
THE CHARACTER SET

```
*9400.95FF
9400- 00 00 00 00 00 00 00 00
9408- 08 08 08 08 08 00 08 00
9410- 14 14 14 00 00 00 00 00
9418- 14 14 3E 14 3E 14 14 00
9420- 08 3C 0A 1C 28 1E 08 00
9428- 06 26 10 08 04 32 30 00
9430- 04 0A 0A 04 2A 12 2C 00
9438- 08 08 00 00 00 00 00 00
9440- 08 04 02 02 02 04 08 00
9448- 08 10 20 20 20 10 08 00
9450- 08 2A 1C 08 1C 2A 08 00
9458- 00 08 08 3E 08 08 00 00
9460- 00 00 00 00 08 08 04 00
9468- 00 00 00 3E 00 00 00 00
9470- 00 00 00 00 00 00 08 00
9478- 00 20 10 08 04 02 00 00
9480- 1C 22 32 2A 26 22 1C 00
9488- 08 0C 08 08 08 08 1C 00
9490- 1C 22 20 18 04 02 3E 00
9498- 3E 20 10 18 20 22 1C 00
94A0- 10 18 14 12 3E 10 10 00
94A8- 3E 02 1E 20 20 22 1C 00
94B0- 38 04 02 1E 22 22 1C 00
94B8- 3E 20 10 08 04 04 04 00
94C0- 1C 22 22 1C 22 22 1C 00
94C8- 1C 22 22 3C 20 10 0E 00
94D0- 00 00 08 00 08 00 00 00
94D8- 00 00 08 00 08 08 04 00
94E0- 10 08 04 02 04 08 10 00
94E8- 00 00 3E 00 3E 00 00 00
94F0- 04 08 10 20 10 08 04 00
94F8- 1C 22 10 08 08 00 08 00
9500- 1C 22 2A 2A 1A 02 3C 00
9508- 08 14 22 22 3E 22 22 00
9510- 1E 22 22 1E 22 22 1E 00
9518- 1C 22 02 02 02 22 1C 00
9520- 1E 22 22 22 22 22 1E 00
9528- 3E 02 02 1E 02 02 3E 00
9530- 3E 02 02 1E 02 02 02 00
9538- 3C 02 02 02 32 22 3C 00
9540- 22 22 22 3E 22 22 22 00
9548- 1C 08 08 08 08 08 1C 00
9550- 20 20 20 20 20 22 1C 00
9558- 22 12 0A 06 0A 12 22 00
9560- 02 02 02 02 02 02 3E 00
9568- 22 36 2A 22 22 22 22 00
9570- 22 22 26 2A 32 22 22 00
9578- 1C 22 22 22 22 22 1C 00
9580- 1E 22 22 1E 02 02 02 00
9588- 1C 22 22 22 2A 22 5C 00
9590- 1E 22 22 1E 0A 12 22 00
9598- 1C 22 02 1C 20 22 1C 00
95A0- 3E 08 08 08 08 08 08 00
95A8- 22 22 22 22 22 22 1C 00
95B0- 22 22 22 22 22 14 08 00
95B8- 22 22 22 2A 2A 36 22 00
95C0- 22 22 14 08 14 22 22 00
95C8- 22 22 14 08 08 08 08 00
95D0- 3E 20 10 08 04 02 3E 00
95D8- 3E 06 06 06 06 06 3E 00
95E0- 00 02 04 08 10 20 00 00
95E8- 3E 30 30 30 30 30 3E 00
95F0- 00 08 14 22 00 00 00 00
95F8- 00 00 00 00 00 00 3E 00
```

HEXADECIMAL DUMP OF THE PROGRAM

```
*9235.93FF

9235- 55 4D 50
9238- 20 4F 46 20 54 48 45 20
9240- 50 52 4F 47 52 41 4D 04
9248- 20 C9 9D D0 0D A9 00 8D
9250- 86 92 8D 88 92 8D 87 92
9258- F0 29 C9 8F D0 0C A9 00
9260- 8D 87 92 A9 FF 8D 88 92
9268- D0 19 C9 98 D0 0C A9 FF
9270- 8D 87 92 A9 00 8D 88 92
9278- F0 09 C9 9E D0 0B A9 7F
9280- 8D 86 92 4C 95 D9 00 00
9288- 00 A9 BA 20 C0 DE A9 C5
9290- 20 C0 DE 20 B9 F6 85 03
9298- 86 04 98 A0 07 84 06 38
92A0- E5 06 08 26 07 06 04 2A
92A8- 28 90 05 E5 06 4C B2 92
92B0- 65 06 88 10 ED B0 03 65
92B8- 06 18 26 07 85 06 A5 07
92C0- 10 05 A2 35 4C 12 D4 C9
92C8- 28 B0 F7 A5 03 C9 C0 B0
92D0- F1 E6 03 20 BE DE 20 7B
92D8- DD 24 11 30 0A 20 34 ED
92E0- 85 9E 84 9F 4C F4 92 A0
92E8- 02 B1 A0 99 9D 00 88 D0
92F0- F8 20 00 E6 20 0A 93 20
92F8- B7 00 F0 0D C9 2C F0 04
9300- C9 3B D0 D2 20 B1 00 D0
9308- F1 60 A0 00 A5 03 85 08
9310- A9 00 85 5F 8C EE 02 B1
9318- 9E D0 01 60 C9 22 F0 FB
9320- 38 E9 20 85 5E A2 02 06
9328- 5E 26 5F CA 10 F9 18 A5
9330- 5E 69 00 85 5E A5 5F 69
9338- 94 85 5F A0 07 A2 07 AD
9340- 87 92 F0 1E B1 5E 99 F0
9348- 02 88 10 F8 A2 07 A0 07
9350- B9 F0 02 4A 99 F0 02 3E
9358- F8 02 88 10 F3 CA 10 EE
9360- 30 08 B1 5E 99 F8 02 88
9368- 10 F8 A2 07 A4 08 88 84
9370- 08 98 20 CB 93 A9 00 85
9378- 09 AD 86 92 5D F8 02 8E
9380- EF 02 A6 06 F0 07 0A 26
9388- 09 CA 10 FA 4A A4 07 51
9390- 04 91 04 C8 C0 28 90 01
9398- 60 A5 09 51 04 91 04 AE
93A0- EF 02 CA 10 C7 AD 87 92
93A8- F0 07 A5 08 85 03 4C C4
93B0- 93 AD 88 92 F0 09 18 A5
93B8- 03 69 08 85 03 D0 05 A4
93C0- 07 C8 84 07 AC EE 02 C8
93C8- 4C 0C 93 C9 C0 90 03 68
93D0- 68 60 29 30 4A 4A 4A 4A
93D8- 05 E6 85 05 A5 08 29 07
93E0- 0A 0A 65 05 85 05 A5 08
93E8- 29 C0 4A 85 04 4A 4A 05
93F0- 04 85 04 A5 08 29 08 F0
93F8- 06 A9 80 65 04 85 04 60
```



Applecart

Monthly review of
Apple in education

Part One

Logo is more than just child's play

By CHRISTOPHER ROPER

WHAT is Logo and who should use it? This introductory article in our new series on the language answers these questions, explains how the Turtle got its name and argues that far from being a toy language, logo has an important role to play both now and in the future.

PROFESSOR Seymour Papert looks like a rumpled gnome, peering over his glasses, and has the gift of talking to nine-year-old children as if they were his intellectual equals.

Although he holds chairs in both education and mathematics at the Massachussets Institute of Technology, his fame in the world rests principally on the programming language Logo, which he first developed with several associates in Boston 16 years ago.

Although not the sole designer of Logo, Papert popularised its message with a book called *Mindstorms – Children, Computers and Powerful Ideas,* which argued that Logo allowed children to appropriate computer culture as part of their general educational development. Instead of a diet of video games or drill and practice programmes, Papert proposed that children of five or six could learn to program computers using Logo.

This led some people to suppose that Logo was a toy language which belonged in the primary school and nowhere else. In fact, while Papert and his colleagues were working with very young children, researchers at Edinburgh University were using Logo in their artificial intelligence work.

Logo is sometimes referred to as a sub-set of Lisp, although in some ways it is an improvement on Lisp. Its syntax is much clearer and programs are more readable.

The key idea of Lisp is that all data is represented as atoms and lists. The lists may be lists of words, numbers, letters, or lists, and indeed you can have lists of lists of lists. For reasons associated with the first machine implementation of Lisp, the first element of a Lisp list is known as CAR and the rest of it is known as CDR.

Lists can be chopped up, examined, sorted, merged, purged and generally rearranged by using CAR and CDR as functions. CAR returns the first word of a list; CDR, pronounced "Cooder", returns the rest of it. Logo has these same functions, but calls them FIRST and BUTFIRST.

If I seem to dwell on the more esoteric list processing functions of Logo, it is because I want to avoid a mailbag full of letters asking why *Windfall* doesn't send Christopher Roper back to the nursery world of sandpits and wooden bricks, where he clearly belongs.

If readers have heard of Logo before, it is usually in connection with Turtle Graphics. One of Papert's key contributions to the development of logo was to see that children needed a conceptual bridge between the material world of their physical experiences and the formal and abstract world of a computer program.

He found his bridge in a wheeled robot, which accepted instructions as follows:

Forward 100 (units)
Right 90 (degrees)

Recalling the robot tortoises of the British neurophysiologist Gray Walters, Papert called his robot a turtle. It represented a sharp break with the model of geometry taught to generations of school children.

Instead of drawing abstract networks of lines on a piece of paper, Papert invited children to see a geometrical figure as paths followed by the turtle. He asked children to play turtle by walking through

→

the same figures, shouting the same commands to one another as those they used to program the robot.

The floor turtle has been replaced by a screen turtle, a spot of light on the monitor screen. But research workers have discovered that children still find the image of a turtle, which they can control and direct, provides the all important bridge from one world into another.

The idea was so powerful that it was used in other contexts. Apple Pascal uses turtle graphics and the new book on Apple Pascal by Ian MacCullum of Essex University uses turtle graphics to introduce the language. MacCullum's course certainly isn't aimed at children.

Turtle graphics can be implemented in any programming language, and if that were all that Logo was about, it would belong in the museum of good ideas, with a meritorious past but no discernible future. The fact that Gary Kildare, author of the CP/M operating system and president of Digital Research, has himself produced an implementation of Logo for the IBM PC, suggests that Logo does have a future. The fact that Kildare sees Logo as the ideal applications programming language for businessmen, suggests that we should all pay careful attention.

There are two good reasons, along with several bad ones, why Logo has not previously been advocated as a general purpose programming language like Basic, Pascal, BCPL, or Forth.

The valid reasons are that Logo, like Lisp, is profligate in its use of memory. Its LIST processing operations build up stacks of data in the memory, and can soon overflow the available space.

The process whereby the program goes through the computer's memory, freeing up cells for reuse, is known as recycling in Logo, and more picturesquely as "garbage collection" in Lisp – one of the few instances in which I prefer the Lisp idiom. The second associated reason is that most Logo implementations are slow, even compared to a good Basic.

The bad reasons are, first, that the communities of Logo users in the 1970s, the artifical intelligence mafia and the professional educationalists, were not particularly interested in pushing Logo as a programming language. (No entrepreneur moved in to take up the cause of Logo. The language did not have its Ken Bowles, the professor at the University of California who made Pascal a global phenomenon on the back of the UCSD p-system.)

Secondly, both computer users and manufacturers were wedded to a view of Logo as a kind of intellectual Lego.

The bad reasons now belong to the past. Byte magazine devoted its August 1982 issue to Logo, introducing the language to a new market of adult micro users, and Logo Computer Systems Inc (LCSI) was formed in Montreal to give the language commercial impetus. There are now enough good books and articles available to correct the earlier misconceptions of Logo, though it is true to say that most of those available are written with an educationalists bias.

The good reasons for Logo's late challenge in the programming language stakes will take longer to overcome. There is now doubt that if you want to write a major applications package for your office, you are better off with Pascal, or even Basic. If you want to write commercial software for the Apple II, you should use Forth, CisCobol or an assembler. The Apple II may never be the right vehicle for Logo.

One interesting fact though, is that the new Atari Logo, also written by LCSI and also using the 6502 microprocessor, is four times as fast as the LCSI Apple II Logo.

It also has four user-definable turtles, which allows animation of a quality which I had never seen from a high level language on a micro.

The Sinclair Spectrum Logo, also written by LCSI, will also offer a better product, with more workspace, than is currently available with Apple II Logo.

The technique of writing an efficient Logo interpreter has advanced substantially over the past three years, and Apple is unlikely to allow itself to be left behind. If Dr Logo (as Digital Research's offering is quaintly known) succeeds in its chosen market, then surely there will be a powerful Logo available for the Lisa – and presumably for Mackintosh, too, when it finally appears on the scene.

A second development is likely to advance the cause of Logo. Basic arithmetical functions, such as addition, subtraction, multiplication, and division have long been taken care of by hardware, and new chips handle increasingly complex mathematical functions. There is no reason why special processors could not be developed to take care of the list processing functions of Lisp and Logo.

There are already specially built LISP machines in the United States, and these have been used to develop LCSI's implementations of Logo. However, a LISP machine still carries a six-figure price tag, but

"There are now enough good books available to correct the earlier misconceptions of Logo"

that will change, and hardware will be available to make Logo fast and economical, as well as powerful and easy.

In the meantime, who should learn Logo? If you already write Basic programs and feel happy with them, or write long programs in Pascal or Forth, you probably won't think it worth lashing out more than £80 for an intellectual adventure.

Apple should certainly think of reducing the basic price of its Logo, which is more expensive than clearly superior rival products from Atari and Sinclair. On the other hand, if you want to introduce your children or your grandmother to programming, then consider Logo as a humane alternative to Basic or Forth. It doesn't matter that they cannot write a purchase ledger with postings to the general ledger and an integral comparison of actual results to budget. If they want to understand what programming is all about, Logo is the place to begin.

Equally, if you use your Apple II for word processing, book-keeping, and Visicalc, but would like to try your hand at programming, then choose Logo. It won't take you long to get started. Better still it's fun.

Finally, there are those with some experience of programming in conventional languages, who are interested in artificial intelligence research, and would like to get the flavour of list processing without going to the trouble of learning Lisp. Logo could be for you, too.

Over the next two or three months I will be writing about a series of Logo programming projects, each designed to illustrate some particular feature of the language, as it is currently implemented for the Apple II and IIe.

I will be delighted to discuss readers' programs and programming projects, but please don't send me discs or listings which have to be returned without also enclosing a stamped addressed envelope. And please don't write to me explaining how you can do everything I can do in Logo in some other language!

I do not claim that Logo is better than any other language. I leave such claims to computer scientists. It happens to be the language I learned. One difference between this series of articles and almost everything else that has been written about Logo is that I do not have a degree in mathematics or computer science.

The programming projects are explorations for me, as much as for my readers. If you read Mindstorms, you will discover that this is wholly consistent with the spirit of Logo.

● Logo for your Apple is readily obtainable from any Apple dealer. It should cost around £80. Terrapin Logo is slightly different, though several cribs exist to allow you to translate programs from one version to another. It offers more advanced facilities for building in machine code routines, and printing out pictures.

Utilities for the Apple II Logo are available at cost to members of the British Logo User Group (secretary Pam Valley, Shell Mathematics Centre, School of Education, Nottingham University). Membership of BLUG costs £7.50 and entitles you to an introductory information pack, a quarterly newsletter, and discounts on some Logo books.

Seymour Papert's book is still the best introduction to the underlying philosophy of Logo, but Hal Abelson (McGraw Hill, Byte Books) and Peter Ross (Addison Wesley) have both written books which are of more use than Mindstorms to the aspiring Logo programmer with an Apple II or IIe. If you do not have a IIe, you will need a language card to run either of the Apple Logos.

American as a foreign language

AMERICAN educational software often has to be "translated" for use by British schools. The most obvious examples are in spelling – there is not much that can be done about color instead of colour as the American word is an Applesoft Basic command, but other spellings only require cosmetic changes to a program.

A different culture and different teaching methods also make some packages unsuitable – although most are easy to use and offer new ideas and approaches.

A man well qualified to discuss their use is Bill Broderick of the Havering Educational Computer Centre. Since last year he has been importing and distributing programs from the Minnesota Educational Computing Consortium, with topics ranging from agriculture, business and administration to maths, science, social studies, special needs and programming utilities.

The authors of the MECC programs are advised by a professional teacher group but the software has not always received extensive classroom testing before publication.

"The strength of the material lies in its breadth rather than its depth", says Broderick.

Some of the pre-reading material and the arts modules are excellent. Other material is best seen as a stimulus for new ideas.

"We have found that primary school teachers in particular feel inhibited about what they can do with micros. They need something, such as the MEPP software, to spark them off".

The HECC was set up by the London borough of Havering. It has received a grant from the MEP, and Apple UK has lent equipment for producing and copying software.

Library members pay a £35 entrance fee and can buy unlimited numbers of programs for a year. Several programs are distributed on one disc together with documentation for £15 to £18 a disc.

Membership is open to all UK schools, although the library's real role is to provide educational services for the borough's 21 schools. They have 53 Apples and download library programs from a central mini using a modem.

Broderick said buying American software should be approached cautiously – "although just the ideas generated by the modules make it well worth having".

He said many teachers buy teaching packages that look good in the brochures, whereas the main criteria should be whether the package worked well in the classroom.

It wasn't easy for teachers to find out whether software was good or bad. "They seem somewhat isolated and need somewhere or someone to refer to for advice".

He suggested that local education authority advisers be encouraged to build a library of material which teachers could examine before committing themselves to a purchase.

"It is better for an authority to make a mistake once in buying a package than to have that mistake repeated by individual schools", he said.

July 1981
MicroModeller: crystal ball of the 80s? – Surround game (listing) – Bach and the Byte (review of Mountain Hardware's music system) – Apple programs that help the handicapped – Computers in primary schools – Why psychologists plump for the Apple – Use of Apple's unique EXEC files – Format 80 word processor review – The man behind Apple's UK success story – Analysis of CIS Cobol and its flexible file handling facilities. PLUS two pages of Compucopia and 11 Appletips.

August 1981
Networking systems (Constellation, Cluster One, Omninet) – Date validation routine – The Limits of My World (mathematical languages) – Textmaster WP review – Getting started with machine code – Running a preparatory school on an Apple – Software swop shop – Synthesiser as teaching aid – Integer to Applesoft Basic conversion – Apple machine language review – Apple user profile: Hill Samuel – The Market for Micro-Modeller. PLUS two pages of Compucopia and five Appletips.

September 1981
Consumers' guide to Apple music. Part I – Games review (Starmines, Creature Venture, Hi-res Soccer) – Ski-run game (listing) – Speed restrictions with variables – Non-linear curve fitting – Machine code techniques, Part II (text insertion) – Crash course in Basic. Part I – Dot matrix printer review – Apples in networks (modems, Prestel) – CAL explosion coming – Computer games for physically handicapped – Apple user profile: SEGAS. PLUS three pages of Compucopia and five Appletips.

Catch up on the articles you missed by sending for earlier issues. And when your collection is complete, keep it in one of our attractive binders. You can order by mailing the coupon on the right – or by phoning 061-456 4157 and quoting your credit card number.

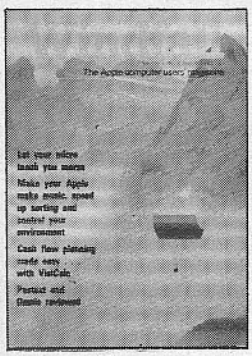
December 1981
Regain Step/Trace in Autostart Apples – Games listings (Apple Casino, Avoid, Calendar) – Games review (German Whist, Wizardry, Galactic Attack, Pool 1.5,) – Sinta Shape Manager review – Machine code techniques, Part IV (sorting arrays) – A/D converter review – Colour systems – Financial Controller review – Wordstar review – Crash course in Basic, Part IV – Debugging the Fortran Compiler – Care of discs – Electronic atlas – Pascal explored. PLUS four pages of Compucopia and seven Appletips.

January 1982
Apple scoop on Tomorrow's World – 1982: The Year of the Apple? – Games review (Wizardry) – Simultaneous equations without tears – Boosting machine code technique – Program Writer/Reporter review – Crash course in Basic, Part V – Machine code techniques, Part V (flagged bubble sorts) – Apple graphics, Part I (Apple's memory map) – Orbit accounting system review – Cost effective terminal computer – Moving hi-res graphics. PLUS four pages of Compucopia and seven Appletips.

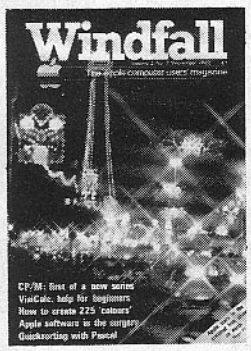
February 1982
Games review (Olympic Decathlon, Dragons Eye) – CP/M: passport to exciting new world – Pascal file conversion program – Machine code techniques, Part VI (EVALuate a new function) – Crash course in Basic, Part VI – Elements of the Apple, Part I – Apple Graphics, Part II (high resolution graph drawing) – Making programs more user friendly – Getting round the memory map muddle – Apple user profile: Sea Fish Authority. PLUS three pages of Compucopia and seven Appletips.

March 1982
Games review (Crush, Crumble and Chomp) – Apple Medical Forum – Data Factory review – Apple Graphics, Part III (displaying histograms) – Printing an annotated DOS disc directory – Crash course in Basic, Part 7 – Start training for the Apple Olympics – Elements of the Apple, Part II – Payroll package for the Apple III – Six educational programs reviewed – DOS 3.3 to 3.2 software switch – Workshop/Wordstar tuition course reviewed. PLUS three pages of Compucopia and four Appletips.

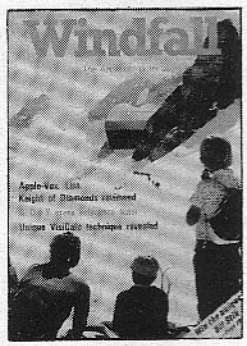
April 1982
Apple speeds the news Games review (Cast Wolfenstein, Threshold, P sident Elect) – DOS Too problems – Linking Apples IBM – Home-grown boa boom – Micro-Finesse revie Basketball match analysis – E ments of the Apple, Part II FMS accounting system revi – DOS disc directory, Part I Apple graphics, Part IV (animation graphics) – Apple Education Forum – A structu approach to teaching. PL four pages of Compucopia a five Appletips.



August 1982
Games review (Bandits, Suicide, Swashbuckler, Fly Wars) – Instruction file editor – Teach yourself Morse, Part I – VisiCalc section – Pastext II review – Asynchronous data transfer, Part II – Omnis review – A melody from your micro – Summary of 10 utilities – Make your own user port, Part II – Mah Jong – Number sorting – Elements of the Apple, Part V – Guidelines for buying a school Apple – Educational programs reviewed – PLUS four pages of Compucopia and two Appletips.

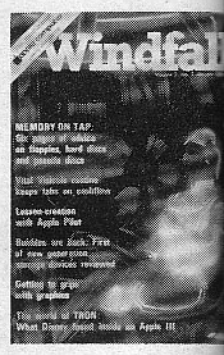


September 1982
Use of CP/M COPY and PIP programs – Games review (Odyssey, Choplifter) – DOS aid to VisiCalc – The VisiCalc phenomenon – Wordscore game (listing) – Tasc compiler review – Med-res graphics, Part I – Snapshot review – Learning Morse, Part II – Button for multiple choice testing – Asynchronous data transfer, Part III – Bag of Tricks review – G-WHIZ review – Medic review – Sorting with Pascal – Memory test program (listing). PLUS four pages of Compucopia and six Appletips.

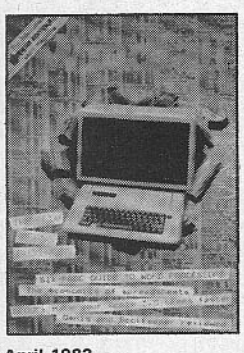


October 1982
Games reviews Knight of Diamonds (the second wizardry scenario) and Pig Pen – Think Tank (with listings) – Med-res graphics, Part II (filling in shapes) – Lisa assembler language review – Magic of VisiCalc – VisiCalc Business Forecasting Model review – Cross reference listing program – Apple-vox speech synthesiser review – Morse Code, Part III – Computerised flash card for schools – French Verb program review. PLUS four pages of Compucopia and seven Appletips.

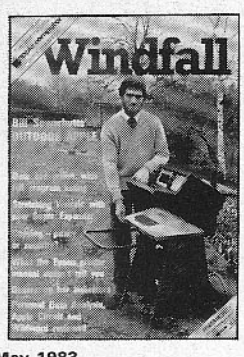
November 1982
A beginner's guide to PEEKs and POKEs, Part I – Games review (Galactic Wars, Night Mission Pinball, Raster Blaster, David's Midnight Magic and three Quick Spins) – Think Tank (with listings) – Three 80 column cards evaluated – Visicalc: Brush up your algebra – Bit Stik graphic system reviewed – Pitfalls in producing educational software – Treasure Islands educational game reviewed – Med-res graphics, Part III (Ampersand routine). PLUS four pages of Compucopia and six Appletips.



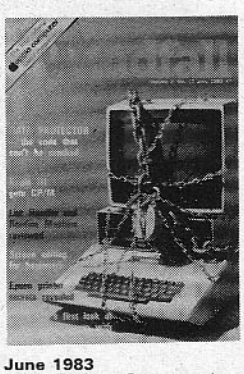
January 1983
Think Tank – Book revie (Apple Graphics and Arca Game Design) – Games revie (Wizard and Princes Transylvania) – Six-page gu to memory storage (guide disc drives, new bubb memory, 128k RAM cards, c back-up, mini-Winches drives, new Apple drives) – W Disney's TRON – Graphma review – Installing Wordsta Business cash flow with Visic – Pilot review – Interact editor-assembler, Part II. PL four pages of Compucopia a eight Appletips.



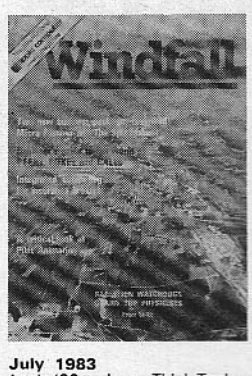
April 1983
Games reviews (Type Attack, Microwave, Tubeway) – Word Processing (Supertext, Executive Secretary, Wordstar, Word Handler) – economics of using electronic worksheets – Fishing (game listing) – Apples in the pet foods and film slides industries – Anatomy of the IIe – Beginner's programming – Reviews (Omnis, Strobe 100 Plotter, Hilderbay Bookkeeper, Turnkey CP/M) – Programming for the classroom – Fickle Finger Proofing Part II. PLUS four pages of Compucopia and six Appletips.



May 1983
Think Tank (Visicalc Magic, Appledarts sound, hi-res routines) – Games reviews (Spy's Demise, Teleport, Beer Run, Prism, Bug Attack) – Moans about manuals – To copy or not to copy – The outdoor Apple – Reviews (Wildword, Apple Circuit, Personal Data Analysis) – Date conversion – Understand the Epson Part I – Visicalc Review of Vergecourt 128k RAMcard and Cdex Visicalc training course – Graphics (generating bar indicators with listing) – Standing Wave Plotter. PLUS Five pages of Compucopia and seven Apple tips.



June 1983
Think Tank – Games reviews (Pie Man, Asteroid Field, Star Thief, Cyclotron, Star Blaster, Warp Destroyer) – Security with Data Encryption – Product reviews (Routine Machine, List Handler, Apple III CP/M Softcard, Savvy, Apple Project Manager and Micronet) – Apple '83 preview – Screen editing for beginners – Understanding the Epson Part II – Book review (Create Word Puzzles with Your Micro) – More Apple Pilot facilities. PLUS five pages of Compucopia and eight Apple tips.



July 1983
Apple '83 review – Think Tank – Games reviews (Zork I,II and III, Hitch-hiker's Guide to the Galaxy, Wavy Navy, Shuffleboard) – Using a printer with DOS – Reviews (Micro Planner and The Spreadsheet) – Visicalc potpourri – Beginners' PEEKs, POKEs and CALLs – Creating a turnkey system – Atomic research Apples – File organisation methods – Insurance broking with an Apple – Pilot Animation – Tip for using both sides of a disc. PLUS five pages of Compucopia and seven Appletips.

August 1983
Think Tank – Reviews (T Accelerator Board – tripling speed of an Apple II; Mic planner Part II; The Ramview and Vision 80 80 column ca for the IIe; SuperPilot – does set a CAL standard?) – Gam reviews (Kabul Spy, Sup Taxman 2, Succession, Ja breaker, Spectre) – III or IIe? Apple III's place in the marke Use indices for What analysis with Visicalc – Ba editing for beginners – Pasc Disc Directory – PLUS f pages of Compucopia and Appletips.

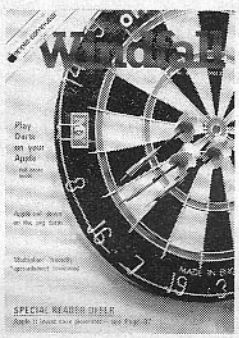
October 1981
Micro Planner review – Games review (Computer Bismark, Battle of Waterlook, Raster Blaster) – Letter square puzzle – Machine code techniques, Part III (dumping screens to printers) – Bulletin boards and personal computer database systems – Teletype terminal program – Crash course in Basic, Part II – Consumer's guide to Apple Music, Part II – Apple user profile: SEGAS, Part II – Apples in South African schools – Programs for primary schools. PLUS two pages of Compucopia and four Appletips.

November 1981
First review of the new Apple III – Games review (Temple of Apshai, Hellfire Warrior, Apple Panic) – Hayden Compiler review – BCPL, a fast language for the Apple – Psychological assessment by the Apple – Beneath Apple DOS book review – New software from the USA – Crash course in Basic, Part III – The role of speech synthesisers in schools – Historical review of computer literacy – Apple user profile: clothing manufacturing. PLUS three pages of Compucopia and six Appletips.

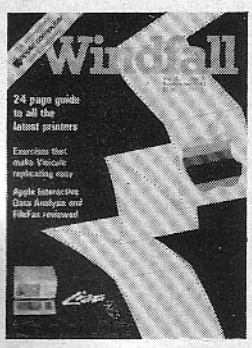
May 1982
A case for Applebus as a new international standard – Games review – Flight Simulator – Hi-res Planet Plotting – Microspeed review – Mathemagic review – Update on Printers (special 16-page printer section) – The Stationery Revolution – Understanding Microcomputers (Part IV) – Simulations Enhance Classroom Work – Computers in Business Education Studies – Speedy Way to Handle Histograms. PLUS four pages of Compucopia and four Appletips.

June 1982
New ways of linking Apples to the outside world – Introduction to Forth, Part I – Games review (The Prisoner, Pinball) – Apples in Medicine – Tasc Compiler review – Micros in process control – Building pictures with machine code – High-speed Apple links to mainframes – Wildport cards review – The Last One and CORP program generators reviewed – Book review (Apple II User's Guide) – Teacher's Toolkit and suite of primary school programs reviewed. PLUS four pages of Compucopia and six Appletips.

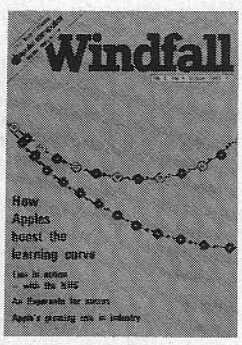
February 1983
Think Tank – Interactive editor-assembler, Part III – Development of Scrabble on the Apple – Visicalc's storage command DIF – Games reviews (Escape from Rungistan, County Fair, Snake Byte, Snack Attack) – Software reviews (Structured Basic, GraForth, Visischeduler and Lisa and the IIe – Pascal Pointers – Network analysis – Handling interrupts – Makeweight grading system – Date-stamping DOS – Educational game (listing) – Formatted Applesoft. PLUS four pages of Compucopia and seven Appletips.



March 1983
Darts game listing – Think Tank – Beginner's look at System Master – Games reviews (Blade of Blackpoole), Banner Magic, Free Fall, Computer Scrabble) – Lower case displays in Basic – Buying a financial spreadsheet – Reviews of Multiplan; Applewriter III; Geometry and Measurement; Drill and Practice; CLIP – News about Lisa and the IIe – Applesoft error handling – Interactive editor-assembler, Part IV – Apple on a pig farm – Fickle Finger proofing, Part I. PLUS four pages of Compucopia and four Appletips.



September 1983
Games reviews (Evolution, Wayout, Aztec, Crisis Mountain) – First impressions of Lisa – Think Tank – Reviews (Apple Interactive Data Analysis, FileFax, Storyboard) – Replicating with Visicalc – Printers Daisywheel v. dot matrix, maintenance contracts, stationery, Pipeline printer buffer and Fingerprint reviewed, new products, printer jargon, A-Z guide to printers, plotters and intelligent interfaces) – Apples and youth training – PLUS three and a half pages of Compucopia and 11 Appletips.



October 1983
Games reviews (Ultima II, Pot O'Gold Plus, Sherwood Forest, Juggler) – Think Tank – In-Circuit Emulation Part One – Lisa (emergency planning with the N.W. Health Authority, developing Busifile) – reviews (Basicode 2, Metacraft's Forth) – Graphics (Digisolve Vector Graphics board and Apple Business Graphics) – Visicalc v. Beebcalc – Training (DIY course selection, what is training, computer-based training) – Package Deal game listing – improving life for the disabled. PLUS Compucopia and Appletips.

ORDER FORM

All prices include postage

SUBSCRIPTIONS — Please enter number required in box — £

UK £12		.....
EIRE £13 (IR £16)		.....
EUROPE £18		.....
OVERSEAS – Surface mail £18		.....
– Air mail £30		.....

BACK ISSUES

☐ UK £1.25
Rest of world
☐ – Surface £1.50
☐ – Air mail £2.50

1981		1982			
JULY	☐	JAN	☐	JAN	☐
AUG	☐	FEB	☐	FEB	☐
SEPT	☐	MAR	☐	MAR	☐
OCT	☐	APRIL	☐	APRIL	☐
NOV	☐	MAY	☐	MAY	☐
DEC	☐	JUNE	☐	JUNE	☐
		JULY	N/A	JULY	☐
		AUG	☐	AUG	☐
		SEPT	☐	SEPT	☐
		OCT	☐	OCT	☐
		NOV	☐		
		DEC	N/A		

TOTAL .....

T-SHIRTS
White - £3.29
(UK & Overseas)

Small – 34"-36"	☐
Medium – 36"-38"	☐
Large – 38"-40"	☐
Extra Large – 40"-42"	☐

TOTAL .....

SWEAT SHIRTS
White - £6.29
Red/Navy - £7.29
(UK & Overseas)

	Red	Blue	White
Age 6-8 28"	N/A	N/A	N/A
Age 10-12 30"-32"	N/A	N/A	
Small 34"-36"	☐	☐	☐
Medium 36"-38"	☐	☐	☐
Large 38"-40"	☐	☐	☐
Extra Large 40"-42"	☐	☐	☐

TOTAL .....

NECKLACES
£4.99
(UK & Overseas) ☐

TOTAL .....

POSTERS
£1.50
(UK & Overseas) ☐

TOTAL .....

TIES
£4.99
(UK & Overseas)

Navy	☐
Brown	☐
Wine	☐

TOTAL .....

BINDERS
UK – £3.95
Overseas – £5.00 ☐

TOTAL .....

Payment: please indicate method (✓) **TOTAL** .....

☐ Access/Mastercharge/Eurocard
☐ Barclaycard/Visa
☐ American Express

Card No. _____

Expiry Date _____

☐ Cheque/PO made payable to Windfall

Name _____

Address _____

_____

Signed _____

Send to: Windfall, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY.
(No stamp needed if posted in UK)

Or you can order by phone quoting credit card number and expiry date.
061-456 4157
9am - 5pm