

Australian Personal Computer

REGISTERED FOR POSTING AS A PUBLICATION - CATEGORY B.
REGISTRATION NO. VBP 3691. ISSN-4415 NZ \$3.00 DECEMBER 1982

\$2.50*

NEW!
EPSON
PORTABLE
BENCHTESTED

AUSTRALIA'S TOP SELLING COMPUTER MAGAZINE



ET ON A CHIP
We review Atari's Extra Terrestrial game

REGULARS

3 PRINTOUT

What's going on in the micro world – and what isn't.

40 CALCULATOR CORNER

A complex arithmetic program for the Sharp PC1211.

51 COMMUNICATIONS

Your chance to moan, whinge, complain – congratulate even!

74 SUBSCRIPTIONS

This month's reason to buy a subscription to *APC*.

82 NEWCOMERS START HERE

APC's intro to computing.

99 DIRECT ACCESS

Our new micro "database" incorporating our existing information columns, DIARY DATA, USERS GROUP INDEX and NETWORK NOTES and incorporating our microcomputer trading-post, MICRO EXCHANGE and our buyer's guide, MICRO MARKET.

101 LAZING AROUND

Another teaser to throw your mental processes into total confusion.

101 BLUDNERS

Every magazine makes mistakes – only *APC* is stupid enough to admit them!

109 PROGRAMS

A mixed bag of readers' software.

120 CHIP CHAT

More malicious jibes, snide asides and spurious innuendos.

FEATURES AND SERIES

18 DATABASE COMPARISON

Kathy Lang compares notes on micro databases.

25 AN INTELLIGENCE TEST FOR COMPUTERS

Bev Mason introduces an interesting calculus program.

57 CLOCK IT TO ME

The conclusion of Bruce Marriott's Apple II clock card design.

77 APC SUBSET

Another collection of useful Assembler routines.

84 ADA – A BRIEF ENCOUNTER

Mike Parr discusses the Ada programming language.

91 WRITING THE SMALL PRINT

Andrew May demonstrates tiny printing on a Centronics 739.

BENCHTESTS

& REVIEWS

22 EPSON HX-20

Dick Pountain peeks inside Epson's hand-held computer.

43 E.T. THE GAME

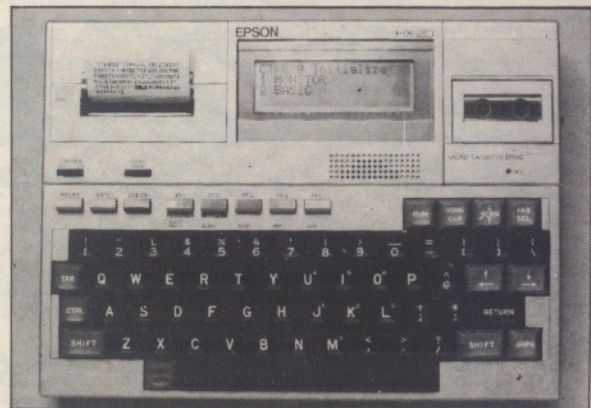
Atari's newest addition to its games repertoire.

49 80 COLUMN CARDS COMPARED

Ian Davies checks-out two popular boards for the Apple II.

95 CARDBOX DATABASE BENCHTEST

Kathy Lang Benchtests a unique data storage system.



22

Editor: Sean Howard Technical Editor: Ian Davies Advertising: Gerard Kohne Printed by: Lewis Printing Produced under licence from: Felden Productions
(03) 544 8855

Published by Howard Productions,
500 Clayton Road, Clayton, Vic 3168. Telephone (03) 544 8855.

Material contained within Australian Personal Computer is protected by the Commonwealth Copyright Act 1968. No material may be reproduced in part or whole without written consent from the copyright holders.



APC reports on the latest news from the world micro scene.

The average store



What constitutes your average micro store in Australia today? Certainly three years ago it wasn't terribly impressive, usually had only one or at the most two makes of microcomputer and catered for a dedicated buyer. Since then there has been a trend to flashier stores each carrying a range of micros which has completely turned the tables on the distributor/retailer relationship. Today the retail

store has no "political" difficulty putting one micro next to another on a shop bench as evidenced by Robs Computer Centre which carries the VIC-20, Atari 400 and 800, Hitachi Peach and Success, Kaypro, Columbia and ICL micros. Robs Computer Store even boasts one of Australia's biggest "computers" on its roof.

New database

Infostar is a new data base management system apparently designed for "non-programmers". It actually consists of two programs, Datastar 1.4 and Reportstar 1.0. Datastar 1.4 has additional features over version 1.101 including increased file limit to 8 Mb, new user-friendly training guide, the ability to name individual fields in addition to numbering them, more memory available for data files by having the ability to define any field as an intermediate field, and not surprisingly the ability to work with Reportstar. It sells for \$610 and more

details can be obtained from Imagineering on (02) 358 3011.

More than games

VIC Education has been quietly developing software for Commodore's VIC-20 over the last half year. It's aimed at young school children and provides a variety of "lessons" on such subjects as maths and spelling. Dennis Argall, MD of VIC Education, says that the software is extensively "tested" by children in the appropriate age groups to assure that bugs and unsuitable material is com-



XMAS FAMILY SYSTEMS

Here is your chance to buy your family a complete computer system for Xmas at a special package price.



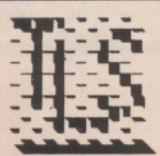
- Apple II Euro Plus.
- Disk Drive.
- 4 Manuals.
- Green Phosphor Display.
- Pair of Game Paddles.
- 4 Software Packages.

\$2950
INC. TAX



- 400 Computer.
- Basic Language and Manual.
- Cassette Program Recorder.
- Pair Joysticks.
- 3 Software Packages.

\$795
INC. TAX



The Logic Shop
Computer Systems.

Ring us now for personal service

- PRAHRAN (Vic.) — Ralph Status (03) 51 1950
- K. MART BURWOOD (Vic.) — Robert LeTet (03) 233 7317
- CHIPPENDALE (N.S.W.) — Ted Keating (02) 699 4919
- HOBBART (Tas.) — James Powell-Davies (002) 31 0818



DIGICARD

FOR THOSE WHO NEED THE BEST

INNOVATIVE
HARDWARE
FOR APPLE
COMPUTER

LATEST RELEASE
FULLY APPLE COMPATIBLE
DISK DRIVE
DEC '82

80 COLUMN CARD

WHEN USING A COMPUTER ALL DAY
THE EIGHTY COLUMN CARD YOU
CHOOSE MUST BE THE BEST

Consider these FACTS before you buy.

THE DIGICARD 80 IS FASTER

**** 1 **** The Central Processing Unit in the APPLE is capable of executing around one million instructions per second. The effective processing speed of the APPLE means it is capable of printing characters at a very high rate. The result is that it is very fast to, say, do a program listing or to print out information. This performance is seriously degraded by other eighty column cards, typically they take over twice the time to list a program, as a standard APPLE. The DIGICARD 80 is typically over 70% FASTER than other popular 80 column cards.

BALANCED VIDEO

**** 2 **** The DIGICARD 80 has a unique feature which allows the video output to be balanced. This means that you can adjust the video to match a wide range of monitors, but more importantly this feature reduces operator eye strain significantly. A very soothing fact for someone who thinks of a computer as more than a toy.

SINGLE KEY APPLESOFT

**** 3 **** The DIGICARD 80 is the ONLY eighty column terminal that creates single key commands in APPLESOFT Basic. Just type ESC and then one other letter to issue one of eighteen commands. eg Suppose you want to CATALOG a disk, all you need to do is type ESC followed by the SPACE bar and the word CATALOG will appear on your screen. All that is needed now is a carriage return and the job is done. Single key commands are great time savers and help speed up program development time.

NO NEED TO MODIFY THE APPLE

**** 4 **** Human engineering has not been forgotten either. The DIGICARD 80 has an on-screen indication of shift lock status, this removes the need for hardware modifications to the APPLE. An audible shift lock indicator is standard on the DIGICARD 80, a very useful feature especially for the professional typist. Then to make typing still easier we added an inverting cursor so that when you move the cursor around the screen it never hides any characters. Then we made the cursor flash on and off slowly so that even on a screen full of inverse characters you can't lose the cursor.

RESPONDS TO GRAPHICS COMMANDS IN ANY LANGUAGE

**** 5 **** The DIGICARD 80 is the only terminal that will respond to graphics commands in ANY language. This feature means you don't have to tediously alter and recompile existing programs.

DIGICARD DISK DRIVE MUCH MORE THAN STANDARD

FULLY
APPLE
COMPATIBLE

FOUR REASONS
WHY DIGICARD DISK
DRIVE IS SUPERIOR

NO LOSS VIDEO SWITCHING

**** 6 **** The DIGICARD 80 has a software controlled video switching system. This feature allows either 40 or 80 columns to be displayed on the terminal. The switching system is a NO LOSS system which means that the normal 40 column display is not degraded.

FULLY COMPREHENSIVE COMMUNICATIONS ON THE CARD

**** 7 **** The DIGICARD 80 has a fully comprehensive COMMUNICATIONS firmware package that has greater versatility than similar systems. A communications package allows you to transfer data from your computer to virtually any other computer be it large or small. This is a very useful feature if you have access to a large mainframe computer or simply another APPLE computer. Data can be transmitted and received up to 48,000 baud SIMULTANEOUSLY without loss of any characters. Even if your terminal gets a message to beep its bell the DIGICARD 80 will still not miss any characters. The communications package enables you to remotely operate another APPLE with a DIGICARD 80 installed. You can even RUN programs on the remote computer. Who else can do all that and still be able to support both the C.C.S. and the Super Serial card with one firmware package.

LOW POWER CONSUMPTION

**** 8 **** You might think with all these features that the DIGICARD 80 would be a power hungry brute, but it only consumes a mere 2.3 watts, quite considerably less than other 80 column terminals with none of the above features. The low power consumption of the DIGICARD 80 means your APPLE will run cool all day long which is a comforting thought.

The DIGICARD DISK DRIVE is guaranteed to run with all software that a standard APPLE DISK DRIVE can operate with.

PRECISION HEAD POSITIONING

The DIGICARD DISK DRIVE uses a precision taut band that locates the head with greater accuracy. When the APPLE is reading data from a diskette it tries up to 48 times to get the correct data off the disk. If the head is positioned more accurately in the first place then the number of read attempts can be greatly reduced. This all means that your program and data can be loaded off the disk faster.

FAST TRACK TO TRACK ACCESS

As well as being a more accurate head positioning system the DIGICARD DISK DRIVE mechanism is capable of changing from one track to another FOUR TIMES FASTER than the standard APPLE disk drive. This further adds to the speed that data can be transferred to or from disk.

TRACK ZERO DETECTION

When a standard disk drive is booted it has to find track zero. The normal way creates a horrible clackety clack noise from the drive mechanism. The DIGICARD DISK DRIVE has a track zero detector that eliminates the unnecessary noise and once again speeds up disk access time.

MORE DATA STORAGE

The standard disk drive stores data on a total of 35 tracks around the disk. The DIGICARD DISK DRIVE has provision for storing data on 40 tracks thus creating storage for up to 20,000 bytes, this is around 15% more storage than a standard drive.

SUMMARY

When using a DIGICARD DISK DRIVE you are operating a FASTER, more PRECISE and HIGHER STORAGE disk drive than the standard APPLE disk drive while maintaining ABSOLUTE COMPATIBILITY.

INSTALLATION IN LESS THAN 60 SECONDS

**** 9 **** INSTALLATION of the DIGICARD 80 could not be easier. The card can be installed and ready to go in less than 60 seconds.

EASY SERVICE

**** 10 **** All integrated circuits are socketed for easy service.

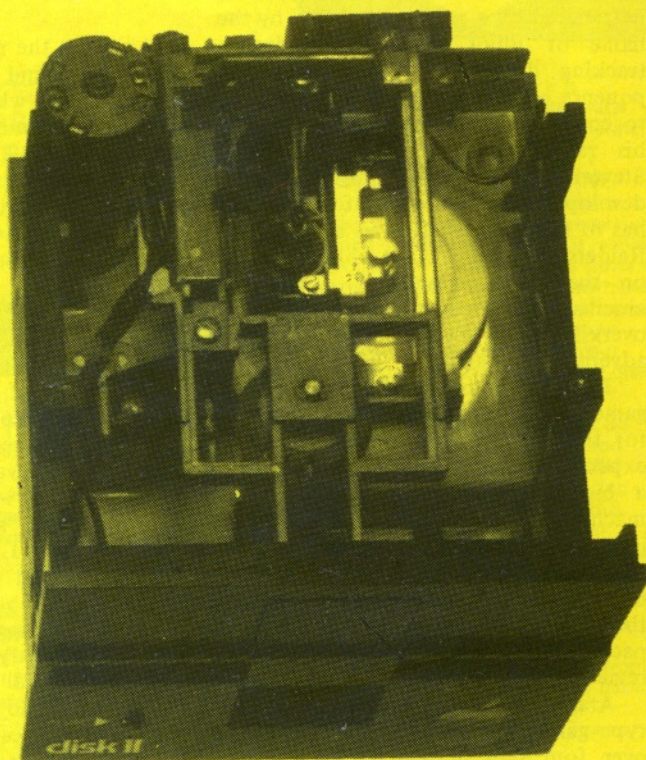
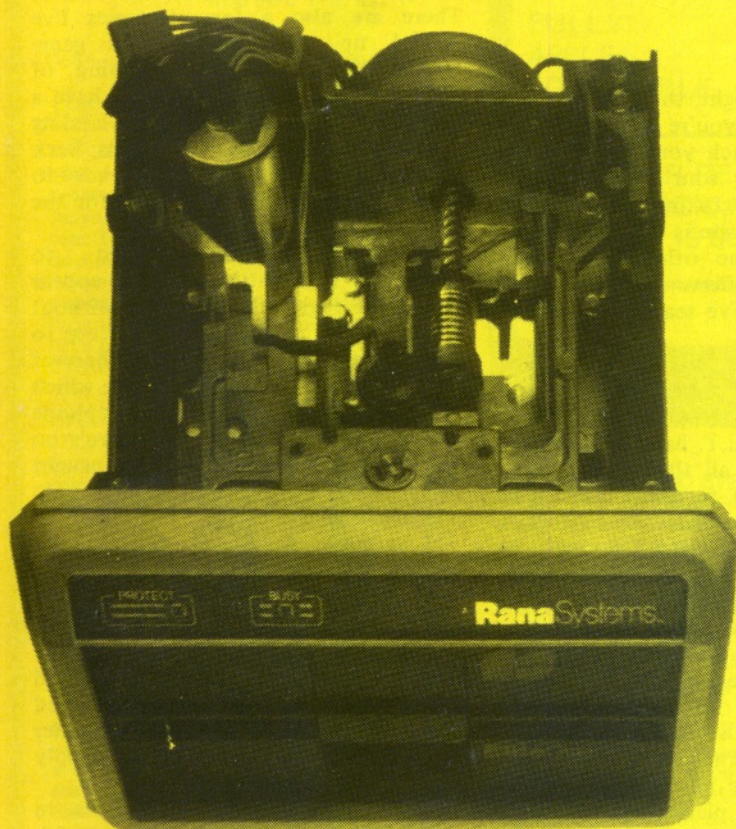
12 MONTH GUARANTEE

**** 11 **** All DIGICARD products are designed and manufactured in Australia by MacLagan Wright & Associates and are backed by a 12 month guarantee. If any product is found to be faulty within the warranty period it will be replaced free of charge.

DIGICARD PRODUCTS are available from your local APPLE dealer. If you require more information call your APPLE dealer or

MACLAGAN WRIGHT & ASSOCIATES PTY LTD on (03) 436 1351.

When you say your disk drive has more juice than Apple's, be prepared to cut one open.



Your Apple computer can grow from ordinary to awesome in a matter of minutes.

The Elite disk drive Series by Rana Systems gives you that kind of magic. Quickly, easily and cost-effectively.

This superb family of sophisticated floppy drives was designed and perfected by a team of industry-respected engineers expressly for Apple computer owners who have long been demanding more disk drive for their money.

Now they have a drive that makes an Apple perform the way it should. For example:

Elite One, the most economical Rana drive, delivers 15% more storage capacity

than Apple's drive. The top-of-the-line Elite Three will provide an astonishing *four-times* more storage, approaching hard disk performance. It's done through exclusive high-density single and double-sided disks and heads.

Here's a tantalising taste of just how formidable your Apple can become.

Next to enormously increased disk space, Apple buyers are most excited about Rana's fingertip Write/Protect feature. A pushbutton on the LED-lit panel gives you a new measure of failsafe control.

While Elite drives are all plug-compatible with Apple's controller, Rana's vastly superior controller card is a *must* for those

with growing data processing demands. With it, you control up to *four* floppy drives using only one slot . . . while still utilizing your Apple computer to achieve dramatic new heights of system flexibility and processing power.

A specially programmed Rana utility disk assures you of far-reaching compatibility, and extra dimensions of business, leisure and learning applications with your Rana-enhanced Apple computer.

There are many, many more operating refinements: Uncanny head-positioning accuracy, 300% faster track-to-track speed, better operating economy, and gentler clamping action through an all-new diskette centering cone, to name just a few.

A disk drive package like this doesn't grow on trees, so write or call today for more information about the Elite Series.

ELITE 1	\$640	(163kb) with Controller	\$790
ELITE 2	\$830	(326kb) with Controller	\$990
ELITE 3	\$1090	(652kb) with Controller	\$1250
CONTROLLER ONLY	\$160	(All prices include Sales Tax)	

CONTACT:

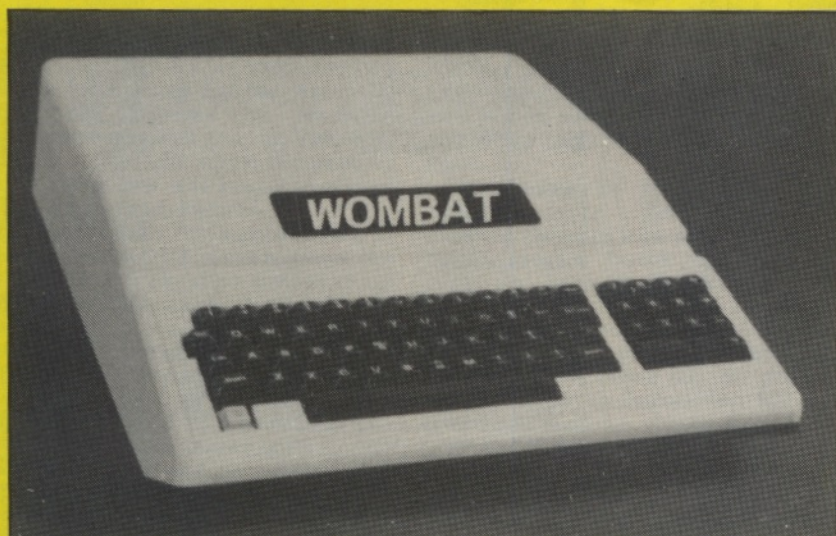
Computer Edge Pty. Ltd.

364 FERRARS STREET, ALBERT PARK, VIC 3206. Telephone: (03) 690 1477



★★ WOMBAT ★★

The world of computing is now available to everyone at very reasonable cost. For parents who wish their children (and themselves) to learn programming, to utilise educational programs, or just to make use of some of the hundreds of programs available the WOMBAT is now released on the Australian market.



* BONUS —

For each WOMBAT sold before January 25th, a choice of any ten programs from the MECC Educational Library will be given free — valued at approx. \$288!

Standard features:

- * built-in floating-point BASIC;
- * built-in disassembler;
- * a ROM-based system control program;
- * color graphics and sound capabilities;
- * sockets for up to 48K bytes of user memory (RAM); 16K with the plug-in Language Card;
- * cassette interface;
- * game input/output connector, and two hand controls for games and other human-input applications;
- * typewriter-style ASCII keyboard;
- * high-efficiency switching power supply;
- * eight accessory expansion slots;
- * autostart monitor;
- * upper/lower case function.

Inputs and Outputs:

- * cassette interface (1500bps);
- * game I/O: four analog to digital inputs, three TTL inputs and four TTL outputs.
- * typewriter-style ASCII keyboard;
- * 7 peripheral board connectors fully buffered, with interrupt and DMA priority structure;

NOTE: THE WOMBAT SOFTWARE IS COMPATIBLE WITH ALL EXISTING PROGRAMS FOR THE APPLE II

CONTACT:

Computer Edge Pty. Ltd.

364 FERRARS STREET, ALBERT PARK, VIC 3206. Telephone: (03) 690 1477



JUST \$940

(plus S.T. — \$120)

IF APPLICABLE



80 Column Cards Compared

by Ian Davies

This month, CHECKOUT examines two eighty column cards for the Apple II: namely the Digicard by Maclagan Wright and Associates and the Vision-80 card by Zofary. The Apple II computer used in the review was kindly furnished by The Logic Shop, Prahran.

In keeping with the high standard of in-depth CHECKOUT investigations, the first action I took after powering up each card was to count the number of columns Eighty on both - drat!! no controversial journalism on this review!

The most noticeable feature about the two cards is their similarity. They both provide much the same features (on the surface) and appear quite similar in construction. Closer inspection of the boards revealed, however, that one was not a rip-off of the other. Each board is based around the 6845 CRT controller chip - a rather gutsy little device that does most of the work of screen control. Both boards also contain two 2716 2k byte EPROMS, one of which is used as a character generator into the 6845 and the other contains 6502 controlling software. The video memory itself is 2k bytes of RAM, implemented through four 2114 static RAM chips (1k x 4 bits) on the Vision-80, and through a single 6116 chip on the Digicard, which appears to be employing slightly more "state of the art" technology. The exception to this is the fact that the Digicard uses an ELEC-TROL Read Relay (presumably to switch between its video output and the standard Apple video output),

whereas the Vision-80 employs a 4016 chip, which is a quad bi-lateral switch and is not subject to the problem of mechanical degradation over a period of years.

Installation of the cards is quite straightforward, simply involving plugging the card into port three on the Apple mother board. A few flying clips do have to be connected to strategic points inside the Apple, but this presented no problems. Although the installation procedures are quite similar, the Vision-80 is probably slightly easier to install, but not by a significant margin.

Both cards provide a "shift-lock" facility. The Digicard alters the cursor character to indicate whether it is in upper case or lower case mode, whereas the Vision-80 displays the fact through a red LED. The LED should be mounted on the Apple console, which involves drilling a hole through the Apple casings. Both manuals stated that the shift mode could be changed by pressing the SHIFT key on the Apple keyboard. The Digicard (which was in the machine when we received it) seemed to prefer the CONTROL key for this purpose. This proved to be caused by incorrect installation in the Apple used in the review, and was not the fault of the Digicard itself.

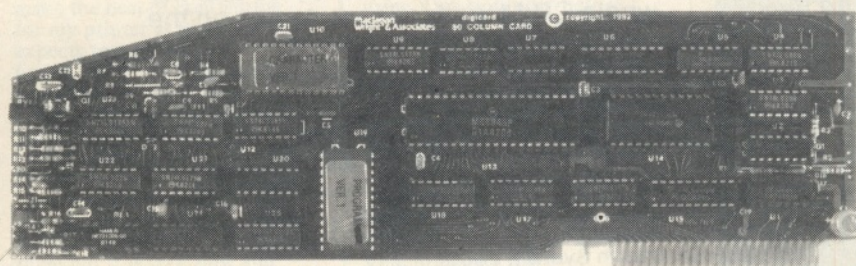
The two cards have extensive screen control features accessed through ESCAPE or CONTROL keys. Included in the features is a rather nice communications package which allows you

to communicate with mainframes in various data formats while still using the Apple disk. As an added bonus, the Digicard also provides 17 single-key keyword entries, for example, pressing ESC 6 produces GOTO.

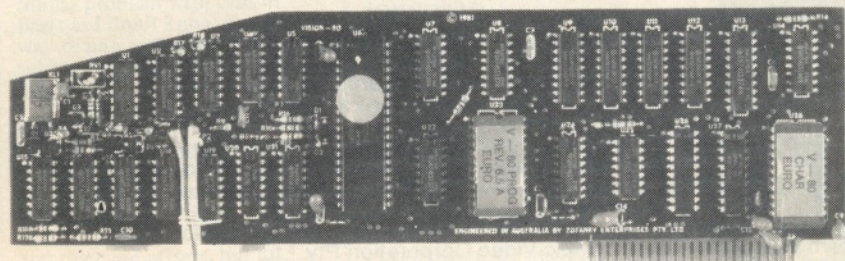
The quality of the output is very good on both boards, although there was some problem with character definition on the Vision-80 (specifically, the horizontal parts of the characters were far too bright) but this appeared to be caused by the fact that the Apple was "tuned" for the Digicard. Both provide excellent high resolution graphics, with the ability to switch between the hi-res screen and the lo-res screen at will. Both boards also allow you to run two monitors - one for hi-res output and one for lo-res. Toggling from one logical screen to the other did produce a slight shudder in the display, but this was common to both.

One area in which the two boards differ is documentation. The manual provided with the Vision-80 seemed significantly superior, including greater detail, and a section at the rear for assembly programmers and troubleshooting. It also contained a circuit diagram.

In summary, the two cards are very much alike, although the Vision-80 probably does have a slight edge due to a couple of extra features and better documentation. Either one is essential for any sort of serious word processing work. The Vision-80 retails for \$337 and the Digicard for \$354.



Digicard



Vision 80

Apple computer authorized dealer

DEMONSTRATIONS
OF THE LATEST
EDUCATIONAL
PROGRAMMES
AT ...



THE EDUCATION SPECIALISTS

250 Parramatta Rd., Annandale, N.S.W.
Phone: (02) 569-5188.

COMMUNICATIONS

scope (the size of chunk it governs in the list).

For a somewhat dramatic illustration, consider next the definition of the function

```
LENGTH:
(DE LENGTH . LIST :
COND : : EQ LIST NIL . 0
: : T . PLUS 1 :
LENGTH . CDR LIST)
```

If you don't think that makes the list structure a lot clearer, did you notice the printer's error in the original? There's an erroneous right parenthesis on the end of the second line.

Now to those lists of lists of lists. The simplest I can think of is a pair of pairs of pairs of atoms. Take your pick - brackets or dots?

```
(SETQ PAIRS '(((A B)(C D))
((E F)(G H))))
(SETQ PAIRS ':A B . C D :
E F . G H)
```

John Kerr

...Lisp or logo

Referring to Mr Kerr's letter suggesting that a Lisp interpreter would run faster if reserved words were to be tokenised and brackets removed, it must be made clear that to the interpreter a program, ie, a list, is represented as a set of linked pointers. The brackets are used to delimit lists only in the input/output routines. In Lisp not only are the reserved words tokenised, all words are! If Mr Kerr desires the elegance of Lisp without so many brackets, then I suggest he considers Logo which is Lisp-based and is friendlier in use. This leads me to my second point.

Mr Parr's Logo compiler written in Basic is an interesting program (though lamentably slow) but contains a bug which could confuse a beginner grappling with recursion. If a previously encountered function is again recognised by the compiler, the number of arguments is not fetched and at run time the function is called with wrong values. This occurs in the Branch program in the September 1982 Logo article. It can be fixed by changing line 9730 to read

```
IF SY$=PN$(W) THEN
N=CP(N); RETURN
```

Ian A Stewart

Space defender

It is a shame that so much had to be left out of Names of the Nameless in the October issue. In particular, I wish room could have been found to

justify the dangerous use of a space as a mathematical symbol in GSB's arithmetic, since this can, as it does here, involve implicit axioms. It seems that the axioms should include (though one can't be sure):

```
space space = space
space   } =   }
   }    } =   }
   }    } =   }
   }    } =   }
   }    } = space
```

which are isomorphic to

```
0 or 0 = 0
0 or 1 = 1
1 or 0 = 1
1 or 1 = 1
NOT 0 = 1
NOT 1 = 0
respectively.
```

If this is so then GSB's arithmetic is nothing more than Boolean algebra and can hardly be said to be more fundamental.

Your article tries to allay the suspicion that *Laws of Form* is a crank book, and to encourage potential readers, but I must regretfully say 'Not convinced'.

James Crook

This is one of several letters from mathematicians who disapprove quite strongly of Spencer-Brown's work. His proof of the four-colour theorem is by no means accepted 'in the trade', and is as far as I know not officially published. Not being a professional mathematician, I do not feel threatened by the unorthodoxy of Laws of Form, nor do I feel that APC readers are likely to come to much harm.

It is indeed a shame that so much had to be left out; I obviously failed to make clear that 'space' is not a symbol, but precisely the space in which a cross stands. There is only one initial symbol, the cross, which indicates the marked state. The idea is certainly more fundamental than Boolean algebra, to the point where having to write it in ink on paper almost subverts its understanding. It may well be dangerous, but then so are flying and mountaineering. - Dick Pountain.

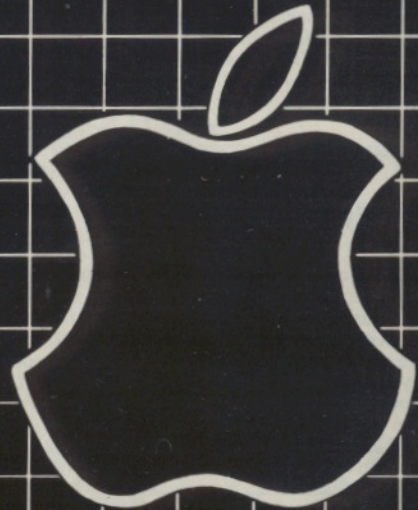
What's in a name?

I am considering selling software for my ZX81 but have a query about copyright. If I see a program working on a computer other than the ZX81, and I decide to write a program based on this idea, but using my own programming ideas, is it a breach of



80 COLUMN CARD

DOES ALL THIS PLUS



The Vision 80 video card is an easy yet sophisticated way to enhance the performance of your Apple II Computer. Just plug it in and immediately your Apple II will display a full 80 column, x 24 line screen.

The Vision 80 is compatible with existing Apple II BASIC software and provides enhanced screen performance with programs written in BASIC, PASCAL, FORTRAN, CP/M (MICROSOFT) and ASSEMBLER.

And the Vision 80 enables your Apple II to be used as a true intelligent terminal to mainframes and communication facilities.

Compatible with most good word processors, including ZARDAX, APPLE WRITER II and WORDSTAR

A superb set of 128 upper and lower case characters in a 9x11 dot matrix, including 3 dot descenders. Shift and shift lock for upper and lower case. Source switches both hardware and software between 40x24 and 80x24 screen.

Plus: Now includes - UTILITIES DISK, BASICS, DIAGNOSTICS, CHARACTER SET EDITOR AND DEMO PROGRAM. Supports PASCAL key press & type ahead buffer, graphics character set & underline set.

Plus: Now available -

VISICALC and APPLEWRITER II (preboot disc) both now available in 80 column full upper and lower case on screen display.

DISTRIBUTED BY:

IMAGINEERING

(02) 358 3011

22 SIR JOHN YOUNG CRES, WOOLLOOMOOLOO 2011

OR

FROM YOUR LOCAL DEALER NOW

CLOCK IT TO ME

Bruce Marriott continues his description of a clock/calendar card for the Apple II with a description of the software drivers.

Last month I presented the hardware needed to interface a clock/calendar card based on the OKI MSM5832 chip to the Apple II, or — in rather less detail — to other systems. This month we'll look at the driving software. Naturally, this is based on the Apple, too, but where appropriate I have included some information on amending the code for other systems.

The 6521 PIA

Since the clock chip is accessed through a 6821 PIA, it is first necessary to know how to control the PIA. What follows is

a resumé on how this is achieved — for a more complete version, see the 6821 data sheet.

The 6821 has two 8-bit ports, A and B; for the purpose of this exercise, they can be considered as identical apart from the unique addresses associated with each. The PIA has four interrupt inputs (two of which may also be used as outputs) which will be discussed later. Each port is controlled by two registers, the data direction register and the peripheral register, which, because of addressing limitations, have the same address (Figure 1), with selection between them being made by the value of bit 2 in a

third register (which has no duplicated address) called the control register. The data direction register for each port governs which bits will act as inputs and which will be outputs; writing a 0 to a bit makes it an input while a 1 makes it an output. The peripheral register allows the setting of levels on outputs and the reading of levels on inputs — a 1 in a bit indicates that it is high (+5 volts) and a 0 low (0 volts).

Figure 2 shows how ports A and B are connected to the MSM5832. It repeats information given in last month's circuit diagram but in a more easily assimilated form. The examples

CONTROL OF INTERRUPT INPUTS CA1 AND CB1

CRA-1 (CRB-1)	CRA-0 (CRB-0)	Interrupt Input CA1 (CB1)	Interrupt Flag CRA-7 (CRB-7)	MPU Interrupt Request IRQA (IROB)
0	0	↓ Active	Set high on ↓ of CA1 (CB1)	Disabled — IRQ remains high
0	1	↓ Active	Set high on ↓ of CA1 (CB1)	Goes low when the interrupt flag bit CRA-7 (CRB-7) goes high
1	0	↑ Active	Set high on ↑ of CA1 (CB1)	Disabled — IRQ remains high
1	1	↑ Active	Set high on ↑ of CA1 (CB1)	Goes low when the interrupt flag bit CRA-7 (CRB-7) goes high

RS1	RS0	Control Register Bit		Location Selected
		CRA-2	CRB-2	
0	0	1	x	Peripheral Register A
0	0	0	x	Data Direction Register A
0	1	x	x	Control Register A
1	0	x	1	Peripheral Register B
1	0	x	0	Data Direction Register B
1	1	x	x	Control Register B

----- CRA

----- CRB

7	6	5	4	3	2	1	0
IRQA1	IRQA2	CA2 Control			DDRA Access	CA1 Control	

7	6	5	4	3	2	1	0
IRQB1	IRQB2	CB2 Control			DDRB Access	CB1 Control	

CONTROL OF CA2 AND CB2 AS INTERRUPT INPUTS
CRA5 (CRB5) is low

CRA-5 (CRB-5)	CRA-4 (CRB-4)	CRA-3 (CRB-3)	Interrupt Input CA2 (CB2)	Interrupt Flag CRA-6 (CRB-6)	MPU Interrupt Request IRQA (IROB)
0	0	0	↓ Active	Set high on ↓ of CA2 (CB2)	Disabled — IRQ remains high
0	0	1	↓ Active	Set high on ↓ of CA2 (CB2)	Goes low when the interrupt flag bit CRA-6 (CRB-6) goes high
0	1	0	↑ Active	Set high on ↑ of CA2 (CB2)	Disabled — IRQ remains high
0	1	1	↑ Active	Set high on ↑ of CA2 (CB2)	Goes low when the interrupt flag bit CRA-6 (CRB-6) goes high

Fig 1. 6821 PIA register addressing

6821 MSM5832		Remarks	
PORT A	PA ₀	A ₀	Address lines to select specific time/date
	PA ₁	A ₁	
	PA ₂	A ₂	Register Always 6821 outputs.
	PA ₃	A ₃	
PORT B	PB ₀	D ₀	Time & date Data
	PB ₁	D ₁	
	PB ₂	D ₂	Bidirectional
	PB ₃	D ₃	
	PB ₄	HOLD	Control lines.
	PB ₅	READ	
	PB ₆	WRITE	
CA ₁	D ₀	1024H2	5832 Interrupt
CB ₂	D ₁	1 Hz	Outputs at
CB ₁	D ₂	1/60 H2	Specified
CA ₂	D ₃	1/3600 H2	rates

Fig 2. 6821 PIA to MSM5832 interconnection

which follow should clarify how the PIA/5832 combination is used.

Clock set and read

The program in Listing 1 allows the clock to be set and read from (Microsoft) Basic and should thus be relatively easy to establish on most popular machines. The program was written in Applesoft Basic and has been annotated. If you are using another machine with the 6821, only a few areas will require amendment, the major one being to the address of the PIA. With the Apple, this address changes depending on which slot is used to house the card; lines 110 to 120 handle this. For most other systems the PIA will be mapped into a specific area of memory, allowing lines 110 and 115 to be deleted and line 120 modified so that variable A0 contains the base address (ie, the address of the first location in the PIA).

Other, more minor, amendments will also be necessary. In line 15 the string variable BELL\$ is set to beep the Apple's speaker; if your machine has no such facility then set BELL\$ to a null value (" "). In line 50 the variable CLR is set to a value which, when CALLED, clears from the current cursor position to the end of the line. The actual CALL is only used once (in line 400) and the two lines should be modified as appropriate. Finally, a few Basic words may require clarification for non-Apples: TEXT declares that an alphanumeric display is required rather than graphics; HOME clears the display and puts the cursor in the top left-hand position; INVERSE declares that all subsequent PRINTing will be black on white; and NORMAL declares that printing will be white on black.

If your design isn't based on a 6821 then, of course, the PEEKs and POKEs to control the interface will also have to be modified; to help you with this, I have REMarked all of them to show what they're doing.

The program has been kept relatively short and simple and will not be dissected. However, it could be substantially improved to make it easier to use. For example, it could allow normal date and time entry (eg, 9/5/82 for date) or automatically calculate the day of the week (see *Some Common Basic Programs*, 3rd edition, by L Poole & M Borchers, pub Osborne/McGraw-Hill, 1979, for a suitable method). Additionally, automatic leap year bit setting, instructions and extensive error-trapping

```

10 AA$ = " CLOCK SET AND READ "
15 BELL$ = CHR$(7): REM PRINTING BELL$ BEEPS SPEAKER
20 DIM AN(12)
50 CLR = - 868: REM CALL CLR TO CLEAR SCREEN FROM CURSOR ONWA
RDS
100 TEXT : HOME : VTAB 1: HTAB (40 - LEN (AA$)) / 2: INVERSE : PRINT
AA$: NORMAL
110 VTAB 3: INPUT "ENTER CLOCK SLOT NUMBER, ";SL
115 IF SL < 1 OR SL > 6 THEN PRINT BELL$: GOTO 110
120 A0 = 49280 + SL * 16
125 A1 = A0 + 1:A2 = A1 + 1:A3 = A2 + 1
150 VTAB 5: INPUT "<S>ET OR <R>EAD ?," ;AN$
155 IF AN$ < > "S" THEN 500
199 REM SET CLOCK
200 POKE A1,0: POKE A0,255: REM CONFIGURE ADDRESS LINES ON A S
IDE AS ALL OUTPUTS
205 POKE A3,0: POKE A2,255: REM CONFIGURE CONTROL & DATA LINES
ON B SIDE AS ALL OUTPUTS
210 POKE A1,4: POKE A3,4: REM GET READY TO SET A & B SIDE LEVEL
S
220 VTAB 7: PRINT "ENTER;": VTAB 9
225 INPUT "UNITS OF MINUTES" ;AN(2)
230 INPUT "TENS OF MINUTES" ;AN(3)
235 INPUT "UNITS OF HOURS" ;AN(4)
240 INPUT "TENS OF HOURS" ;AN(5)
245 INPUT "UNITS OF DAYS" ;AN(7)
250 INPUT "TENS OF DAYS" ;AN(8)
255 INPUT "UNITS OF MONTHS" ;AN(9)
260 INPUT "TENS OF MONTHS" ;AN(10)
265 INPUT "UNITS OF YEARS" ;AN(11)
270 INPUT "TENS OF YEARS" ;AN(12)
275 INPUT "DAY OF WEEK (0 SAT, 6 FRI)" ;AN(6)
280 INPUT "NEXT FEB GOT 29 DAYS? (Y OR N)" ;AN$
285 IF AN$ = "Y" THEN AN(8) = AN(8) + 4: REM ADJUST FOR LEAP YE
AR
290 AN(5) = AN(5) + 8: REM WORK IN 24 HR FORMAT
300 PRINT : INVERSE : PRINT "SWITCH WRITE ENABLE ON (SWITCH NO.
4)" ;BELL$
310 INPUT "PRESS RETURN TO SET TIME" ;AN$
315 NORMAL
320 POKE A2,16: REM TAKE HOLD LINE HIGH & STOP CLOCK
330 FOR I = 0 TO 12
340 POKE A2,AN(I) + 16: REM SETUP DATA LINES
350 POKE A0,I: REM SETUP ADDRESS LINES
360 POKE A2,AN(I) + 80: REM TAKE WRITE LINE HIGH
370 POKE A2,AN(I) + 16: REM TAKE WRITE LINE LOW
380 NEXT I
390 POKE A2,0: REM TAKE HOLD LINE LOW & START CLOCK
400 VTAB 6: CALL CLR: VTAB 8: INVERSE : PRINT "SWITCH WRITE ENAB
LE OFF";BELL$: NORMAL
410 REM AUTOMATICALLY FALL INTO READ CLOCK
499 REM
500 POKE A1,0: POKE A0,255: REM CONFIGURE ADDRESS LINES ON A S
IDE AS ALL OUTPUTS
510 POKE A3,0: POKE A2,240: REM CONFIGURE B SIDE WITH LOWER 4 B
ITS AS INPUTS (DATA) & UPPER 4 BITS AS OUTPUTS (CONTROL)
520 POKE A1,4: POKE A3,4: REM GET READY TO SET A & B SIDE LEVEL
S
530 POKE A2,16: REM TAKE HOLD LINE HIGH TO STOP CLOCK
540 POKE A2,48: REM TAKE READ LINE HIGH
550 FOR I = 0 TO 12
560 POKE A0,I: REM SETUP ADDRESS
570 AN(I) = PEEK (A2) - 48: REM READ & STORE DATA
580 NEXT I
590 POKE A2,16: REM TAKE READ LINE LOW
600 POKE A2,0: REM TAKE HOLD LINE LOW ( & ALLOW CLOCK TO CONTIN
UE)
610 AN(5) = AN(5) - 8: REM TAKE OUT 24 HR BIT
620 IF AN(8) = > 4 THEN AN(8) = AN(8) - 4: REM ALLOW FOR LEAP
YEAR BIT
630 VTAB 10: HTAB 8: PRINT "TIME ";AN(5);AN(4);",";AN(3);AN(2);
",";AN(1);AN(0)
640 VTAB 11: HTAB 8: PRINT "DATE ";AN(8);AN(7);"/";AN(10);AN(9)
;"/";AN(12);AN(11)
650 VTAB 15: INVERSE : PRINT "USE 30 SEC ADJUST SWITCH IF NECESS
ARY": NORMAL
660 IF AN(6) = 6 THEN TD$ = "FRIDAY": GOTO 690
665 IF AN(6) = 5 THEN TD$ = "THURSDAY": GOTO 690
670 IF AN(6) = 4 THEN TD$ = "WEDNESDAY": GOTO 690
675 IF AN(6) = 3 THEN TD$ = "TUESDAY": GOTO 690
680 IF AN(6) = 2 THEN TD$ = "MONDAY": GOTO 690
683 IF AN(6) = 1 THEN TD$ = "SUNDAY": GOTO 690
686 TD$ = "SATURDAY"
690 VTAB 18: PRINT "( BYE THE WAY: TODAY IS ";TD$;" !!)"
700 GOTO 530: REM REPEAT READ & PRINT

```

Listing 1

would also be useful, but space does not allow such a lengthy program to be printed here.

ROM software

As discussed last month, the Apple I/O facility allows each peripheral card a

256-byte driving program. Having an intelligent card makes for much easier application programming when wishing to access time and date. Listing 2 shows a Basic program that gets and prints the time and date (using the yet-to-be-given driving program) and this should be compared with the much longer and more

CLOCK IT TO ME

fiddly clock read part of Listing 1.

For convenience, I decided to use the well-known 2716 (single rail) EPROM for storing the driving programs. Although this device has room for eight driving programs, it is connected in such a way that only four spaces are available; selection of only one space is determined by the settings on two switches. Figure 3 shows how the switches relate to the EPROM memory map and where the driving program which follows (called 'normal format') should be located.

Listing 3 is the annotated assembler listing of the normal format driving program. This does not follow Apple's standard protocol for slot use (as briefly described in the Apple Reference Manual). Normally the Apple takes input, one character at a time, from the input device, stores it in the input buffer (\$20-2FF) and outputs it to the current output device. If the input is the clock driving routine and the output is the Apple screen (as would normally be the case) this would dictate that every time the program wanted the time and date this information would automatically be printed on the screen, which is unduly restrictive and, for most programmers, would be an irritant.

A solution to this is not to use the standard I/O protocol for every character but to fill the input buffer with all characters at the same time and hence fool the Apple into thinking that it has handled each character separately. This works fine but there's still another problem to overcome: whenever an input statement is processed a question mark is sent to the current output device (which could be the printer or a

Address (Hex)	Not Available for driving program Storage (Address line A10 permanently tied to ground).	DIL Switch Setting	
		1 (A8)	2 (A9)
400 3FF	NORMAL FORMAT	off	off
300 2FF	(FREE)	on	off
200 1FF	(FREE)	off	on
100 FF	(FREE)	on	on

Fig 3 Relationship between 2716 memory map and Apple clock card Dil switch settings

```

5 TEXT
10 I$ = CHR$(4)
20 PRINT I$"IN#4"
23 INPUT DA$,TI$
26 PRINT I$"IN#0"
30 PRINT "DATE " ; DA$
40 PRINT "TIME " ; TI$
90 END
    
```

Listing 2

C400-	78	SEI		Stop interrupts.
C401-	A9 E0	LDA	#\$E0	Load a space character.
C403-	A4 24	LDY	\$24	
C405-	91 28	STA	(\$28),Y	Print space at current output position.
C407-	20 10 FC	JSR	\$FC10	Backspace cursor.
C40A-	A9 E0	LDA	#\$E0	
C40C-	A4 24	LDY	\$24	Repeat above 4 lines.
C40E-	91 28	STA	(\$28),Y	
C410-	20 10 FC	JSR	\$FC10	
C413-	08	PHP		
C414-	68	PLA		Save status.
C415-	8D FC 03	STA	\$03FC	
C418-	D8	CLD		
C419-	20 58 FF	JSR	\$FF58	
C41C-	BA	TSX		
C41D-	8D 00 01	LDA	\$0100,X	Find which slot we are in.
C420-	8D FB 07	STA	\$07FB	(See Page 81 in Apple Reference Manual.)
C423-	0A	ASL		(Note that there is no need to save)
C424-	0A	ASL		(the 6502 Registers, via a call to FF4A).
C425-	0A	ASL		
C426-	0A	ASL		
C427-	AA	TAX		
C428-	A9 00	LDA	#\$00	
C42A-	9D 81 C0	STA	\$C081,X	
C42D-	A9 FF	LDA	#\$FF	
C42F-	9D 80 C0	STA	\$C080,X	Configure PIA as follows;
C432-	A9 04	LDA	#\$04	A side (5832 addresses) — Outputs
C434-	9D 81 C0	STA	\$C081,X	B side (Bits 4-7; Control) — Outputs
C437-	A9 00	LDA	#\$00	B side (Bits 0-3; Data) — Inputs
C439-	9D 83 C0	STA	\$C083,X	
C43C-	A9 F0	LDA	#\$F0	
C43E-	9D 82 C0	STA	\$C082,X	
C441-	A9 04	LDA	#\$04	
C443-	9D 83 C0	STA	\$C083,X	
C446-	A9 10	LDA	#\$10	
C448-	9D 82 C0	STA	\$C082,X	Take hold and read lines
C44B-	A9 30	LDA	#\$30	to 5832 high.
C44D-	9D 82 C0	STA	\$C082,X	
C450-	A0 0C	LDY	#\$0C	
C452-	84 FF	STY	#\$FF	
C454-	A5 FF	LDA	#\$FF	
C456-	9D 80 C0	STA	\$C080,X	Read in all time & date registers,
C459-	8D 82 C0	LDA	\$C082,X	setting high bits and storing
C45C-	18	CLC		temporarily in a part of the
C45D-	69 80	ADC	#\$80	input buffer.
C45F-	A4 FF	LDY	#\$FF	
C461-	99 10 02	STA	\$0210,Y	
C464-	C6 FF	DEC	#\$FF	
C466-	A5 FF	LDA	#\$FF	
C468-	10 EC	BPL	\$C456	
C46A-	A0 AF	LDY	#\$AF	Put '/'s in correct place for format.
C46C-	8C 02 02	STY	\$0202	
C46F-	8C 05 02	STY	\$0205	
C472-	88	DEY		
C473-	8C 08 02	STY	\$0208	Put ','s in correct place for format.
C476-	8C 0E 02	STY	\$020E	
C479-	A9 AC	LDA	#\$AC	Put ',' in correct place for format.
C47B-	8D 08 02	STA	\$0208	
C47E-	AD 17 02	LDA	\$0217	
C481-	8D 01 02	STA	\$0201	
C484-	AD 1A 02	LDA	\$021A	
C487-	8D 03 02	STA	\$0203	
C48A-	AD 19 02	LDA	\$0219	
C48D-	8D 04 02	STA	\$0204	
C490-	AD 1C 02	LDA	\$021C	
C493-	8D 06 02	STA	\$0206	
C496-	AD 18 02	LDA	\$0218	
C499-	8D 07 02	STA	\$0207	Take time and date values
C49C-	AD 14 02	LDA	\$0214	from temporary positions and
C49F-	8D 0A 02	STA	\$020A	store in correct part of
C4A2-	AD 13 02	LDA	\$0213	input buffer.
C4A5-	8D 0C 02	STA	\$020C	
C4A8-	AD 12 02	LDA	\$0212	
C4AB-	8D 0D 02	STA	\$020D	
C4AE-	AD 11 02	LDA	\$0211	
C4B1-	8D 0F 02	STA	\$020F	
C4B4-	AD 15 02	LDA	\$0215	Tens of hours value — always working
C4B7-	38	SEC		in 24 hour format so remove bit 3
C4BB-	E9 08	SBC	#\$08	which is set.
C4BA-	8D 09 02	STA	\$0209	
C4BD-	A9 08	LDA	#\$08	
C4BF-	9D 80 C0	STA	\$C080,X	
C4C2-	8D 82 C0	LDA	\$C082,X	Tens of days value—reread and
C4C5-	29 FB	AND	#\$FB	mask off bits (29 day Feb.) which
C4C7-	18	CLC		might be set.
C4CB-	69 80	ADC	#\$80	
C4CA-	8D 00 02	STA	\$0200	
C4CD-	A9 10	LDA	#\$10	
C4CF-	9D 82 C0	STA	\$C082,X	Take read and hold lines to 5832 low.
C4D2-	A9 00	LDA	#\$00	
C4D4-	9D 82 C0	STA	\$C082,X	
C4D7-	A2 11	LDX	#\$11	Load x Reg with number of characters in buffer.
C4D9-	AD FC 03	LDA	\$03FC	
C4DC-	48	PHA		Reload status.
C4DD-	28	PLP		
C4DE-	A9 8D	LDA	#\$8D	
C4E0-	8D 11 02	STA	\$0211	Add carriage return character
C4E3-	58	CLI		
C4E4-	60	RTS		Allow interrupts again.
C4E5-	FF	???		Back from whence we came.

Listing 3

CLOCK IT TO ME

DOS file) and the writing position of that device is advanced twice. If we assume — indeed force — the output device to be the Apple screen, this can be overcome by backspacing the cursor twice with overprinting by blanks.

Listing 3 shows how all this is done in practice. Some of the register contents are based on material in the excellent *What's Where in the Apple?* (W F Luebbert, Micro Ink Inc, 1981). In particular this relates to the overprinting and backspacing requirement and the setting up of registers after the input buffer has been filled.

Although the normal format listing is given in Slot 4 address space, the code itself is slot-independent since it automatically finds which slot it's in, as described in the Apple reference manual. Note, however, that in this context the call to save register values (as suggested by Apple) does not need to be executed and has not been included.

You can easily modify the program to produce different formats; you have only to arrange the time and date differently in the input buffer and remember to add a carriage return character at the end and load the X register with the total number of characters. If you want to extract the day of the week from the 5832, remember that this is available in part of the input buffer as a product of the normal format code; add the following line to the Basic program in Listing 2: 60 PRINT PEEK(534)—176

This will print a number between 0 for Saturday and 6 for Friday, which can easily be decoded to print the day names, as shown at the end of Listing 1.

The ROM software developed here is totally Apple-dependent and can't be used with other machines. However, there's nothing to stop you developing your own intelligent firmware specific to your machine. Probably the easiest way to do this would be to find out how variables are stored and then declare the variables TI\$ and DA\$ as the first variables in any program, followed by CALLs to the ROM software or USR routine to access the card and fill in the variables whenever necessary.

Timing things

As I discussed last month, the MSM5832 can generate interrupts, allowing accurate time intervals to be produced or, as shown here, accurate timing between events. The interrupt rate should, of course, be as fast as possible for the highest resolution. The 5832 can generate interrupts at 1024 Hz so it's possible to time to the nearest millisecond, which should be accurate enough for most micro-based applications.

Listing 4 shows the machine code to set up and count interrupts, and Listing 5 gives a driving Basic program; both were written for the Apple II. The easiest way to understand how they work is to follow the listings through in the way that the computer would execute them. Starting with the Basic program in Listing 5, a variable, TI%, is zeroed. This is the actual timing variable

```

0000: 2 LST ON
0000: 3 *****
0000: 4 *
0000: 5 *
0000: 6 * DEMO TIMER.FP
0000: 7 *
0000: 8 *
0000: 9 *
0000: 10 *
0000: 11 * THIS CODE USES THE CLOCK II TO GENERATE INTERRUPTS WITCH
0000: 12 * ARE COUNTED INTO AN INTEGER VARIABLE IN THE TIME BETWEEN
0000: 13 * PUSH BUTTONS 0 AND 1 BEING PRESSED. THE INTERRUPT RATE
0000: 14 * IS 1024HZ AND THUS THE RESOLUTION IS (APPROXIMATELY) TO
0000: 15 * THE NEAREST 1/1000TH OF A SECOND.
0000: 16 *
0000: 17 * LIMITATIONS:
0000: 18 *
0000: 19 * ONLY WORKS WITH APPLESOFT.
0000: 20 * CLOCK II IS ASSUMED TO BE IN SLOT 4.
0000: 21 * PRESSING RESET STOPS INTERRUPTS.
0000: 22 * COUNTS INTO FIRST DECLARED VARIABLE
0000: 23 * WHICH MUST BE INTEGER.
0000: 24 *
0000: 25 *****
0000: 26 *
0000: 27 *
-----
NEXT OBJECT FILE NAME IS DEMO TIMER.FP,ORJO
0300: 28 ORG $300
03FE: 29 IRQVEC EQU $03FE ;INTERRUPT VECTOR
C061: 30 PBO EQU $C061 ;PUSH BUTTON 0 - START
C062: 31 PB1 EQU $C062 ;PUSH BUTTON 1 - STOP
C0C0: 32 SLOT4 EQU $C0C0 ;CLOCK CARD ADDRESS - SLOT NO. 4
0069: 33 VARTAB EQU $69 ;APPLESOFT VARIABLE TABLE POINTER
03FD: 34 TEMPY EQU $03FD ;TEMP STORE FOR Y REG
03FC: 35 TSTATUS EQU $03FC ;TEMP STORE FOR STATUS REG
0300: 36 *
0300: 37 *
0300:78 38 SEI ;NO INTERRUPTS
0301:AD 56 03 39 LDA IRQHAND+1 ;SETUP IRQ LINKAGE
0304:8D FE 03 40 STA IRQVEC
0307:AD 57 03 41 LDA IRQHAND+2
030A:8D FF 03 42 STA IRQVEC+1
030D: 43 *
030D: 44 * SETUP CLOCK FOR INTERRUPTS
030B: 45 *
030D:A9 00 46 LDA #$00 ;MAKE A SIDE ALL OUTPUTS
030F:8D C1 C0 47 STA SLOT4+1
0312:A9 FF 48 LDA #$FF
0314:8D C0 C0 49 STA SLOT4
0317:A9 04 50 LDA #$04
0319:8D C1 C0 51 STA SLOT4+1
031C:A9 00 52 LDA #$00 ;B SIDE SETUP, LOWER 4 BITS INPUTS
031E:8D C3 C0 53 STA SLOT4+3 ; UPPER 4 BITS OUTPUTS
0321:A9 F0 54 LDA #$F0
0323:8D C2 C0 55 STA SLOT4+2
0326:A9 04 56 LDA #$04
0328:8D C3 C0 57 STA SLOT4+3
032B:A9 0F 58 LDA #$0F ;TAKE ADDRESS LINES HIGH
032D:8D C0 C0 59 STA SLOT4
0330:A9 20 60 LDA #$20 ;TAKE READ LINE HIGH
0332:8D C2 C0 61 STA SLOT4+2
0335:A9 05 62 LDA #$05 ;INTERRUPT RATE OF 1024HZ
0337:8D C1 C0 63 STA SLOT4+1 ;ROUTED VIA CA1 OF PIA
033A: 64 *
033A:AD 61 C0 65 * START LDA PBO ;LOOK FOR START SIGNAL
033D:10 FB 66 BPL START
033F:58 67 CLI ;START TIMING
0340: 68 *
0340:AD 62 C0 69 * STOP LDA PB1 ;LOOK FOR STOP SIGNAL
0343:10 FB 70 BPL STOP
0345:78 71 SEI ;STOP HANDLING INTERRUPTS
0346:A9 00 72 LDA #$00 ;STOP PRODUCING INTERRUPTS
0348:8D C1 C0 73 STA SLOT4+1
034B:8D C3 C0 74 STA SLOT4+3
034E:8D C0 C0 75 STA SLOT4
0351:8D C2 C0 76 STA SLOT4+2
0354:60 77 RTS ;BACK TO BASIC
0355: 78 *
0355:4C 58 03 79 IRQHAND JMP IRQ ;DUMMY JUMP - NOT EXECUTED
0358: 80 *
0358: 81 * INTERRUPT ROUTINE
0358: 82 *
0358:BC FD 03 83 IRQ STY TEMPY
035E:08 84 PHP
035C:68 85 PLA
035D:8D FC 03 86 STA TSTATUS
0360:D8 87 CLD
0361:A0 03 88 LDY #$03 ;INCREMENT FIRST DECLARED
0363:B1 69 89 LDA (VARTAB),Y ;BASIC VARIABLE BY 1
0365:18 90 CLC ;IT IS ASSUMED TO BE OF
0366:69 91 ADC #$01 ;TYPE INTEGER
0368:91 69 92 STA (VARTAB),Y
036A:88 93 DEY
036B:B1 69 94 LDA (VARTAB),Y
036D:69 00 95 ADC #$00
036F:91 69 96 STA (VARTAB),Y
0371:AD C0 C0 97 LDA SLOT4 ;RESET 6821 IRQ REGISTERS
0374:AD C2 C0 98 LDA SLOT4+2
0377:AD FC 03 99 LDA TSTATUS ;LOAD 6502 REGISTERS
037A:48 100 PHA
037B:28 101 PLP
037C:AC FD 03 102 LDY TEMPY
037F:A5 45 103 LDA $A5 ;NOT FORGETTING ACC - SAVED BY MON
0381:40 104 RTI ;BACK TO LOOK FOR STOP SIGNAL

```

Listing 4

CLOCK IT TO ME

and is declared first since its absolute position in Apple memory can easily be established by referencing some page zero locations. Note also that it's an integer variable and values in it are contained in two bytes as opposed to 5-byte floating point numbers. Since we're counting interrupts, we only need an integer variable to store the total, but it does limit the maximum time that can be measured to about 32 seconds — more on this later.

Following the variable declaration, the screen is cleared, a title printed, the machine code routine (Listing 4) is loaded and the instructions are given (lines 10-50, Listing 5). The next line does a CALL to the machine code and attention now shifts to Listing 4.

First, the Apple has to be told the starting address of the routine which handles interrupts ('Setup Irq Linkage'). Then the clock card, which is assumed to be in slot 4, is configured to generate interrupts. The four interrupts which the 5832 generates all emerge together, one on each of the four data lines — see Figure 3 in last month's article. At this point (line 60 in Listing 4), the 5832 is generating interrupts but they are not yet linked to the 6502 interrupt line so the Apple knows nothing about them. Each interrupt rate line (data line) is connected to a 6821 interrupt input (Figure 2) and these are controlled by the control register — see Figure 1. Lines 62 and 63 set up the PIA's CA1 (which is connected to the 5832 data line 0 with the 1024 Hz signal) to pass interrupts through to the 6502 IRQ line. At this point the 6502 receives interrupts but ignores them as an SEI (set interrupt disable status) command was given earlier. The code now looks to see if pushbutton 0 is pressed (active when bit 7 is set) and, when this occurs, interrupts start to be handled (CLI) and a stop signal, on pushbutton 1, is looked for.

When an interrupt occurs, the Apple monitor automatically saves the contents of the 6502 accumulator and jumps through a page 3 address (which we earlier filled) to the routine starting at line 83 in Listing 4. This first stores the Y register and processor status and then increments the timing variable. It does this by using the Applesoft variable table pointer VARTAB, combined with

the knowledge that for an integer variable the data is held in bytes 2 and 3 relative to its entry; see page 137 in the Applesoft manual. Before finishing the interrupt routine, and apart from reinstating the temporarily-saved registers, the interrupt flag has to be cleared. When the 6821 actually detects an interrupt, a flag is set and, if the control register is suitably configured, the message that the flag is set is passed on to the micro via the interrupt line. The flag has to be reset by the micro referencing the appropriate peripheral register in the 6821, as it won't reset itself; if it isn't reset, the interrupt line will remain permanently low and the 6502 will continually execute the IRQ routine.

Eventually the stop button is pressed, interrupts are stopped and control returned to Basic. At this point, TI% now contains the number of interrupts which occurred at the 1024 Hz rate. The rest is easy: convert and round the number, print it, reset TI% to zero and start again. The CALL-950 is a reference to an Apple monitor routine that clears from the current cursor position to the end of the line.

Before you go crazy trying to test the timer, note the 32 seconds' maximum time between events. This is because the highest value an integer variable can hold is 32767 — 32767/1024 (the interrupt rate) gives 32 seconds. With a little ingenuity this can be doubled to 64 seconds with one extra line of Basic — try it!

Conversion to Microsoft Basic on a 6502 machine shouldn't be too difficult; you just need to check on where VARTAB is held in your machine's page zero memory and that integer variables are stored in the same way as in the Apple. This is something at the core of most Microsoft implementations and will probably not have changed. If you can't discover this information, then a less elegant solution is to count into two of your own declared locations at the beginning or end of the machine code and then PEEK the contents into Basic.

Concurrent processing

If an interrupt structure is properly set up, a micro can apparently handle two or more jobs at the same time. In reality, of course, the micro is only ever doing one job — but to the user it all happens so quickly that the distinction is invisible. For instance, how many

people know that their PET stops working on their problems every 1/60th of a second and goes off to update the time variable?

It would be useful to have the latest time and date continually displayed on the Apple screen, with the variables automatically updated, as on the PET, so that at any point they can be used without the hassle of INPUT TI\$, DA\$, etc. The code to do this, called CLOCK II.OBJO.HIGH, is given in Listing 6 and is quite long as I have endeavoured to keep the system as flexible as possible. Because of space limitations a complete and detailed breakdown cannot be given; the rationale for this being that most readers would prefer to have something relatively sophisticated rather than something simple (admittedly with a full explanation) and pointers to the brilliant things that are possible by adding 'a few extra lines'.

Unfortunately, at this level of software and machine interaction, it is hard to give directions for non-Apple users on how to amend the program. The best advice I can give is for you to look at the overall structure of the code and then work out the specifics for your own machine.

The CLOCK II.OBJO.HIGH code is managed from Applesoft Basic by the use of the '&' command which when encountered causes processing to jump to it via a page 3 vector. Program details and specific control mechanics are given at the beginning of Listing 6.

Before I discuss actual code, most users will want to type it in and play with it. Listing 7 is a short Basic program for testing Clock II Interrupt Handler. The section up to line 40 loads and prepares the system; those after 40 are for experimentation. Note that the machine code is BRUN not BLOADED and that ONERR is used if the clock card can't be found. Also note the setting up of the three interface variables (T%, TI\$ and DA\$) before any other variables are declared.

Lines after 40 demonstrate how to set up for printing and updating time and date to the screen. Line 110 appears somewhat lame; all that happens is that the program sits and waits at the INPUT. By this point in processing, the time and date are periodically (every second) being updated, while the programmer is collecting input from the user. In fact, your options here are many: you can interact with disk files, print results, use graphics etc. The only limitation is that the string variables TI\$ and DA\$ can't be put on the left hand side of an expression. Thus

```
TI$ = TI$ + "AM"
is not allowed, but
TT$ = TI$ + "AM" is.
```

Although not manipulated in the demonstration (but necessary in the program), print the three variables T%, TI\$ and DA\$ after you have stopped the program. T% will contain the number of seconds since &TI was given, and TI\$ and DA\$ will hold the last time and date.

The machine code in Listing 6 breaks down into three main parts: initialisation, '&' command handling, and interrupt handling.

Initialisation occurs automatically on BRUNning the code and hence is done only once. This includes discovering which slot the clock card occupies (from the EPROM software on it). If

```
5 TI% = 0
10 TEXT : HOME : INVERSE
15 PRINT " DEMONSTRATION TIMER " : NORMAL
20 D$ = CHR$(4) : PRINT D$ "BLOAD DEMO TIMER.FP.OBJO"
30 VTAB 4 : PRINT "PRESS GAME PUSH BUTTON 0 TO START"
40 PRINT "          AND"
50 PRINT "GAME PUSH BUTTON 1 TO STOP TIMER"
60 CALL 768
70 TI = TI% / 1024
75 TI = ( INT ( TI * 1000 + .5 ) ) / 1000
80 PRINT : PRINT : PRINT "WELL DONE!!!!!!"
90 PRINT : PRINT TI ; " SECS ELAPSED BETWEEN PUSHES " ;
92 CALL - 958 : PRINT " "
95 TI% = 0
100 GOTO 30 : REM REPEAT TIMING
```

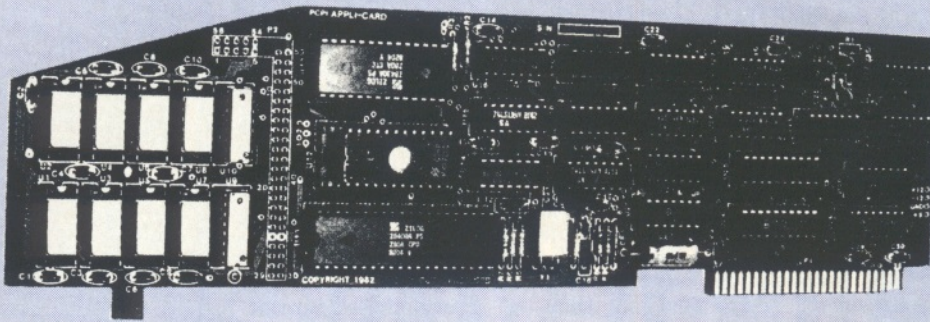
Listing 5

NEW... For Apple* II & III
 from **PERSONAL COMPUTER PRODUCTS, INC.**

\$495.00
 EX TAX

APPLI-CARD™

4 or 6 mhz Z-80 and 64K ON-CARD MEMORY



THE **ONE CARD** SOLUTION TO EXECUTE CP/M* APPLICATION PROGRAMS

- Development Languages Available
- Applications Available
- Compare Our Features And Get The Best Value And Performance For Your Money.

FEATURES	Z-CARD*	SoftCard*	APPLI-CARD
6 mhz Z-80 available	No	No	Yes
64K on-card memory	No	No	Yes
CP/M with card	Yes	Yes	Yes
SB-80* available	No	No	Yes
40 col. to 255 col. horizontal scroll	No	No	Yes
Choice of application	No	No	Yes
2K PROM on the card	No	No	Yes
Real time clock available on the card	No	No	Yes
Expansion interface on the card	No	No	Yes
70 col. upper & lower case	No	No	Yes
A self-contained Z-80A or Z-80B with memory	No	No	Yes
One-card WordStar* execution	No	No	Yes
63K available for program development or execution	No	No	Yes
Menu driven set up	No	No	Yes

FMS SOFTWARE AND HARDWARE FOR CP/M BASED SYSTEMS

95 CANTERBURY ROAD, MIDDLE PARK, VICTORIA 3206.
 Telephone: (03) 699 9899. Telex: AA31604.

the card is not found a jump to Basic error handling occurs. The interrupt and & page 3 vectors are then set up before a return is made to Basic.

When an '&' command is given processing automatically goes to ENTRYPT at line 183 in Listing 6. Time and date string variable table entries two and three are told where their data is held and the commands following the '&' are decoded. &E (end) is looked for, then &T and finally &I. If &T is found (printing of time and date), then all variables associated with the screen at that point are saved. Note that output does not have to go to the screen - it goes to the output device current when the '&' command was given. This is so 80-column video cards can be catered for, although this will involve extra coding and saving of 80-column screen variables. Whenever &T or &I are decoded there must be an associated number between 1 and 3 to set the rate of date and time revision. The GETRATE routine checks that a valid number exists and stores it for future reference.

Most of the remaining listing consists of subroutines. SETIRQ sets the clock card to interrupt at the specified rate. PRINT updates the time and date on screen by first saving current screen

variables, then substituting screen variables it saved when the '&' command was given. The time and date are then printed and original screen variables restored. GETIME sets up the clock card and reads the time and date into variable space - note that this does not leave the clock card as an interrupt producer and JSR SETIRQ should normally be executed after JSR GETIME.

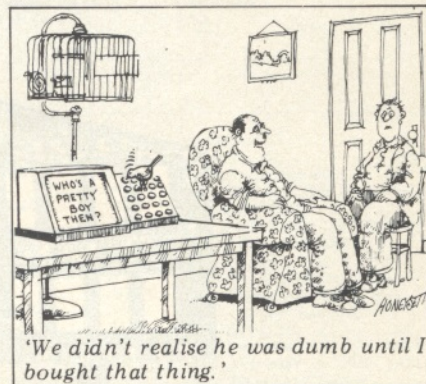
The final section is the interrupt handling code (IRQ), which saves all 6502 registers, updates the first integer variable by one and gets/stores the latest time and date. Depending on the contents of the PRINT.TD flag, the screen may also be updated. Finally the interrupt flags in the 6821 PIA are reset and the 6502 registers reinstated.

The Listing 6 machine code has not been optimised for speed or memory conservation as I suspect the savings would not justify the extra work. As an experiment I have benchmarked the system to determine the overhead involved in using this facility. Ordinarily an empty 1 to 50000 FOR...NEXT loop executes in 70.5 seconds. If &I1 is in operation this becomes 71 seconds and if &TI is used this comes to 72.5 seconds - a maximum increase in execution time of about three percent

which applies across the board regardless of specific coding.

Conclusion

These articles have described a low-cost, high-specification clock/calendar card for the Apple II and similar micros. The emphasis has been placed on providing suitable driving software both as an example and for direct application. Inevitably there are improvements to be made and I hope that users who develop routines and modifications will document them in a future issue...



```

0000: 2 LST ON
0001: 3 *
0002: 4 *****
0003: 5 *****
0004: 6 *
0005: 7 *****
0006: 8 *
0007: 9 *****
0008: 10 *
0009: 11 *****
0010: 12 *
0011: 13 *****
0012: 14 *
0013: 15 *****
0014: 16 *
0015: 17 *****
0016: 18 *
0017: 19 *****
0018: 20 *
0019: 21 *****
0020: 22 *
0021: 23 *****
0022: 24 *
0023: 25 *****
0024: 26 *
0025: 27 *****
0026: 28 *
0027: 29 *****
0028: 30 *
0029: 31 *****
0030: 32 *
0031: 33 *****
0032: 34 *
0033: 35 *****
0034: 36 *
0035: 37 *****
0036: 38 *
0037: 39 *****
0038: 40 *
0039: 41 *****
0040: 42 *
0041: 43 *****
0042: 44 *
0043: 45 *****
0044: 46 *
0045: 47 *****
0046: 48 *
0047: 49 *****
0048: 50 *
0049: 51 *****
0050: 52 *
0051: 53 *****
0052: 54 *
0053: 55 *****
0054: 56 *****
0055: 57 *****
0056: 58 *****
0057: 59 *****
0058: 60 *****
0059: 61 *****
0060: 62 *****
0061: 63 *****
0062: 64 *****
0063: 65 *****
0064: 66 *****
0065: 67 *****
0066: 68 *****
0067: 69 *****
0068: 70 *****
0069: 71 *****
0070: 72 *****
0071: 73 *****
0072: 74 *****
0073: 75 *****
0074: 76 *****
0075: 77 *****
0076: 78 *****
0077: 79 *****
0078: 80 *****
0079: 81 *****
0080: 82 *****
0081: 83 *****
0082: 84 *****
0083: 85 *****
0084: 86 *****
0085: 87 *****
0086: 88 *****
0087: 89 *****
0088: 90 *****
0089: 91 *****
0090: 92 *****
0091: 93 *****
0092: 94 *****

```

```

90 INVLG EQU $32
91 CH EQU $24
92 CV EQU $25
93 PZ1 EQU $FE
94 PZ2 EQU $FF
95 YSAV1 EQU $35
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *

```

CLICK IT TO ME

```

9324:91 69 189 STA (VARTAB),Y
9326:CB 190 INY
9327:AD 15 93 191 STA TIMEADDRESS+2
9328:91 69 192 STA (VARTAB),Y
932C:1A0 10 193 LDA #0
932E:A9 08 194 STA (VARTAB),Y
9330:91 69 195 INY
9332:CB 196 INY
9333:AD 17 93 197 STA DATEADDRESS+1
9336:91 69 198 STA (VARTAB),Y
933B:CB 199 INY
9339:AD 18 93 200 LDA DATEADDRESS+2
933C:91 69 201 STA (VARTAB),Y
933E:A9 00 202 LDA #000 ;STOP ANY PRINTING OF
9340:8D 85 92 203 STA PRINT.D ;TIME & DATE
9343:1A0 03 204 LDY #000
9345:81 88 205 LDA (TXTPTR),Y
9347:C9 45 206 CMP #645 ;#E* -- SEE IF &E
9349:1D0 14 207 BNE NEXT1 ;AND -- SO LOOK FOR ANOTHER COMMAND
934B: 208 *
934B: 209 * END INTERRUPTS
934B: 210 *
934B:AE 83 92 211 LDA SLOT
934E:A9 00 212 LDA #000
9350:9D 81 00 213 STA BASE+1,X
9353:9D 83 00 214 STA BASE+3,X
9356:9D 80 00 215 STA BASE,X
9359:9D 82 00 216 STA BASE+2,X
935C:4C A7 93 217 JMP RTS
935F: 218 *
935F: 219 *
935F:C9 54 220 NEXT1 BNE #654 ;#* -- SEE IF &T
9361:D0 29 221 BNE NEXT2 ;AND
9363: 222 *
9363: 223 * PRINTING OF TIME & DATE REQUIRED.
9363: 224 *
9363:AD 53 AA 225 LDA DOSCSWL ;SAVE CURRENT VALUE OF OUTPUT
9366:8D 89 92 226 STA CSWLPNT ;ADDRESS, THIS IS NORMALLY
9369:AD 54 AA 227 LDA DOSCSWL+1
936C:8D 8A 92 228 STA CSWLPNT+1 ;THE APPLE SCREEN (MONITOR)
936F:A5 32 229 LDA INVFLG ;SCREEN FUNCTION LOCATIONS
9371:8D 92 230 STA INVNT ;ARE SAVED AS WELL.
9374:A5 24 231 LDA CH
9376:8D 86 92 232 STA XPNT
9379:A5 25 233 LDA CV
937B:8D 87 92 234 STA YPNT
937E:A0 FF 235 LDA #FFF ;SET PRINT FLAG ON
9380:8D 85 92 236 STA PRINT.TD
9383:20 AE 94 237 JSR GETIME ;PUT TIME & DATE ON
9386:20 F2 93 238 JSR PRINT ;SCREEN IMMEDIATELY
9389:4C 90 93 239 JMP GETRATE
938C:C9 47 240 NEXT2 CMP #649 ;#* -- SEE IF &I
9390:1D0 17 241 BNE RTS ;AND -- NO MORE COMMANDS SO BACK TO BASIC
9390: 242 *
9390: 243 * INTERRUPT RATE CHECKING
9390: 244 *
9390:AD 01 245 GETRATE LDY #01
9392:81 88 246 LDA (TXTPTR),Y
9394:C9 31 247 CMP #631 ;#* (1 SECOND INTERRUPT RATE)
9397:81 08 248 BEQ DOWN
9399:C9 32 249 CMP #632 ;#* (1 MINUTE .....)
939C:A9 04 250 BEQ DOWN
939E:A9 04 251 CMP #633 ;#* (1 HOUR .....)
939F:C9 33 252 BNE RTS ;AND MORE OPTIONS SO BACK TO BASIC
939E:8D 84 92 253 DOWN STA RATE
93A0:8D 8A 93 254 JSR SETIRQ
93A4:5B 255 CLI ;OK START INTERRUPTING
93A7:4C 95 09 256 RTS ;MOVE TXTPTR & GO BACK TO BASIC
93AA: 257 *
93AA: 258 * SETUP INTERRUPTS
93AA: 259 *
93AA:AE 83 92 260 SETIRQ LDY SLOT
93AD:A9 00 261 LDA #000
93AF:9D 81 00 262 STA BASE+1,X
93B2:A9 FF 263 LDA #FFF
93B4:9D 80 00 264 STA BASE,X
93B7:A9 04 265 LDA #04
93B9:9D 81 00 266 STA BASE+1,X
93BC:A9 00 267 LDA #000
93BE:9D 83 00 268 STA BASE+3,X
93C1:A9 F0 269 LDA #F0
93C3:9D 80 00 270 STA BASE+2,X
93C6:A9 04 271 LDA #04
93C8:9D 83 00 272 STA BASE+3,X
93CB:A9 0F 273 LDA #0F
93CD:9D 80 00 274 STA BASE,X
93D0:A9 20 275 LDA #20
93D2:9D 82 00 276 STA BASE+2,X
93D5:AD 84 92 277 LDA RATE
93D8:C9 31 278 CMP #631 ;#*
93DB:A9 06 279 BNE TRY2
93DD:A9 0C 280 LDA #0C
93DE:9D 83 00 281 STA BASE+3,X
93E1:60 282 RTS
93E1:C9 32 283 TRY2 CMP #632 ;#*
93E4:4C 04 284 BNE MUSTBE3
93E6:A9 05 285 LDA #05
93E8:9D 83 00 286 STA BASE+3,X
93EC:70 287 RTS
93EC:A9 0C 288 MUSTBE3 LDA #0C
93EE:9D 81 00 289 STA BASE+1,X
93EF:160 290 RTS
93F1: 291 *
93F2: 292 * PRINT TIME & DATE (NORMALLY TO SCREEN)
93F2: 293 *
93F2: 294 *
93F2:A5 20 294 PRINT LDA UNDLFT ;SAVE EXISTING SCREEN
93F4:8D 92 92 295 STA LEFT ;FUNCTION DATA.
93F7:A5 21 296 LDA UNWDWTH
93F9:8D 93 92 297 STA WIDTH
93FC:A5 22 298 LDA UNBDTH
93FE:8D 90 92 299 STA TOP
9401:A5 23 300 LDA UNBDTH
9403:8D 91 92 301 STA BOTTH
9405:A5 35 302 LDA YSAU
9408:8D 94 92 303 STA YSAVTEMP
940B:A5 24 304 LDA CH
940D:8D 8E 92 305 STA XTEMP
9410:A5 25 306 LDA CV
9412:8D 8F 92 307 STA YTEMP
9415:A5 36 308 LDA CSWL
9417:8D 88 92 309 STA CSWLTEMP
941A:A5 37 310 LDF CSWL+1
941C:8D 8C 92 311 STA CSWLTEMP+1
941F:A5 32 312 LDA INVFLG
9421:8D 8D 92 313 STA INVTEMP
9424: 314 *
9424:A9 00 315 LDA #000 ;FILL SCREEN FUNCTION LOCATIONS
9426:85 20 316 STA UNDLFT ;WITH TIME & DATE
9428:85 22 317 STA UNBDTH ;PRINTING DATA.
942A:A9 28 318 LDA #28
942C:85 21 319 STA UNWDWTH
942E:A9 18 320 LDA #18
9430:85 23 321 STA UNBDTH
9432:AD 88 92 322 LDA INVNT
9435:85 32 323 STA INVFLG
9437:AD 86 92 324 LDA XPNT
943A:85 24 325 STA CH
943C:AD 87 92 326 LDA YPNT
943F:85 25 327 STA CV
9441:AD 89 92 328 LDA CSWLPNT
9444:85 36 329 STA CSWL
9446:AD 8A 92 330 LDA CSWLPNT+1
9449:85 37 331 STA CSWL+1
944B:20 22 FC 332 JSR VTAB
944E: 333 *
944E: 334 *
944E:A2 00 335 LDY #000 ;PRINT TIME & DATE

```

```

9450:8D 95 92 336 AGAIN LDA TIME,X
9453:3F 09 337 BEQ DATEPRT
9455:18 338 CLC
9456:69 80 339 ADC #60
945B:20 ED FD 340 JSR COUT
945B:EB 341 INX
945C:0D F2 342 BNE AGAIN
945E:EB 343 DATEPRT INX
945F:8D 95 92 344 LDA TIME,X
9462:F0 0D 345 BEG FINPRT
9464:E6 25 346 INC CV
9466:20 22 FC 347 JSR VTAB
9469:AD 86 92 348 LDA XPNT
946C:85 24 349 STA CH
946E:4C 50 94 350 JMP AGAIN
9471: 351 *
9471: 352 *
9471:AD 92 92 353 FINPRT LDA LEFT ;PUT BACK ORIGINAL
9474:85 20 354 STA UNDL I ;SCREEN FUNCTION DATA.
9476:61A9 93 92 355 LDA WIDTH
9479:85 21 356 STA UNWDWTH
947B:AD 90 92 357 LDA TOP
947E:85 22 358 LDA BOTTH
9480:AD 91 92 359 LDA UNBDTH
9483:85 23 360 LDA XTEMP
9485:AD 8E 92 361 STA CH
9488:AD 8F 92 362 LDA YTEMP
948B:85 25 363 STA CV
948F:AD 88 92 364 LDA CSWLTEMP
9492:85 36 365 STA CSWL
9494:AD 95 92 366 LDA CSWLTEMP+1
9497:85 37 367 STA CSWL+1
9499:AD 88 92 369 LDA INVTEMP
949C:85 32 370 STA INVFLG
949E:AD 94 92 371 LDA YSAVTEMP
94A1:85 35 372 STA YSAU
94A3:20 22 FC 373 JSR VTAB
94A6:60 374 RTS
94A7: 375 *
94A7: 376 *
94A7: 377 *
94A7:9D 80 00 378 FETCH STA BASE,X ;GET SPECIFIC TIME/DATE ELEMENT
94A8:18 82 00 379 LDA BASE+3,X
94A9:60 380 RTS
94AB: 381 *
94AB: 382 *
94AB:AE 83 92 383 GETIME LDY SLOT ;SETUP REGISTERS TO GET
94B1:A9 00 384 LDA #000 ;TIME & DATE
94B3:9D 81 00 385 STA BASE+1,X
94B6:A9 FF 386 LDA #FFF
94B8:9D 80 00 387 STA BASE,X
94BB:A9 04 388 LDA #04
94BD:9D 81 00 389 STA BASE+1,X
94C0:A9 00 390 LDA #000
94C2:9D 83 00 391 STA BASE+3,X
94C5:A9 F0 392 LDA #F0
94C7:9D 82 00 393 STA BASE+2,X
94CA:A9 04 394 LDA #04
94CC:9D 83 00 395 STA BASE+3,X
94CF:A9 10 396 LDA #10
94D1:9D 82 00 397 STA BASE+2,X
94D4:A9 30 398 LDA #30
94D6:9D 82 00 399 STA BASE+2,X
94D9: 400 *
94D9:A9 05 401 LDA #05 ;LOAD & STORE TIME &
94DB:8D A7 94 402 JSR FETCH ;DATE REGISTERS
94DE:18 403 CLC
94DF:A9 F8 404 ADC #F8
94E1:8D 95 92 405 STA TIME
94E4:A9 04 406 LDA #04
94E6:20 A7 94 407 JSR FETCH
94E9:8D 96 92 408 STA TIME+1
94EC:A9 2E 409 LDA #2E
94EE:8D 97 92 410 STA TIME+2
94F1:8D 9A 92 411 STA TIME+5
94F4:A9 03 412 LDA #03
94F6:20 A7 94 413 JSR FETCH
94F9:8D 98 92 414 STA TIME+3
94FC:A9 02 415 LDA #02
94FE:20 A7 94 416 JSR FETCH
9501:8D 99 92 417 STA TIME+4
9504:A9 01 418 LDA #01
9506:20 A7 94 419 JSR FETCH
9509:8D 98 92 420 STA TIME+6
950C:A9 00 421 LDA #00
950E:20 A7 94 422 JSR FETCH
9511:8D 9C 92 423 STA TIME+7
9514:A9 08 424 LDA #08
9516:20 A7 94 425 JSR FETCH
9519:29 F8 426 AND #F8
951B:8D 9E 92 427 STA DATE
951E:A9 07 428 LDA #07
9520:20 A7 94 429 JSR FETCH
9523:8D 9F 92 430 STA DATE+1
9526:A9 2F 431 LDA #2F
9528:8D A0 92 432 STA DATE+2
952B:8D A3 92 433 STA DATE+5
952E:A9 0A 434 LDA #0A
9530:20 A7 94 435 JSR FETCH
9533:8D A1 92 436 STA DATE+3
9536:A9 09 437 LDA #09
9538:20 A7 94 438 JSR FETCH
953B:8D A2 92 439 STA DATE+4
953E:A9 0C 440 LDA #0C
9540:20 A7 94 441 JSR FETCH
9543:8D A4 92 442 STA DATE+6
9546:A9 0B 443 LDA #0B
9549:20 A7 94 444 JSR FETCH
954B:8D A5 92 445 STA DATE+7
954E: 446 *
954E: 447 *
954E:A9 10 447 LDA #10 ;DESELECT READING OF TIME
9550:9D 82 00 448 STA BASE+2,X ;& DATE REGISTERS
9553:A9 00 449 LDA #00
9555:9D 82 00 450 STA BASE+2,X
9558:160 451 RTS
9559: 452 *
9559: 453 * INTERRUPT HANDLING
9559: 454 *
9559:86 46 455 IRQ STX XREG ;SAVE 6502 REGISTERS
955B:84 47 456 STY YREG ;NOTE THAT ACC IS SAVED
955D:08 457 PHA ;AUTOMATICALLY BY APPLE MONITOR)
955E:168 458 PLA
955F:85 48 459 STA STATUS
9561:08 460 CLD
9562:A0 03 461 LDY #03 ;INCREMENT FIRST DECLARED
9564:81 69 462 LDA (VARTAB),Y ;SIMPLE VARIABLE WHICH
9566:18 463 CLC ;IS ASSUMED TO BE INTEGER
9567:69 01 464 ADC #01
956A:91 69 465 STA (VARTAB),Y
956B:88 466 BEY
956C:81 69 467 LDA (VARTAB),Y
956E:69 00 468 ADC #00
9570:91 69 469 STA (VARTAB),Y
9572:20 AE 94 470 JSR GETIME ;UPDATE TIME & DATE STRINGS
9575:AD 85 92 471 LDA PRINT.TD ;WANT TO PRINT TIME & DATE-Y
9578:110 03 472 BFL OUT ;AND
957A:20 F2 93 473 JSR PRINT ;GO PRINT TIME & DATE
957D:20 AA 93 474 OUT JSR SETIRQ ;SETUP IRQ'S AGAIN (WHICH GETIME
9580: 475 * ;DESTROYED)
9580:AE 83 92 476 LDY SLOT ;RESET 6821 IRQ REGISTERS
9583:8D 80 00 477 LDA BASE,X
9586:8D 82 00 478 LDA BASE+2,X
9589:A6 46 479 LDX XREG ;LOAD 6502 REGISTERS
958B:A4 47 480 LDY YREG ;+
958D:A5 48 481 LDA STATUS ;+
958F:A8 48 482 PHA ;+
9590:28 483 PLP ;+
9591:A5 45 484 LDA ACC ;-(INCLUDING ACCUMULATOR)
9593:40 485 RTI ;BACK TO BASIC.

```

Listing 6



Run rings around other Apple users

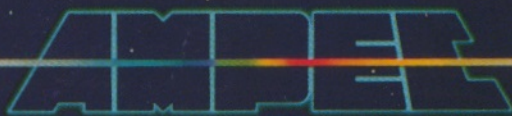
Introducing a new and powerful Apple-compatible hard-disk drive . . . the RO 100 series from Ampec. This drive/subsystem offers a complete add-on mass storage subsystem for the Apple II microcomputer.

Available in capacities from 4 Mbytes to 16 Mbytes, these Saturn hard-disk drives have on-board microprocessors and feature fast access times to give you considerable improvement in system throughput.

This current release of the Saturn subsystem includes DOS 3.3, Apple, Pascal and CP/M software support. With the subsystem installed, and using the supplied support programmes, you can have up to 16 Mbytes of additional disk storage — that's roughly equivalent to

adding 80 Apple mini floppy disk drives to your Apple computer.

Now at your Ampec dealer — the Saturn RO 100 disk drives. They really put the icing on the cake — or, if you like, the cream on your Apple pi.



AMEPEC

The absolute in peripheral thinking.