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.

Editor: Sean Howard
Technical Editor: Ian Davies
Advertising: Steve Kohn
Printed by: Lewis Printers
Produced under licence from Ebelin Productions

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

Material contained within Australia's Personal Computer is protected by the Commonwealth Copyright Act 1965. 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

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 Imagingon 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.

Denis 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

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

$2950 INC. TAX

ATARI

- 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) 659 4818
HOBART (Tas.) — James Powell-Davies (002) 31 6818

Australian Personal Computer  Page 3
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
operating costs significantly. A very
soothing fact for someone who thinks of a
computer as more than a toy.

SINGLE KEY APPLESOF
** 3 ** The DIGICARD 80 is the ONLY
eighty column terminal that creates single
key commands in APPLESOF Basic. Just
type ESC and then one other letter to issue
one of eighteen commands. ESC Icon 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 inverse 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.

RESPONS 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 eighty or eighty 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 communicate 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 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 consump-
tion of the DIGICARD 80 means your
APPLE will run cool all day long which is
a comforting thought.

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.

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.

12 MONTH GUARANTEE
** 11 ** All DIGICARD products are designed and manufactured in Australia by Maclagen 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.

<table>
<thead>
<tr>
<th>Drive</th>
<th>Price</th>
<th>Size</th>
<th>Price with Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELITE 1</td>
<td>$640</td>
<td>(163kb)</td>
<td>$790</td>
</tr>
<tr>
<td>ELITE 2</td>
<td>$830</td>
<td>(326kb)</td>
<td>$990</td>
</tr>
<tr>
<td>ELITE 3</td>
<td>$1090</td>
<td>(652kb)</td>
<td>$1250</td>
</tr>
<tr>
<td>CONTROLLER ONLY</td>
<td>$160</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(All prices include Sales Tax)

CONTACT:
Computer Edge Pty Ltd.
364 FERRARS STREET, ALBERT PARK, VIC 3206. Telephone: (03) 690 1477
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 Zofarly. 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! This 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 gassy 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 in the machine when we received it. Neither card is "flawed" for the Digicard. Both provide excellent high resolution graphics, with the ability to switch between the hires screen and the lo-res screen at will. Both boards also allow you to run two monitors -- one for hires output and output for lo-res. Toggling from one logical screen to the other did produce a slight shimmer 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 slightly 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.
COMMUNICATIONS

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):

\[
\begin{align*}
\text{space} & \quad \text{space} \quad = \quad \text{space} \\
\text{space} & \quad \text{space} \\
\text{space} & \quad \text{space}
\end{align*}
\]

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 no 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 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 ACP 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 space. 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 mountain-eering. - Dick Pountain.

...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 SYS-PSNS(W) THEN 
N=C(P(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 for

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 ZDRAV, APPLE WRITER II and WORDSTAR.

A superb set of 128 upper and lower case characters in a 9 x 11 dot matrix, including 3 dot ascenders. Shift and shift lock for upper and lower case. Source switches both hardware and software between 40 x 24 and 80 x 24 screen.


Plus: Now available

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

DISTRIBUTED BY:

IMAGINEERING.
(02) 368 3111
22 SIR JOHN YOUDE cres, WOOLLOOMOOLLO 2011
OR
FROM YOUR LOCAL APPLE DEALER NOW.

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
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 resume 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...
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 (i.e., the address of the first location in the PIA).

Other, more minor, amendments will also be necessary. In line 15 the string variable BELLS? is set to beep the Apple's speaker; if your machine has no such facility then set BELLS? 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 300) and two lines have been deleted 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 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-unique driving program) and this should be compared with the much longer and more
CLOCK IT TO ME

fiddy 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 ($80-$FF) 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 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

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>DIL Switch Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>3FF</td>
<td>0</td>
</tr>
<tr>
<td>400</td>
<td>0</td>
</tr>
<tr>
<td>402</td>
<td>0</td>
</tr>
<tr>
<td>403</td>
<td>1</td>
</tr>
<tr>
<td>404</td>
<td>2</td>
</tr>
<tr>
<td>405</td>
<td>3</td>
</tr>
<tr>
<td>406</td>
<td>4</td>
</tr>
<tr>
<td>407</td>
<td>5</td>
</tr>
<tr>
<td>408</td>
<td>6</td>
</tr>
<tr>
<td>409</td>
<td>7</td>
</tr>
<tr>
<td>410</td>
<td>8</td>
</tr>
<tr>
<td>411</td>
<td>9</td>
</tr>
<tr>
<td>412</td>
<td>10</td>
</tr>
<tr>
<td>413</td>
<td>11</td>
</tr>
<tr>
<td>414</td>
<td>12</td>
</tr>
<tr>
<td>415</td>
<td>13</td>
</tr>
<tr>
<td>416</td>
<td>14</td>
</tr>
<tr>
<td>417</td>
<td>15</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>TEXT</th>
<th>D</th>
<th>$ = CHRS (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>PRINT B$&quot;IN84&quot;</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
<td>INPUT DA$+TI$</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>PRINT B$&quot;IN80&quot;</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>PRINT &quot;DATE&quot;,DA$</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>PRINT &quot;TIME&quot;,FT$</td>
</tr>
<tr>
<td>60</td>
<td>70</td>
<td>END</td>
</tr>
</tbody>
</table>

Listing 2

<table>
<thead>
<tr>
<th>TEXT</th>
<th>D</th>
<th>$ = CHRS (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>PRINT B$&quot;IN84&quot;</td>
</tr>
<tr>
<td>20</td>
<td>30</td>
<td>INPUT DA$+TI$</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>PRINT B$&quot;IN80&quot;</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>PRINT &quot;DATE&quot;,DA$</td>
</tr>
<tr>
<td>50</td>
<td>60</td>
<td>PRINT &quot;TIME&quot;,FT$</td>
</tr>
<tr>
<td>60</td>
<td>70</td>
<td>END</td>
</tr>
</tbody>
</table>

Listing 3

Solve interrupts. Load space character.
Print space at current output position.
Backspace cursor.
Repeat above 4 lines.
Save status.
Find which slot we are in.
Configure PIA as follows:
A side (6522 addresses) — Outputs
B side (Bits 4-7 Control) — Outputs
B side (Bits 0-3, Data) — Inputs
Take hold and read lines to 6522 high.
Configure PIA as follows:
A side (6522 addresses) — Outputs
B side (Bit 7) Control — Outputs
B side (Bits 0-3, Data) — Outputs
Take time and date values from temporary positions and store in correct part of input buffer.
Tens of hours value — always working in 24 hour format so remove bit 3 which is set.
Tens of days value, read and mask off bits (29 day Feb.) which might be set.
Take read and hold lines to 6522 low.
Load $E flag with number of characters in buffer.
Reload status.
Add carriage return character.
Allow interrupts again.
Back from whence we came.
CLOCK IT TO ME

DOS file) and the writing position of that device is advanced twice. If we assume the index register — 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 5682, 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 be easily 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 T1 and DAS 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 MSMS5832 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 highest resolution. The 5682 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, T1%, is zeroed. This is the actual timing variable

Listing 4
CLOCK IT TO ME!

and is declared first since its absolute position in the Applesoft basic code is 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 don’t use interrupts, what we need only 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 Iq Linkage’). Then the clock card, which is assumed to be in slot 4, is configured to generate interrupts for interrupts (‘Gap 821 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 3) 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 63 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 in 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, T1% now contains the number of interrupts which occurred at 1024 Hz rate. The rest is easy: convert and round the number, print it, reset T1% to zero and start again. The CALL-950 is a reference to an Apple monitor routine that clears 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 one of your own-declared locations at the beginning or end of the machine code and then SEEK 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 in programs. The code to do this, called CLOCK IL.OBJ.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 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 who want to add this to 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 specifies for your own machine. The CLOCK IL.OBJ.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 of the variables 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 the 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 is not compatible. Also note the setting up of the three interface variables (T%, T1% and DAS) before any other variables are declared.

Lines after 40 demonstrate how to set up for timing 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 the OptionOne card etc. The only point here is that the string variables T1% and DAS can’t be put on the left hand side of an expression. Thus T1% = T1% + "AM" is not allowed, but TTS = TTS + "AM" is.

Although not manipulated in the demonstration (but necessary in the program), print the three variables T%, T1% and DAS after you have stopped the program. T% will contain the number of seconds since &TI was given, and T1% and DAS will hold the last time and date.

The machine code in Listing 6 breaks down into three main parts: initialisation of ‘&’ 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
NEW... For Apple* II & III
from PERSONAL COMPUTER PRODUCTS, INC.

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.

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

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 ENTR YPT at line 183 in Listing 6. Time and date string variable table entries two and three are tied where their data is held and the command 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 read in. 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 GETDATE 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 interrupts at a specified time. 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. GETTIME 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 GETTIME.

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 6521 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 &I is in operation this becomes 71 seconds and if &T 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. Hopefully there are improvements to be made and I hope that users who develop routines and modifications will document them in a future issue...

"We didn't realise he was dumb until I bought that thing!"
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 pie.

AMPEC
The absolute in peripheral thinking.