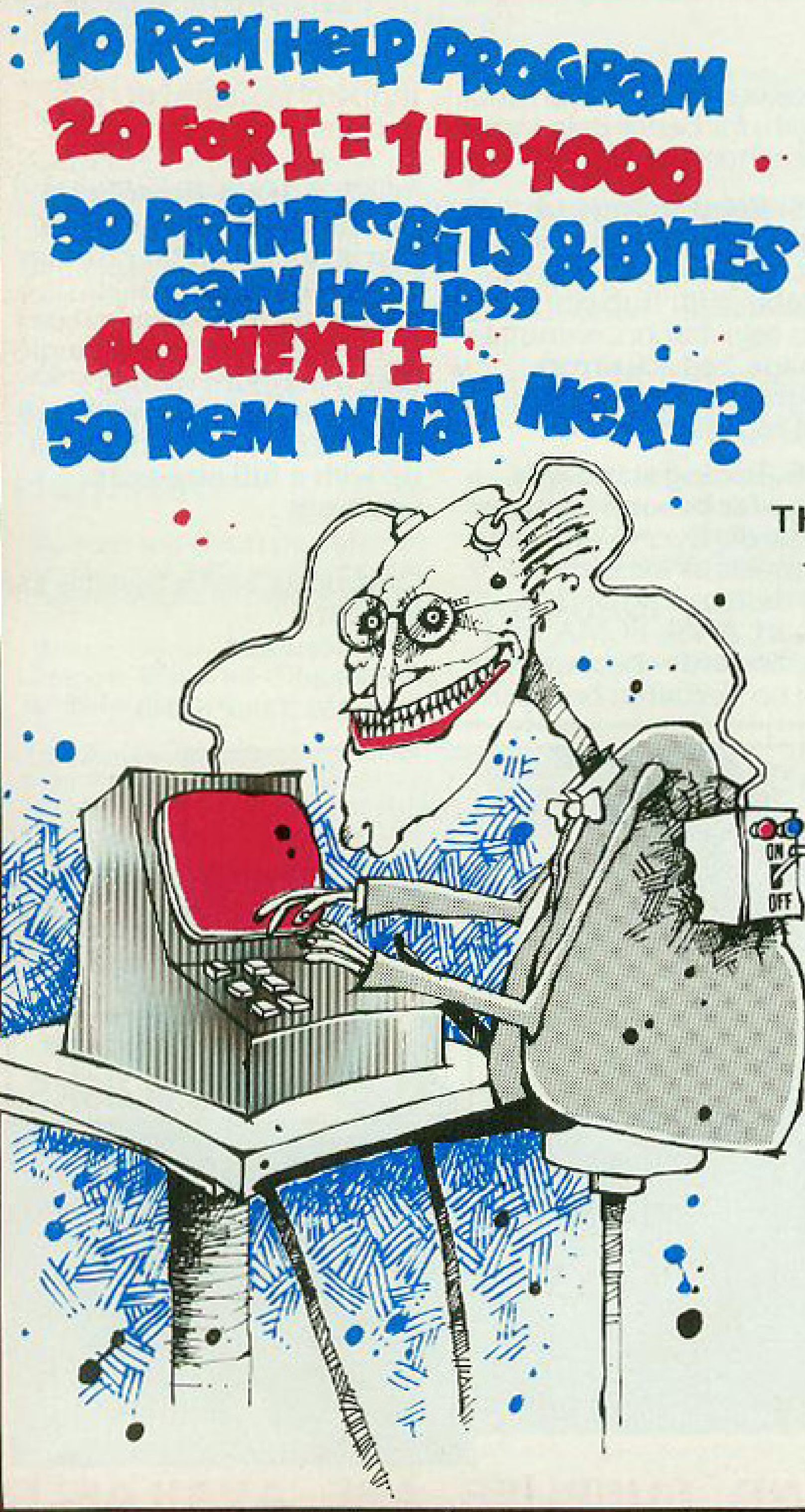
[redacted] ZEALAND'S PERSONAL COMPUTER MAGAZINE

BITS & BYTES

Issue No.3, November 1982: $1.00



A prof of a micro
-the Microprofessor studied

Micro computers Galore!

Computer Games

School Competition

THE B.B.C. MICROCOMPUTER
-two points of view

How to select a small business microcomputer

Micros in the office

And now for the performance of a lifetime!

Introducing 7 data-shielding improvements from Verbatim for greater disk durability and longer data life.

Improvements to protect your data from head-to-disk abrasion. Improvements to shield your data against loss due to environmental conditions. Improvements that will deliver a longer lifetime of trouble-free data recording, storage and retrieval than ever before possible.

1. A longer-lasting lubricant. Our new lubricant is more resistant to diffusion, to protect against data-destroying head-to-disk contact.

2. An improved liner. Our new liner cleans and removes debris better. It also enables more lubricant to reach the recording head, protecting against head wear.

3. A thicker, more durable coating. Our disks have a more uniform oxide coating for more adhesive and cohesive strength. We've also made it thicker; providing 10% more protective lubricant and an optimized signal resolution for the new recording heads.

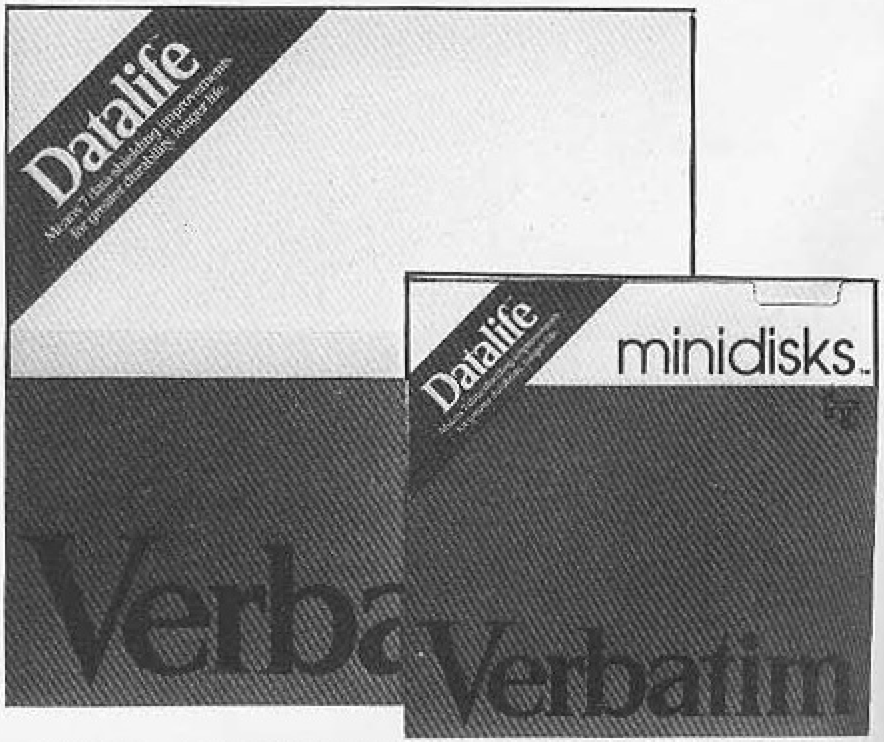
4. Advanced polishing techniques. Our burnishing makes our disks uniformly smooth, for better data transfer, less head wear.

5. Reinforcing hub rings. For critical applications, Verbatim disks are available with hub rings to aid in registration, eliminate slippage, reduce errors, and give you better alignment repeatability.

6. Testing standards that go far beyond industry standards. Every Verbatim disk meets or exceeds the most demanding of IBM, Shugart, ANSI, ECMA, and ISO standards–because we insist on Verbatim being the industry standard of excellence.

7. A 100% Error-Free Certification. Our certification is not based on random sampling, or statistical averaging, it is based on extensive testing of every single disk. Therefore we can state that our disks really are 100% error-free, and back it up with a **full one-year warranty.**

Compudata Media Systems Ltd.
Box 3273
48 Ellice Road
Auckland
4446-085 Telex 60044



VERBATIM DISKS AND SUPPLIES ARE AVAILABLE THROUGH ALL WHITCOULLS COMMERCIAL STORES

BITS & BYTES Contents

"BITS & BYTES" is published monthly, except January, by Neill Birss and Paul Crooks.
Address: Box 827, Christchurch
Subscription rate: $8 a year (11 issues) adults
$6 a year for school pupils

Representatives (advertising and editorial):
Auckland - Cathy and Selwyn Arrow Ph: 491-012
Hamilton - Dave Brown and Mark Bullivant. Ph: 54-692
Wellington - Annie Carrad Ph: 723-431

Advertising:
South Island-Graham Beecroft. Phone 554-265 Chch.
Or contact our advertiser co-ordinator.
Dave Telford Phone 794-787 Chch.

Technical editor: Chris O'Donoghue
Production: Roger Browning, Graeme Patterson, David Bimler.
Typesetting: Gail Rankin
Cover: Trace Hodgson Graphics: Harriet Gurney.

Subscription addresses:
When sending in subscriptions please include postal zones for the cities. If your label is incorrectly addressed please send it to us with the correction marked.

Distribution inquiries: Bookshops-Gordon and Gotch, Ltd. Computer stores- direct to the publishers.

Printed in Christchurch by D. N. Adams, Ltd.

Getting the good news out

Some readers will have received their October copies of "BITS & BYTES" late. We apologise for this. There will be delays with new subscribers from time to time as we get their details into our microcomputer addressing system.

The addresses on some subscription cards have not helped. They have brought home to the editors how many duplicated place names there are in New Zealand: Palmerstons, Havelocks, and Silverstreams to name a few.

If your magazine has reached you by a circuitous route; if one of the digits in the address number is wrong; even if we have misspelt your name or street and it irritates you... please send in the incorrect label with the correct address marked on it, and we'll make the change. Postal zones would also be helpful, as these are required in the posting of registered publications.

Contributions still are coming in, but we want more programs and articles for hand-held computers, for Sinclairs, and we'd like to hear something from Poly users and from enthusiasts with machines other than those usually catered for. Programs should be reasonably short (they should not take up more than a page of the magazine). Programs will be reproduced directly in "BITS & BYTES", so please make sure they are listed on white paper in strong, black lettering. Please try to keep the lines from being too long; this way they do not have tc be reduced so much for reproduction.

Originality and copyright are two important concepts to bear in mind when submitting articles and programs for publication anywhere. One submitted contribution "BITS & BYTES" will not be using is a game filched from a recent edition of a British microcomputer magazine. You may develop others' progams. But make sure there is a strong element of originality in your contribution, and in the covering remarks mention your source. If in doubt, let us decide whether you have developed it sufficiently.

Please keep the material coming in. We are delighted to hear your comments, suggestions, and any criticism of "BITS & BYTES". We would welcome more letters to the editors for publishing: others may be interested in your views on the computer tax, on various products, on other matters. We would welcome more inquiries from potential contributors. We would welcome more use of our classified advertisements, which are free to non-commercial readers (for up to 20 words). We would welcome more news about what various schools are doing with microcomputers, and in computer education generally.

Finally, a special welcome to this month's guest readership: the nation's accountants. Each month "BITS & BYTES" sends sample introductory copies to a selected group. This month it is accountants. Next month, we will be sending them to farmers, so if you have any country friends interested, or potentially interested in microcomputers, drop us a line and we will put them on the mailing list.

COMING UP IN BITS & BYTES

Farmers

Our focus is on microcomputers and farming in December with an in-depth look at the growing amount of software becoming available for farm applications.

Business

John Vargo and Kerry Marshall continue their informative series on selecting a small business microcomputer and designing business software respectively.

Beginners

More BASIC for beginners. Write and tell us if there are any terms you would like added to our glossary.

Education

Continuing our look at how different schools are using microcomputers.

MICROCOMPUTERS GALORE!

The spate of new microcomputer releases in New Zealand shows no signs of easing.

B.B.C. Microcomputer

Released in September the cheaper model of the B.B.C. is retailing in New Zealand for $1595. See our feature this month.

HX-20 Portable Computer

Due to be released in New Zealand this month (see our preview this issue) it will also retail around $1600.

Panasonic JR 100

A new personal computer from National Panasonic of Japan. Efforts are being made to have it on sale before Christmas at a retail price of $595.

The JR 100 has 16K of RAM, 8K of ROM, a cassette interface, an unusual keyboard with rubber keys, each key having five shift modes incorporating numerous graphic symbols.

Texas Instruments 99/4A

Expected to be on sale here in January and February for "less than a thousand dollars" according to sole New Zealand distributors, Whitcoulls.

The TI 99/4A is to the forefront of the 'home computer' sales boom in the United States where its price has dropped from $500 a year ago to $199 today. Heavy demand in Australia has held-up its New Zealand release but when it does go on sale here 150 plug-in program packages (including games) will also be available according to Whitcoulls.

Manurewa High and the language of the future

By R. THORNTON

If you were to walk into the computer room at Manurewa High School you could be excused for thinking that you had interrupted a foreign language class. What with Peeks and Pokes, Roms and Rams, Bits and Bytes it could almost pass as a recipe for Kung Fu Agriculture in pieces - but it isn't. It's the language of the future, the communication skill necessary to survive the 80's and beyond, the key to unlocking future discoveries... well that's what everyone keeps telling us, but is it true?

Just as the majority of people who drive motor cars don't understand how to do a valve grind or fine tune the motor, so it is with computers. A small percentage of computer users become 'nuts' or 'freaks' and specialised enough to modify hardware or develop software but the larger percentage will use the computer as a convenient, very fast, machine.

At Manurewa High we are aiming to cater for all levels of pupil interest and enthusiasm. From the whizz kid with Arrays and For/Next loops to the beginner learning to push the right button, pupils are being exposed to the technology of tomorrow which they are hungry to receive. Today's young person is the computer revolution - they don't have to adapt to it. They accept the computer as part of their future and are ready to use it as naturally as we use the ballpoint pen.

The adjustments to the technology are having to be made by those who have missed out on those informative space invader, galaxian parlour teens. The generations who had to settle for make believe Daniel Boone and the Lone Ranger can't hope to compete with the realism of Buck Rogers and the Force being with you!

But does that mean a generation gap? I don't think so. Each generation has something to contribute to the other's learning and it really is quite simple to make a computer do the right things.... a lot simpler than tuning a car and slightly more difficult than turning on the TV.

There is a danger, however, that computer 'experts' can become elites, separated from the general mass, unable to communicate with mere humans, saving all their energies for their precious machine moments.

Our computer awareness course for all fourth formers aims to spread computer confidence across all abilities and to break down the many barriers and myths associated with computer technology, i.e. you don't have to be male, skinny, wear spectacles, talk and write Japanese before you are allowed to touch the might beasts from computer land - and no they don't bite or is that byte?

Computer clubs have also been started, using every available moment, (the computers are in use all day every day) to allow pupil exploration of the computer's capabilities and to enjoy the many games that make learning fun (oops!! I do hope that learning is allowed to be enjoyable... because our pupils have a ball!!)

At the senior level we offer Form 6 Applied Mathematics where a course of computer related study is undertaken so that the computers are used to assist learning and to aid analysis in other subject fields.

The real 'experts' at Form 7 level use the computers within their Applied Mathematics course - where most computers first gained their entry into the school environment.

We have a collection of computers - three Commodores, one Apple, one Sinclair, and with the advent of the new low cost home computers, like the Commodore VIC 20, we will hopefully be able to purchase more.

The revolution has arrived but it isn't the computer. The revolution is within the minds of our young people who see the exciting potential of their future. Anyone involved in the education of such young people has a responsibility to make sure that they don't just learn how to push buttons and manipulate machines but to be aware of the impact, influence and consequences that this technology will have on their world of tomorrow.

Mr Thornton is teacher in charge of computer education programmes at Manurewa High School.

THE BBC: Here is the news...

As read by Pip Forer

One of the great divides in civilisation today splits those who follow Z-80 based microcomputers from those who owe their allegiance to 6502 based machines.

First with the Pet and latterly the Atari and Apple I have always been a 6502 man.

The newly arrived machine from Acorn Computers, the BBC Microcomputer, can only serve to confirm me in my established ways.

The BBC microcomputer reviewed here is a so-called model B. There are two current models. The B ($1995 retail) has larger memory and enhanced facilities from the A($1595) although A's can be upgraded to B's at a later date.

These prices cover the keyboard and processor which come built into a single unit. This unit will deliver colour graphics through a normal colour television receiver (although output plugs for video monitors and RGB monitors are also standard). It also has an interface to an audio cassette tape recorder for data and program storage.

One could spend some time and much space discussing the capabilities of the BBC computer. Perhaps at a later date that should be a regular feature of "Bits & Bytes.

This review just briefly outlines some of the features that make this machine stand out. These boil down to three clear characteristics.

Firstly the machine has been laid out with a clear growth path in mind: you can see how the design has room for expansion.

Secondly the machine as it stands is highly flexible. It possesses the 'openness' of the Apple plug-it-in peripheral card philosophy, the screen software flexibility of the Atari 400 and 800 machines and the teletext compatibility of Poly.

Finally it seems to have taken all the little edges that most current machines possess, rounded them off and then French polished them.

Firstly a look at the outside. A smart looking machine with a case that is a little too flexible (a notice warns against standing heavy objects on it). The two things that catch the eye are the keyboard and the enormous number of sockets, plugs and pins that appear to offer great built-in possibilities for interfacing.

The keyboard is pleasant with a good feel to it. No numeric keypad but nonetheless a good range. Full upper and lower case and several non-standard symbols including a pound sign for those writing to maiden aunts in Epping. There is a row of special function keys that can be used to allow the user direct access to special routines or give 'at a stroke' keyword typing. Naturally there is a shift key and both shift lock and capitals lock buttons with associated red LED status lights.

There are also four 'arrow' keys for moving the cursor around the screen plus copy and delete buttons. In editing these six keys work remarkably well. If a program line is anywhere on the screen the arrow keys can be used to place the cursor at the start of the line for editing. Hitting copy then moves the cursor along the line and copies the letters scanned to a blank space below. If the user wishes to insert a new instruction he simply types it in and it appears in the updated copy low down. To omit parts of the original text the arrow keys can be used to skip over these in the original line and copy then resumes copying to the new version. This makes editing and correction a breeze. In fact far more of a breeze than trying this verbal explanation of how it all works.

All screen interaction is also speeded up by the fact that any key that is held down for any time automatically repeats. The arrows can thus be used to move over the screen at speed and with great ease.

Continued on page 6

Pip Forer at the keyboard of the BBC micro he tested

As read by Chris O'Donoghue

The first thing that struck me when I pulled the BBC microcomputer from its polystyrene box was the keyboard. It looked good. It was even better than I thought, with a most professional feel.

The second was while looking for any stickers headed Warning, Important, or such. I noticed one saying not to put anything on top of the BBC micro. This was a minor disappointment as the flat top is ideal for my small monitor.

The BBC micro is available in two versions, A and B.

The Model A costs $1595 and has 16K of RAM. The Model B costs $1995, has 32K of RAM, a serial printer interface, a parallel printer interface, a 4 channel A/D converter and an 8-bit user port.

The A can be upgraded to a B at any time.

The machine I reviewed was a Model B. It had two video outputs (UHF and mixed video) as New Zealand televisions use VHF, the UHF output was no use, but I assume/hope this will be changed on the consumer models.

Theoretically, REB output is also standard but there was just a hole in the case on my model.

To get colour graphics was a little difficult (as my monitor is only B/W). This was achieved by taking the video output to the camera input of a video tape recorder and the VHF output to the television (i.e. using the VTR as a RF modulator).

The BBC micro is a truly expandable machine. By just plugging in a few chips you can have a floppy disc controller, a speech synthesiser/ROM loader, Teletext input, networking and more.

I will deal with some of the hardware features in detail, one at a time.

- Construction: lightweight plastic case 415mm x 350mm.
- Keyboard: has normal QWERTY layout, 10 user function keys, 5 editing keys. All keys have autorepeat.
- Serial printer interface: When I first saw this socket on the back of the machine I was confused. What is RS423? It looked like a 5-pin DIN. But I discovered it could be used just like RS232 with only RTS, CTS, data-in, data-out and 0V.
- Parallel printer port: This is a Centronics standard, 7 data and

Continued on page 29

BBC

From page 4

model B basic price. These include standard fare like parallel and serial printer interfaces, three video outputs, joystick (analogue) input, cassette, RS423 and a disk socket.

Among the more unusual are an interface to the Econet networking system, a 1 Megahertz BUS and something called the TUBE. The physical ports, but not all the circuits, are provided to use these.

The Econet network is a cheap system for sharing disks and printers. It reputedly has a fairly slow transmission speed (unlike systems such as NESTAR and Omninet for instance) but is quite well field tested.

The 1 Megahertz BUS is to allow future interfacing to laboratory equipment and such, presumably via an extension chassis.

Perhaps the TUBE is the most exciting, although it is still some way from an applied reality. It will

Unusual features

allow the user to harness a second processor to the existing 6502, effectively using the 6502 just to drive the screen and other I/O devices. This second processor can be another 6502, a Z-80 (here come the CP/M software crowd) or, most excitingly, a 16 bit processor (a 16032) with a potential 16 Megabytes of RAM.

Other machines offer something like this (the Apple through Z-80 cards and piggy-back processors like 'The Mill'). The difference here is that the incorporation has been designed from the ground up.

I have only peeked inside the machine through the back grill. I know though that it contains a built-in clock and a very nice cousin to Microsoft BASIC.

The BASIC is worth a review in its own right but amongst its good features can be numbered the graphics commands and the ability to define Procedures (ah, structured programming in BASIC at last).

The text and graphics capabilities are very flexible, largely through two commands: MODE and VDU.

The first of these sets a screen mode. Among the options for this are 80 column lines (with remarkable clarity), a high-resolution option that stretches the monitor capability to its limits and various text/colour/resolution combinations with up to 8 colours (16 if you include flashing options).

The VDU command is especially useful. It lets you do things like redefine colour sets and create an active screen graphics area, protecting the rest of the screen for text.

Although the actual PLOT commands are limited in having no shape-drawing mode like the Apple they do permit the drawing of dotted lines and color infilling through a BASIC verb. Combined with the VDU command PLOT can also do things like draw things in repeatedly flashing colour which can create some nice effects.

The top resolution graphics options are VERY heavy on memory (20k), storing the screen image from the top of free RAM downwards. Only one screen is available for display at any time the manual seems to imply.

One can not do justice to reviewing the BASIC without further space while the disk statements are hard to judge without seeing a specimen drive in action.

However to pass on without mentioning the Sound capability would be wrong. The BBC microcomputer offers four 'voices' directly programmable from BASIC which can be used to produce chords or various complex sound effects. The important aspect of this is that these do not tie the machine up while the sounds are being generated so can be integrated with ongoing processing or screen display. An on-board speaker produces the sound.

This differs from the Atari, also with four voices. No speaker is required in the monitor and the user has far greater control over the type of sounds produced. A voice-synthesiser has also been developed in the U.K., allegedly replicating the vowels of newsreader Richard Baker.

To briefly summarise the BASIC is good and the machine has great potential. Any computer which quietly states 'Mistake' rather than screams 'Error No. 97' at my initial blunderings has to have something going for it. It is easy to use and the demonstration programs are excellent for familiarisation. I especially liked the Yellow Kingdom game, a sort of animated graphics HAMURABI.

What is wrong with it? At present it is low on software and there are no disk drives locally available. Nor, inevitably with a new machine, is there an established user community.

It seems likely that these situations will alter. The disk situation within a short time and the others over a longer timespan. This is, after all, the machine of the BBC computer literacy project and one of the machines attracting a government subsidy in the United Kingdom education system (not to mention adopted as a recommended model by several Australian states).

'Low on software'

Software and peripherals will duly appear, certainly with more certainty than for some of the other good hardware around. More news on this in a later issue.

It is a far more serious machine than the ZX-81's and VIC's of this world, and even than the Ataris (admittedly at a somewhat higher cost). It is much cheaper (and not that much less powerful) than many other up-market machines. In all it is very likeable and good value.

The caveat, inevitably, is that if the BBC can do this on an average budget in minimal time then what is lurking in the wings from more affluent and longer-standing competitors?

Who can say. The one certainty is that this machine offers new yardsticks for 8-bit, low budget home/educational machines. If you are in the market right now it demands attention, as much for what it can become as for what it currently is.

Pictures need not be worth a thousand swear words

By GORDON FINDLAY

One of the strangest things about the personal computer revolution is that the personal computers are much better at some things than the big mainframes. The most noticeable is that the little machines have it all over the big ones when it comes to graphics. Of course, those of us with TRS or System 80's may not be as well off as the Apple or Atari brigade, but the 80 is still no slug. In this article I want to share a few ideas with you. Many of these will be old hat to experienced programmers among you - if so, please be patient. Beginners may find some useful ideas though. Those of you who want more of a challenge skip to the end of the column.

The screen of your 80 is usually accessed by one of the many forms of the PRINT statement, or by SET and RESET for graphics. However, there are other ways of accessing the screen; some are as much as six times faster.

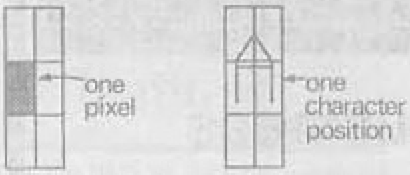


Figure 1

Before we start, remember that the screen can be divided into a 64 x 16 array of characters (letters, numbers etc.), or into a 128 x 48 array of pixels (graphics spots). A pixel (meaning "picture element") is the smallest part of the screen which can be turned on (by the SET command) or off (by the RESET command). Each character position uses the space of six pixels - as shown in figure 1.

However, the character positions may be used to set or reset graphics pixels, up to six at a time. Try this short program:

```
10 FOR I = 0 to 1023
20 PRINT@I, CHR$(128 + RND (63));
30 NEXT I
```

You will see the screen fill with various, random graphics blocks - one at each character print position.

Graphics characters

To understand where these graphics came from, we must understand that the 80 stores everything - including letters, punctuation marks, etc - in numerical form. There are a number of standard "codes" for doing this: the 80 uses the ASCII code (American Standard Code for Information Interchange). Every letter, number or symbol is stored as a number from 0 to 255. The collection of all the symbols a machine can handle is called its "character set". To see your machine's character set run this simple program:

```
10 CLS
20 FOR I = 32 TO 182 STEP 6
30 FOR J = I TO I+5
40 PRINT J'CHR$(J); " ";
50 NEXT J
60 PRINT
70 NEXT I
80 FOR I = 188 TO 191
90 PRINT I;CHR$(I);
100 NEXT I
110 END
```

(You will need to use SHIFT-@ to make the display pause.)

Notice that "A" has ASCII code 65 (in other words, A = CHR$(65), 'B' has ASCII code 66, and so on. The function CHR$(J) converts from a number (J) to the character with that code (the reverse function is ASC(A$) : ASC("C") = 67). A space is CHR$(32).

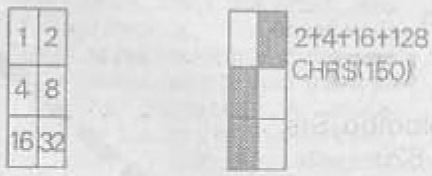


Figure 2

The characters from 128 to 191 are the graphics characters. They are made up from 6 pixels, by adding up the values of the pixels in the block which are to be turned on, and adding 128 to the result. These characters may be used by PRINTing them at the required locations.

(For completeness sake, let's add that the characters from 0 to 31 are used for special purposes - like line feed, arrow keys etc., and the codes from 191 to 255 are used for 'space compression' - to conserve storage space. You can see the space compression codes by running a little routine like this:

```
10 FOR I = 192 TO 255
20 PRINT "#" + CHR$(I) + "#"
30 NEXT
```

(The "#"s are to isolate the blanks being printed.)

Now that we've got the graphics characters, what can we do with them? First off, we can handle them in groups, by tying them together in a string, say as A$ = CHR$(129) + CHR$(176) + CHR$(167). This isn't much use, until we incorporate the arrow keys, which we can include as follows:

up arrow = CHR$(27)
down arrow = CHR$(26)
right arrow = CHR$(25)
left arrow = CHR$(24)

By including these in our strings we can build up quite complex figures.

Suppose, in a same program perhaps, we want a stick figure of a person (see figure 3). This figure can be 'decomposed' into the graphics characters shown, and the ASCII codes for them strung

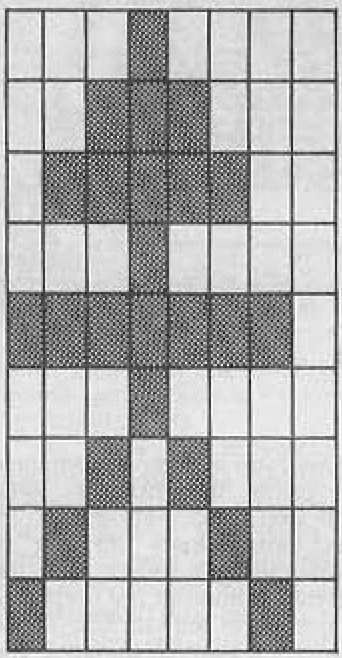
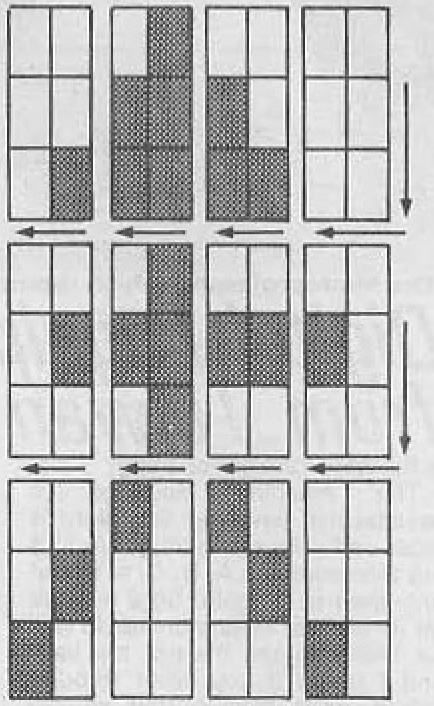


Figure 3



together. By incorporating down - and left-arrows in the appropriate places, we can get the man into a string:

```
MN$ = CHR$(160) + CHR$(190)
 + CHR$(180) + CHR$(32)
 + CHR$(26) + STRING$(4,24)
 + CHR$(140) + CHR$(174)
 + CHR$(140) + CHR$(32)
 + CHR$(26) + STRING$(4,24)
 + CHR$(152) + CHR$(129)
 + CHR$(137) + CHR$(144)
```

This little figure can be printed at any location required on the screen - and quite quickly too! For example, to place this figure at the middle of the screen, try PRINT@347,MN$.

There are some techniques which can make the long lines of CHR$(X)s shorter: 1. A string of four left arrows is given by STRING$(4,24). Similar functions can be used for other repeated characters. 2. The character codes can be listed in DATA lines, and read into the strings as required. This technique is used in the demonstration of animation below.

PRINTing a figure on the screen is all very well, but what about removing it? The simplest way is to print a block of blanks the same size, made up from the arrow characters, and CHR$(32) - which is a space. The figure of a man above may be erased by PRINTing the string

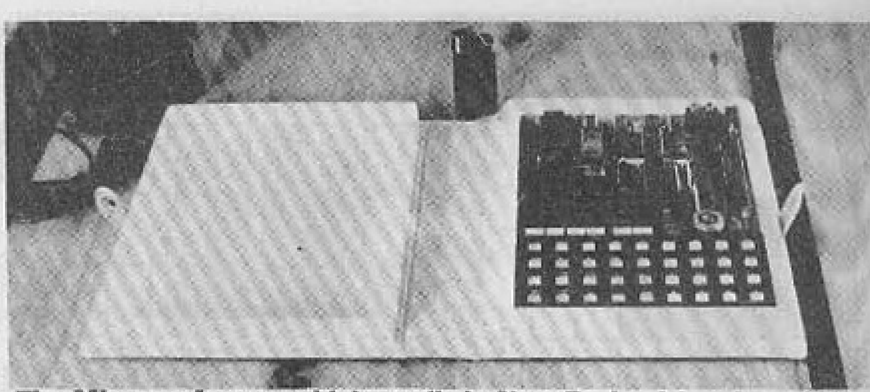
```
BL$ = STRING$(4,32) + CHR$
(26) + STRING$(4,24) + STRING$
(4,32) + CHR$(26) + STRING$(4,24
+ STRING$(4,32).
```

As a simple demonstration of these techniques, here is a program which does some (very simple) animation. Run it to see what happens!

```
10 CLS :CLEAR 500 'clean screen
   and string space.
20 SD$ = CHR$(26) 'down arrow
30 SB$ = STRING$(5,24)  '5 left
   arrows
40      BL$ = STRING$(5,32) + SD$
   + SB$ 'blanking string is 5 spaces
   long
50 CL$ = BL$ + BL$ + BL$      'and
   three rows deep
60 DIM MN$(3) '3 figures are
   constructed
70 FOR I = I TO 3 'This loop reads 3
   figures
80 FOR J = I TO 3 'This loop reads 3
   rows each figure
90 FOR K = 1 TO 5 'This loop reads
   5 characters per row
100 READ X
110 MN$(I) = MN$(I) + CHR$(X)
120 NEXT K
130 MN$(I) = MN$(I) + SD$ + SB$
   'down/back to start of row
140 NEXT J
150 NEXT I
160 PRINT@859, "STAR-JUMPS"
```

Continued on page 30

By N. E. WHITEHEAD



The Microprofessor, which retails in New Zealand for about $210.

Digital bagpipes on micro from Taiwan

Early this year I bought a Microprofessor. Made in Taiwan. About $200. Good for the soul. And definitely not for the computer enthusiast who really only wants one so that he can avoid paying zillions of dollars at the local amusement arcade. However, good value for money.

You don't get a video display for that price, nor any easy way of hooking one on. Instead there is a line of six digits in red LED, but you do get an output and input for cassette tape, 2K ROM, 2K RAM (expandable to 4K), a little speaker, and a keyboard, but non-QWERTY.

It's intended to teach you machine-level programming, and all the innards are indecently exposed for hardware additions of your fancy. Machine language? That nasty black beast? Don't worry, it's quite tameable, and I must say that following the excellent manual I found the black beast starting to perform tricks for me quiet well.

I bought it specifically to learn how to create a digital electronic organ. You see I have this crazy ambition to create an organ which sounds just like the bagpipes so I have to do a lot of work, because those crazy sassenachs won't market one. The principle is: take the waveform of the instrument you want to copy, and turn it into numbers, store it in your memory - well, it might be better to use your computer's... then write a little routine to output them to a little digital to analog converter (circuit on request - quite simple and cheap) and into your stereo.

In principle this is far, far more versatile than using a synthesiser, and I have found the information I have gained on the way through experimenting with subtle changes in the waveform ear-opening!

The machine language is hexadecimal which at first sight is most odd. The numbers go up to 9 but then continue A, B, C, to F, and only then to O again. So a number for an address in memory could well be 1AB3. Again, it's not too bad, and I guess if you went through school when they taught you to think in bases other than 10 it should be a cinch.

Because you have access at machine-language level to the machine, you can do marvellous things with the LED display. I delighted to make it display non-standard characters, and wrote a little program such that when my wife started it, it displayed a fairly recognisable I Luv U! One program they give in the manual makes a segment of LED apparently chase around the periphery of the display. With a bit of practice you should be able to do things your friends would find almost impossible on their higher level machines.

The manual gives a large number of carefully analysed and annotated programs even including a primitive organ using the built in little speaker, and an accurate clock.

It is extraordinary for me to compare this little device with the PDP-8 mini-computer my branch of DSIR bought in the late 60s. It had 4K memory a teletype terminal, and a paper tape reader and punch, but cost thirty thousand pounds. (That's right, not $).

I haven't compared it in detail, but I suspect the Microprofessor has a faster cycle time. The PDP-8 used moderate wattage - the Microprofessor runs on 500mW, and I have run it even in the bus off those nice sealed lead-acid small batteries now available - it should run for close on 4 hours I think on the set I have.

I hear that a speech synthesiser will be coming out for it soon, and also a miniBasic. Nice, but really a little beside the point. Who wants those when you can curdle your neighbours' blood with enhanced synthetic bagpipes inaccessible to users of more expensive personal computers?

Fighting the Martians

Alien Destroyer is a variation of the arcade game 'Space Invaders' for Apple computers.

The main difference is that there is only one martian in play at a time, each one faster than the last. Of course, the object of the game is to gain as many points as you can before the martian lands on you (it doesn't drop bombs).

To make like more interesting, (your only life) you can choose what level (or speed) to play at, 1 being the slowest.

In addition to the Applesoft program, there are also the shape tables to create.

Enter the following:

CALL 151 (to enter the monitor)

*IDFC: 03 00 08 00 14 00 24 00 24 25 25 2D 2E 2E 36 3F 3C 37 3F 00 95 AD 32 3F DF 3F 2E 55 12 F5 1F FF 0E 4D 05 00 49 6D 3A DF BF 6D 49 35 FF FF 37 0D 4D 0D 05 00 (return)

Don't forget the spaces!

*BSAVE DESTROYER SHAPES, A$1DFC, L52 (To return to Applesoft press 'RESET')

Before running the program for the first time, type RUN 1010. This creates a new file called DESTROYER HIGH SCORE. It is a 'Text File' and in it is stored the current high score (which at the moment contains 0).

Then press 'Y' in answer to the question "Do you want another encounter?" or CTRL-C, RUN to see the rules.

Happy Destroying!

NB: The arms of the martian actually go up and down!

By Richard Hobbis, a fifth form student at Otumoeta College, Tauranga.

Write to us

```
60  REM
70 D$ =  CHR$ (4): REM CTRL-D
80  REM GET HIGH SCORE FROM DISC
90  PRINT D$;"OPEN DESTROYER HIGH SCORE": PRINT D$;"READ DESTROYER HIGH SC
     ORE"
100  INPUT HSC
110  PRINT D$;"CLOSE DESTROYER HIGH SCORE"
120  REM GET SHAPE TABLES FROM    DISC
130  PRINT D$;"BLOAD DESTROYER SHAPES"
140  POKE 232,252: POKE 233,29
150  GOSUB 1120
160  REM MAIN PROGRAM
170  HOME
180  RESTORE
190  IF B < 1 THEN L = 1
200  POKE  - 16368,0
210  HGR
220  HCOLOR= 7
230  REM PLOT STARS
240  FOR I = 1 TO 100
250 W =  INT ( RND (1) * 279)
260 Z =  INT ( RND (1) * 138)
270  HPLOT W,Z
280  NEXT I
290  HPLOT 0,150 TO 279,150
300  REM SET VARIABLES
310 GS = 50
320 X = 4:XX = 4
330 YY = 36
340  POKE  - 16368,0
350  REM CHECK FOR KEY PRESSED
360 LAS =  PEEK ( - 16384)
370  IF LAS = 202 THEN X = X - 6
380  IF LAS = 203 THEN X = X + 6
390  IF X < 4 THEN X = 4
400  IF X > 270 THEN X = 270
410  HCOLOR= 7
```

Continued over

Picking winners

This is something I ran the other day to pick up winners for punters. Although it asks a lot of seemingly intelligent questions the clue to the whole operation lies in the variables at the end of lines 135, 145, and 160.

It runs on the Sinclair ZX81 in 1K with a little room for graphics or other comments left.

Program contributed by Paul Kinley of Christchurch

```
2   CLS
5   PRINT "MEETING CODE?"
7   INPUT A$
9   CLS
15  PRINT "HOW MANY STARTERS?"
20  INPUT X
40  LET R = INT (RND * X) + 1
50  CLS
60  PRINT "WHAT IS THE RACE NO.?"
70  INPUT Y
80  CLS
90  LET N = INT (RND * X) + 1
95  IF N = R THEN LET N = N + 1
100 CLS
110 PRINT "WHAT DISTANCE?"
115 INPUT Z
120 CLS
125 PRINT "HERE ARE MY PICKS FOR RACE,"; Y
130 PRINT
135 PRINT "1ST PICK NO.>"; R
140 PRINT
145 PRINT "2ND PICK NO.>"; N
150 PRINT
155 IF X <= 9 THEN GOTO 190
160 PRINT "3RD PICK NO.>"; D
190 PAUSE 800
200 RUN
```


Alien destroyer From page 11

```
420  ROT= 0
430  SCALE= 1
440  DRAW 1 AT X,144
450  IF T = 0 THEN  DRAW 3 AT XX,YY
460  IF T = 1 THEN  DRAW 2 AT XX,YY
470  IF LAS = 160 THEN  GOTO 490
480  GOTO 770
490  REM FIRE ROUTINE
500 FX = X + 3
510  FOR I = 20 TO 25
520  POKE 768,I: POKE 769,2
530  CALL 770
540  NEXT I
550  FOR F = 136 TO YY - 1 STEP  - 9
560  HCOLOR= 7
570  HPLOT FX,F TO FX,F + 4
580  DRAW 1 AT X,144
590  IF T = 0 THEN  DRAW 3 AT XX,YY
600  IF T = 1 THEN  DRAW 2 AT XX,YY
610  HCOLOR= 0: HPLOT FX,F TO FX,F + 4
620  DRAW 1 AT X,144
630  IF T = 0 THEN  DRAW 3 AT XX,YY:T = 1: GOTO 660
640  IF T = 1 THEN  DRAW 2 AT XX,YY:T = 0
650  REM CHECK IF MARTIAN HIT
660  IF FX >  = XX AND FX <  = XX + 8 AND F >  = YY AND F <  = YY + 4 THEN
     680
670  GOTO 730
680  IF T = 0 THEN  HCOLOR= 0: DRAW 3 AT XX,YY: SCALE= 2: HCOLOR= 7: DRAW
     3 AT XX - 4,YY: GOTO 700
690  IF T = 1 THEN  HCOLOR= 0: DRAW 2 AT XX,YY: SCALE= 2: HCOLOR= 7: DRAW
     2 AT XX - 4,YY: GOTO 710
700  IF T = 0 THEN  FOR V = 1 TO 5: NEXT V: HCOLOR= 0: DRAW 3 AT XX - 4,YY

710  IF T = 1 THEN  FOR V = 1 TO 5: NEXT V: HCOLOR= 0: DRAW 2 AT XX - 4,YY

720  GOTO 870
730 XX = XX + L
740  IF XX > 270 THEN XX = 4:YY = YY + 18:GS = GS - 5
750  NEXT F: REM END OF FIRE     ROUTINE
760  POKE  - 16368,0
770  HCOLOR= 0: DRAW 1 AT X,144
780  IF T = 0 THEN  DRAW 3 AT XX,YY:T = 1: GOTO 800
790  IF T = 1 THEN  DRAW 2 AT XX,YY:T = 0
800 XX = XX + L
810  IF XX >  = 270 THEN XX = 4:YY = YY + 18:GS = GS - 5
820  REM THE MARTIANS HAVE LANDED
830  IF YY > 132 THEN  GOSUB 1560: FOR K = 1 TO 7: READ MT: READ MD: GOSUB
     1590: NEXT K: GOTO 930
840  GOTO 360
850  REM INCREASE SCORE,
860  REM AND MARTIAN SPEED
870 SC = SC + GS:L = L + 1: HOME : VTAB 23: FOR I = 10 TO 15: POKE 768,I:
POKE 769,2: CALL 770: NEXT I
880  PRINT "     SCORE=";SC:M = M + 1: HTAB 28: VTAB 23: PRINT "LEVEL=";L
890  VTAB 22: HTAB 15: PRINT "HI-SCORE=";HSC
900  FOR I = 1 TO 10: NEXT
910  HCOLOR= 0: DRAW 3 AT XX - 4,YY - 2
920  GOTO 300
930  HOME : VTAB 22: PRINT "INVADERS SHOT DOWN ...          ";M
940  PRINT
950  PRINT "FINAL SCORE          ...          ";SC
960  FOR I = 1 TO 2000: NEXT
970  IF SC > HSC THEN HSC = SC: GOTO 1020
980  GOTO 1070
990  REM NEW HIGH SCORE
1000  REM SAVE IT
1010 D$ =  CHR$ (4)
1020  PRINT D$;"OPEN DESTROYER HIGH SCORE"
1030  PRINT D$;"OPEN DESTROYER HIGH SCORE"
1040  PRINT D$;"WRITE DESTROYER HIGH SCORE"
1050  PRINT HSC
1060  PRINT D$;"CLOSE DESTROYER HIGH SCORE"
1070  HOME : VTAB 22: PRINT "DO YOU WANT ANOTHER ENCOUNTER?";: GET A$
1080  GET A$
1090  IF A$ = "Y" THEN SC = 0:M = 0:ZZ = 69645: GOTO 1500
1100  IF A$ = "N" THEN  TEXT : HOME : END
1110  GOTO 1070
1120  REM INTRO ROUTINE AND RULES
1130  GOSUB 1550
1140  HOME : TEXT
1150  VTAB 3: HTAB 15: PRINT "ALIEN DESTROYER"
1160  HTAB 15: PRINT "===== ========"
1170  FOR I = 1 TO 50
1180 J =  INT ( RND (1) * 255 + 1)
1190  POKE 768,J: POKE 769,10: CALL 770
1200  NEXT I
```

Continued on Page 14

'Epson aims at non-computer' people

AUCKLAND REPORTER

Epson the brand name of the Shinshu Seiki company of Japan is to sell computer products outside Japan for the first time. Well known for its range of dot matrix printers Shinshu Seiki has been successfully selling microcomputers on its local market since 1978.

Its new HX-20 Portable Computer is about the size of an A4 notebook.

Microprocessor Developments, Ltd, which has been New Zealand distributors of Epson for five years, will handle the New Zealand release.

Mr Mark Bond, manager of M.D.L.'s Epson computer division in Auckland, recently spent three and a half weeks in Japan.

The new microcomputer will be available through local New Zealand dealers (providing there are no customs complications) about the middle of November.

Aiming sales at professional and personal computerists alike, a lot of attention is being given by Epson to "human interfacing" such as shape, angle, colour and feel.

"Epson of Japan aim to build computer products for non-computer people" says Mr Bond. "They also aim to offer products which first-time users can learn to fully operate within an hour."

Mr Bond said that because of the influence the hobbyist has on decision making persons in an office situation, Epson feels he/she is one of the most important people in the market place.

The Epson HX-20 will sell in New Zealand for $1599.84 (including tax).

This amount buys the HX-20 portable computer (ivory colour), with carrying case, 240V AC 50hZ power supply and battery charger, operation manual and basic reference manual, built-in printer, ink ribbon cartridge and roll paper.

Standard features of the HX-20 include:

1. The basic unit has the advantage of a full size ASCII keyboard. In addition a numerical keypad utilises part of this keyboard. Changing from typewriter keyboard to numerical keypad or reverse simply involves using the "number" function key.

- Cursor keys.
- Special functions keys.
- 10 programmable function keys.
- 16K RAM (Random Access Memory).

Continued over

From previous page

- 32K ROM (Read Only Memory)
- The liquid crystal display (LCD) shows segments 20 columns by four lines. Maximum screen area, however, is 255 columns or characters (per line) wide by 255 lines long. A new feature on the LCD is the adjustable viewing angle. It is dot addressable (any screen dot may be turned on or off); and a bit image can be easily printed using the self contained printer.
- A 24 column dot matrix ribbon printer (not thermal) which prints 42 lines per minute.
- RS 232 port, transmits up to 4800 bits per second.
- Serial port at 38K bits per second for connection to peripherals such as video monitors and disc drives.
- Reset button.
- Bar code reader input.
- Cassette port for standard cassette machine operation.
- A built-in speaker is provided for audio feed back.
- Real time clock displaying time of day/date of year.
- 40 hours on internal batteries when not using printer. Continual operation with printer eight hours.

The Epson HX20

Options:

- Internal micro cassette at approximately $270
- ROM cartridge option at $100
- 16K RAM expansion unit is priced at $290
- Bar code reader $311

Further options to be released in January include.

- Dual floppy disk drives (640K total) in one box (320K bytes per drive).
- Acoustic coupler.

Currently there are about 50 software packages available, with half for business and about a quarter each for education and hobbyists.

February will see the release of a larger Epson microcomputer in New Zealand, the QX-10 Valdocs system, designed for the average consumer.

"In a few years time" says Mr Bond, "Epson aim to have their products sitting beside fridges in home appliance stores as part of home furniture."

An in-depth product review of the new HX-20 Portable Computer will appear in a coming issue of "BITS & BYTES".

Alien destroyer

From page 12

```
1210  FOR K = 1 TO 500: NEXT K
1220 A$ = "WRITTEN BY RICHARD HOBBIS...JUNE 1982...  (30 SPACES)
     "
1230 B$ = "   (30 SPACES)                    "
1240  GOSUB 1640
1250  FOR K = 1 TO 500: NEXT K
1260  VTAB 11: HTAB 2: PRINT "DO YOU WANT TO SEE THE RULES? (Y/N)";
1270  FOR I = 1 TO 10:J =  INT ( RND (1) * 255 + 1): POKE 768,J: POKE 769,
   10: CALL 770: NEXT I
1280  GET A$
1290  IF A$ = "Y" THEN 1320
1300  IF A$ = "N" THEN  HOME : GOTO 1500: HGR : RETURN
1310  GOTO 1260
1320  HOME
1330  VTAB 3: HTAB 15: PRINT "THE RULES"
1340  HTAB 15: PRINT "=== ====="
1350  FOR I = 1 TO 10:J =  INT ( RND (1) * 255 + 1): POKE 768,J: POKE 769,
   10: CALL 770: NEXT I
1360  VTAB 7: PRINT "YOU HAVE TO SHOOT DOWN AS MANY MARTIANS  AS YOU CAN B
   EFORE THEY LAND ON YOU."
1370  PRINT
1380  PRINT " THE MARTIANS ONLY ATTACK ONE BY ONE."
1390  PRINT : PRINT "TO MOVE USE THE BUTONS."
1400  PRINT
1410  PRINT "                  'J'  FOR LEFT"
1420  PRINT "                  'K'  FOR RIGHT"
1430  PRINT : PRINT "AND TO FIRE, PRESS THE 'SPACE BAR'"
1440  VTAB 23: PRINT "PRESS ";: INVERSE : PRINT "ESC";: NORMAL : PRINT " T
   O END, ";
1450  INVERSE : PRINT "SPACE BAR";: NORMAL : PRINT " TO CONTINUE"
1460  IF  PEEK ( - 16384) = 155 THEN  HOME : END
1470  IF  PEEK ( - 16384) <  > 160 THEN 1460
1480  POKE  - 16368,0: HOME
1490  FOR I = 1 TO 20:J =  INT ( RND (1) * 255 + 1): POKE 768,J: POKE 769,
   10: CALL 770: NEXT I
1500  HOME : TEXT : VTAB 12: PRINT "WHAT LEVEL DO YOU WANT TO START AT ";:
   INPUT B
1510 L = B
1520  HOME : HGR : IF ZZ = 69645 THEN  GOTO 170
1530  RETURN
1540  END
1550  REM SET UP FOR PLAYING                 MUSIC
1560  FOR I = 770 TO 790: READ J: POKE I,J: NEXT
1570  DATA 173,48,192,136,208,5,206,1,3,240,9,202,208,245,174,0,3,76,2,3,9
   6
1580  RETURN
1590  REM PLAY MUSIC
1600  POKE 768,MT: POKE 769,MD: CALL 770: RETURN
1610  REM DATA LOCATION
1620  DATA 120,100,158,40,158,40,142,100,158,200,127,100,120,200
1630  REM  + + MOVING SIGN + +
1640 LA =  LEN (A$):LB =  LEN (B$)
1650 B$ = B$ + A$
1660  FOR I = 1 TO  LEN (B$) - LB + 1
1670  VTAB 8: HTAB 6: PRINT  MID$ (B$,I,LB): GOSUB 1740
1680  FOR D = 1 TO DY: NEXT D
1690  GOSUB 1740: POKE  - 16336, PEEK ( - 16336) =  PEEK ( - 16336): GOSUB
   1740
1700  NEXT I
1710 B$ =  MID$ (B$, LEN (B$) - LB + 1,LB)
1720  RETURN
1730  REM  + + NOISE (WAIT) LOOP + +
1740  IF  PEEK ( - 16384) > 127 THEN  POKE  - 16368,0: WAIT  - 16384,128,0
   : POKE  - 16368,0
1750  FOR K = 1 TO 10: NEXT K
1760  RETURN
```

Aid for new tax codes

This subroutine for the new tax codes is by DON RICHARDS, of Hawke's Bay. It is running within a payroll progam on his TRS80 model 111. We would be pleased to hear from other readers on their programming for the changes - the Editors.

```
1 "Tax Tables from 1st October 1982
2 REM Only "A" "G" , "SEC" & No Tax deductable
3 REM R is the record number of the employee
4 REM T$(R) is the TAX CODE
5 REM PP is the Pay Periods for one year
6 REM T(R) is the amount of the pay
7 REM TX(R)is the tax calculated
8 REM ST is the Standard Deduction (2.5% or Max $52)
9 REM PI is the Principal Income Earner Rebate
10 CLS:CLEAR 500
100 INPUT"EMPLOYEE NAME";NM$
110 INPUT"RECORD NUMBER";R
115 IF R>10 THEN 1  REM Greater Number Requires DIM Statement
120 INPUT"TAX CODE......";T$(R)
125 IF T$(R)="A" OR T$(R)="G" OR T$(R)="SEC" OR LEFT$(T$(R),1)="N" THEN 130:ELSE
 120
130 INPUT"GROSS PAY.....";T(R)
140 INPUT"NO OF  PAYS IN A YEAR";PP
150 GOSUB 1000
160 CLS:PRINTNM$:PRINT"GROSS PAY "T(R):PRINT"TAX "TX(R):PRINT"NET "T(R)-TX(R)
170 END
1000 REM   calculation of TAX TABLES
1005 IF LEFT$(T$(R),1)="N" THEN TX(R)=0:RETURN
1010 IFMID$(T$(R),1,3)="SEC" THEN TX(R)=INT((T(R)*PP*.31)/PP*100)/100:RETURN
1015 ST=INT((T(R)*PP*2.5))/100:IF ST>52 THEN ST=52
1020 TA=(T(R)*PP-ST):IF TA >6000 THEN TX=TA-6000:GOTO 1030
1025 TT=INT(TA*20-.5)/100:GOTO 1035 '20c. in Dollar
1030 TT=(TX*.31)+1200   '31c. in Dollar
1035 IF LEFT$(T$(R),1)="A"THEN PI=INT(TA*5.5+.5)/100:ELSE PI=0:GOTO1050
1040 IFPI>312 THEN PI=312
1045 IFTA>12000THEN RD=(TA-12000)*.12:PI=INT((PI-RD)*100)/100
1050 X=(TT-PI)/PP:TX(R)=INT(X*100-.5)/100
1055 IFTX(R)<.02THEN TX(R)=0
1060 RETURN
```

Post that subscription card today

THE SOFTSIDE:

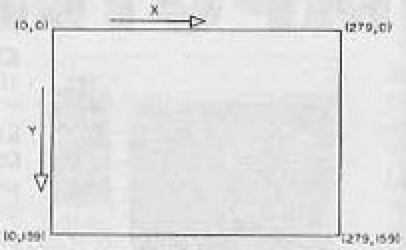
On the importance of good language

By PIP FORER

The focus in this month's article is the importance of software; especially graphics commands in BASIC. So far we have dealt with the display side of graphics, essentially WHAT the machine is able to display for you. Predominantly this has been a discussion based on hardware. We have discussed WHAT can be displayed but not HOW you might display it. In fact though, for anyone who wants to program graphics, the HOW can be the most important question. The most elegant graph in the world may not be worth the trouble if it takes several hours of nail-biting programming to get it to display correctly. What is needed is a fluent, easy language for creating images. This article looks at just what BASIC has to offer in this respect. This differs markedly on different machines. Later on we will see what other languages are around.

The character graphics that we discussed in the first issue of Bits and Bytes was never simple to use. It was true that you could draw a heart shape very easily on the early PETS just by using the appropriate key with the graphics shift held down. But any form of complex diagram or flexible display was never simple. Most machines have since come to offer vector graphics using a PLOT command and this provides a far simpler means of obtaining graphics pictures.

The fundamental idea behind vector graphics is that the screen can be divided into a rectangular grid of Cartesian co-ordinates: X units across the screen and Y units up and down it. This is like having invisible graph paper on the screen. Although the number of addressable units on each axis vary between machines (and, through software, within machines) the axis arrangement is standard. The only departure from this is that some machines number the Y axis from the top for the Y axis (the point 0, 0 is at the top left) while others number from the bottom. The latter then has its origin (0,0) at the bottom left corner which is more the way we usually draw graphs.



Working with this framework the computer usually provides a variety of facilities. Among these are line drawing, color setting, clearing the screen, drawing shapes, displaying various different pages of graphics, mixing text and graphics, and restricting drawing to only part of the screen. Various machines do some or all of these things with varying degrees of competence, in different ways and with contrasting levels of ease for the user.

Simple Things First

The first requirement of drawing graphics is to display a screen to draw on. With most machines this is a simple command in BASIC which has the effect of both clearing and displaying the graphic screen (or one of several). With some machines (like the Apple) this CLEAR offers a Henry Ford screen clear... any colour of screen so long as it is black. In others a background colour can be set for the whole screen. (If you know the Apple well enough you will know that this can also be achieved on the Apple by using some sneaky routines in the monitor and a CALL statement). The 'clear' statements can also select a particular display page if a choice is available. This may also imply the setting of different graphics resolution and capabilities.

Next we need to set the colour for any lines to be drawn. Here most machines offer a fundamental statement where each colour equates to a number. Thus on the Apple HCOLOUR = 3 sets the 'pen' to colour 3 (which is white) for any future drawing operations. As covered last month, some machines offer indirect colour addressing which allows selection of display colours from a wider choice than those actually displayable at any one time. Also some machines have colour codes which flash individual lines on and off or rhythmically alternate the colour they are displayed in. This can be useful.

Having got your blank screen and colour set you need to draw on it. Almost universally this is done with a command PLOT (or a similar word). However, a variety of PLOT species exist. The simplest PLOT is PLOT X, Y to draw a point at the screen location X, Y. PLOT X, Y TO X1, Y1 will draw a line on some machines. Some even offer PLOT X, Y TO X1, Y1 TO X2, Y2... et cetera, by which means a single statement can draw a series of joined lines, i.e. producing a square. These sorts of PLOT are pretty self explanatory. Less so are PLOTS of the form PLOT 1, X, Y, 0, 0, 0. Some machines have these sophisticated, not to say devious, PLOTs.

Sinclair group

John Mitchell, of P.O. Box 33-098, Barrington, Christchurch, is trying to form a Sinclair ZX user group in Christchurch. Those interested can also contact him by phone 385-141 (home) and 62-199 (work). Mr Mitchell is also interested in forming a national Sinclair users' club.

New Atari group

Atari Users Group inaugural meeting. Wednesday, 10 November, 7.30 p.m., David Reid Electronics, Vivian Street, Wellington. Contact, Eddie Nickless, phone 731-024.

The Compucolor II was one of the first machines I noted with this. In its case the extra numbers controlled how plotting was done and in what colour. The BBC microcomputer takes this several stages further. Apart from MOVE and DRAW commands which cater for simple line drawing it has a complex PLOT command of the form PLOT N, X, Y. On this machine the first value in a PLOT statement controls several functions. It can for instance control whether PLOT draws a line or simply plots a point and how the new plot interacts with the existing background colours. Alternatively it can instruct the machine to interpret the X and Y values as relative or absolute co-ordinates. That is to say if X is two and Y is two the PLOT command would draw either to the absolute co-ordinate (2,2) OR draw to a position on the screen which is 2X and 2Y increments higher than the last drawing action. It can also do a colour fill on a triangular area, a function performed by a special function on the Atari 400 and, for more complex outlines, by the POLY system.

Drawing shapes and mixing text

PLOT lets you draw lines. Now you CAN draw any computer graphics image by a suitable control of colour and lots of PLOT statements. It will be tedious, though. One operation that is frequently required is drawing a particular outline or shape. The character graphics machines handled this rather simply (but rather inflexibly) by having keyboard keys represent particular symbols. If you use vector graphics there are basically three approaches that you can employ in drawing frequently used symbols or shapes. The first is to use PLOT statements and set them in a subroutine or procedure. Then each time you want the shape you simply call that subroutine, passing to it information on where on the screen the shapes is to be drawn.

The second approach is to set an area of memory to values which, when inserted into the memory holding the graphics display, give a particular shape on the screen. This is used in most space games on microcomputers. The Atari has a very advanced version of this called player-missile graphics which allows several such shapes to be defined and for priority to be assigned between them so that any of the eight possible shapes will pass in front of or behind any other.

Finally you can do what the Apple does. This approach is very flexible. Simply the Apple can interpet binary values in the RAM as plotting movements (up/plot, left/no plot etcetera). Thus any shape can be encoded like this and displayed through a simple command DRAW N AT X,Y where N is the number of a shape you want to draw. (The shape drawing code, in a shape table, has to have been previously loaded into memory). These shapes can be ANY size and can be rotated and scaled (although with some lumpiness when this is done). This is a very powerful feature and perhaps the greatest trump in the Apple II's hand.

The disadvantage with the Apple has always been that text and graphics do not easily mix. To get text on the graphics screen means using one of the shape drawing approaches above, with each shape being a letter. You can get lots of very interesting fonts that way but the procedure is messy, requiring a lot of software. On the other hand some machines allow text to be written with a simple PRINT statement to any part of the graphics screen. Mixed text and graphics are offered by most machines released in New Zealand since 1980, including the Atari, BBC and Poly brands.

Further Features

If you write educational programs you quickly learn how important it is to be able to format the screen in different ways. In some cases you need a lot of small text and a little diagram. In other cases larger text overprinting some graphics. In other cases still graphics in centre screen with a margin of related text.

The nicest and most foolproof way to do this is by creating VIEWPORTS (sometimes called WINDOWS) on the screen.

Basically these allow graphics to appear in only one part of the screen, text in another. Clearing text leaves graphics unaffected and clearing graphics has no effect on text or graphics in other parts of the screen. It becomes impossible to inadvertently confuse or overwrite text and graphics.

Different BASICs offer different capabilities on this. Many machines let you drop the bottom of the

Continued on page 33

THE BUBBLESHUFFLE

This program was written by Keith Paintin when he found a need for a list of numbers in random order.

"All the existing programs ran very slowly because they did repeated checks to see that no duplicate numbers were generated," he writes.

"The idea of using a BubbleSort technique in reverse, a BubbleShuffle then occurred to me. The result a very fast and short routine which works very well on a System 80 and with line 60 modified to

60 LET N = INT (RND*X) + 1

works just as well on a Sinclair ZX81.

"I have used it as a subroutine in 'Mastermind' type games, a turn-a-round game, and in screen-printing programs. It also works to sort cards with the limit number changed."

Keith Paintin is a science lecturer at the Palmerston North Teachers' College.

```
BUBBLE-SHUFFLE

1 RANDOM
10 DIM A(9)
20 FOR I=1 TO 9
30 LET A(I)=I
40 NEXT I
50 LET X=9
60 LET N=INT(RND(X))
70 LET S=A(N)
80 LET A(N)=A(X)
90 LET A(X)=S
100 LET X=X-1
110 IF X>0 GOTO 60
120 FOR I=1 TO 9
130 PRINT A(I);" ";
140 NEXT I
```

7	4	9	1	8	5	6	3	2
7	9	5	8	3	1	6	2	4
2	9	7	1	4	8	3	6	5
8	4	6	1	2	9	3	5	7
7	1	2	9	6	8	3	5	4
2	4	8	1	7	3	5	6	9
6	7	2	9	5	3	4	1	8

By GERRIT BAHLMAN

To the uninitiated, bits and bytes can be a world of confusion. How often have you heard people talk about machine-code routines, registers, addressing modes and other sorts of jargon without having a clue what they are on about? People who are "into" their machines always want it to do things which it wasn't designed for and, somehow they seem to do it. How? What is machine code? How does it work? What is the difference between machine code and 'non' machine code?

Obviously, this article can't answer all your questions. To try would involve looking at a specific machine and sorting out how it operates. We haven't got space for that but we can describe what goes on generally and give you some idea of where it all fits in.

Fundamentally, writing programs is about a conversation between you and a machine. What you say must be very carefully phrased because the machine has a very limited number of words it can understand. The trouble is that what a machine can understand best we can only barely understand and what we understand the machine has to work at before it can understand.

A high level language is one

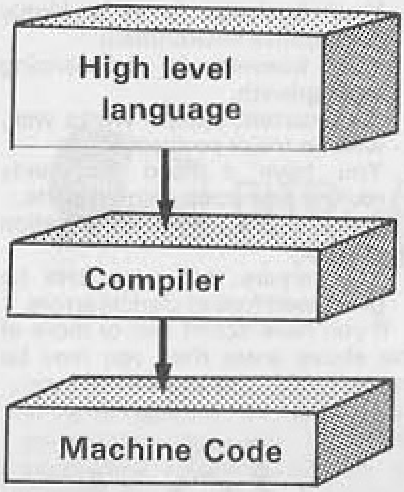
MACHINE LANGUAGE: In the belly of the beasts

which we find easy to understand and the machine has to work extremely hard to understand. The work the machine does is called a translation of a high level language to a low level language. The lowest language the machine translates to is machine code. There is a lower level still called micro code but we will leave that for now.

Common high level language are COBOL (Common Business Oriented Language), FORTRAN (Formular Translation Language), BASIC (Beginners' All-purpose Symbolic Instruction Code), ALGOL (Algorithmic Oriented Language), PL/1, Pascal, Forth, LISP (List Processing Language), RPG (Report Generating Language) to mention just a few. There are many more, SNOBOL, APL, etc, each of which is designed to do a particular job. They are designed for use by people and have to be translated to machine language.

Compilers

The translation process is called "compiling" and is done by a program which has already been translated into machine code. Obviously, each machine must have a special 'compiler' for each language and worse than that each machine type will have to have a special compiler for the same language. Have a look at the diagram



When you write a program in a high level language it must pass through each of the stages shown. There are two sorts of compilers. Some of them will work on the whole of the high level language and produce all the machine code at once. The other type, called an interpreter, takes only a small part at a time, produces the machine code for that and then lets the machine obey the instructions at one. BASIC is usually an interpreted language, i.e. the compiler takes a line of a BASIC program, converts it to machine code and then runs it. When that line is finished the compiler converts another line of the program. Why don't all high level languages work that way? Can you think of any disadvantages in "interpreting" a high level language?

I have been a little inaccurate in my description of compilers in that I have given the impression that they are responsible for "running" or "executing" the machine code once they have produced it. Only interpretive compilers can do this. Most compilers just produce the machine code and other machine code programs take that produced code and link it and load it into the machine. Obviously, I have to explain what is meant by loading and linking.

Loader

Machine code is nothing more than simple instructions that the machine can understand. To be understood those instructions have to be placed into the machine's memory. It can understand the instructions only if they are placed in its memory. What is more, they have to be placed in a strict order and position. Interpretive compilers load the machine code into the special places in memory and switch the machine on so that it knows when to start running. A "loader" is a special program which is part of an interpretive compiler but is a separate program for non interpreters. A loader is a program which places machine code or instructions into memory and prepares them for execution.

The loader is a much simpler program than a compiler. It has a much simpler job to do. Can you think of a reason why compilers and loaders are separate programs?

There are two main reasons. First, the memory of the computer is limited. When it has the compiler program inside it there is not a lot of room left. But, there still has to be enough room to allow the compiler to do its job. The loader cannot do its job until the compiler has finished so it is not needed at that time.

The second reason is tied up with the fact that each machine must have a different compiler because its memory is different. The loader "knows" where everything is. It "knows" where to put programs that are to run. So, it would be simpler if compilers were similar as possible from machine to machine, for the same high level language, and the loaders were the programs that did most of the adapting to the different machines. This would save a great deal of adapting work.

Linker

This is another program which does a particular job. Again, in interpretive compilers the linker program is inside the compiler program. The linker brings together bits and pieces from the high level language before the loader puts the machine code into the computer's memory. Specifically, the high-level language will often want to use special program sections that have already been compiled into machine code and they have to fit neatly into the total list of machine instructions before they can be loaded.

I haven't yet talked about how the machine code programs such as compilers, linkers and loaders are written. I haven't discussed machine language in terms of how it works or what it looks like.

Next time I will try to take you just a step further into the depths of the computer's innards and we will look at some of those terms mentioned.

Selecting a micro for the small business

By John Vargo

This is the first of a three-part series intended to provide business and individuals with a set of criteria for purchasing and implementing a small-business computer. Recommendations will include highlighting the major steps in this selection and implementation process as well as pointing out some of the major pitfalls.

Article one in the series, entitled "Considerations", covers advantages and disadvantages of an in-house small business computer. Also included is a discussion on the uses of a small-business computer and whether or not your business is ready for one.

The second article, "Selection", will cover the actual selection process. Topics to be covered will include: the real cost of your computer; obtaining the right size system for your needs; how to get the right combination of hardware, software, and service. Suggestions for evaluating the financial benefits to the company will also be given.

Converting your current manual system to a computerised system will be considered in the final article of this series, "Implementation". Minimising the risk of conversion, providing for internal control, and timing the conversion so as to disrupt business as little as possible will also be a major focus of the discussion in article three.

Considerations:

Is your business ready for a computer..

If you are the owner, or manager of a small business and you do not yet own a computer, chances are you are considering purchasing one. With the falling price of hardware and rising cost of labour the incentive to automate your accounting and control systems can be quite strong.

Before you go down to the local computer store you must arm yourself with an understanding of your needs and a reasonable grasp of computer jargon. The first practical suggestion is for you to begin reading the computer magazines regularly to orient yourself to the language of the field.

Let us take a look at some of the indications that you may be readv for automation, as well as some of the indications that you are not.

Read through the following check-list and tick off those items which apply to you:

- Your accounting system has many routine transactions that follow a planned schedule.
- You have a need for information that is not currently provided by your system.
- Your business is in a highly competitive environment
- Your business is experiencing rapid growth.
- Your current system works well, with no major problems
- You have a need for many routine time consuming reports.
- You require quick information retrieval
- You require that your data be processed free of clerical errors.

If you have ticked two or more of the above areas then you may be ready to seriously consider acquisition of a small business computer.

Now, look at the next list and tick off those items which apply:

- You are very keen to purchase a computer but your managers and employees will require much convincing.
- Your transactions consist of mainly non-repetitive events.
- Your business is in a mature industry with little growth potential.
- Your current accounting system is giving you fits, and you think a computer could get you out of the current mess.
- Your current manual system is more than satisfactory, and you have no particular need for specialized information.

If any of the above situations apply to you, then computerizing your business at this point might be an unwise decision.

The above check-lists points out some of the features of a computer system that you must take into consideration before you make such a critical decision. If you decide to automate unwisely you may be in worse condition than when you started.

"First time computer users often find that the first effect is not to solve problems but rather to rearrange them in a different pattern."[1]

A successful computer installation requires co-operative and interested staff, a disciplined and smoothly running system, a measure of routine transactions, and a need for up-to-date information.

Selection of a small-business computer requires a disciplined approach and is not to be confused with the more relaxed way you might select a personal/home computer.

Alternatives

If you conclude that you are in a position to automate your system (or some portion of it), what alternatives do you have?

Owning your own business microcomputer is the alternative we will look most closely at, but you should consider the other alternatives as well. Simply upgrading your current manual system may give the benefits you are looking for. A computer service bureau, or a timesharing system may service your needs better than your own in-house system.

Consideration must also be given to the possibility that a micro based system will be inadequate for your current and near future needs. You may choose a mini-computer or larger machine instead.

> "...beware of a 'strait-jacket' system that requires you to fit your business to the software's mould."

What are some of the advantages and disadvantages compared to the above alternatives? Advantages would include:

- Quicker availability of reports (re service bureaus etc)
- Less expensive than minis, and potentially less than service bureaus.
- In a multi-user network environment, back up equipment is part of the system (e.g. if you have a network of 10 micro's and one goes down, you still have nine working).
- Easy access to a broad spectrum of inexpensive business software.
- Greater encouragement to use hardware for other applications.

Some of the disadvantages would include:

- You are responsible for keeping the system up and running, providing maintenance for the software and the hardware.
- If the hardware becomes obsolete or inadequate for your needs, you are responsible for replacement.
- Limited disc space
- Limited directly addressable memory.

Obviously, the advantages must be weighed against the disadvantages for your particular situation. As this series progresses we will be looking at specific remedies for some of these disadvantages.

Potential Applications

Word processing can be one of the most useful applications for a small business computer. This is especially true if you have standard correspondence with customers and vendors where the same formats are used repeatedly.

Another obvious candidate for computerization is the accounting system. You may consider an integrated accounting package which includes: General Ledger, Debtors, Creditors, Payroll and Inventory Control (or some combination of these elements).

We will speak more about specific considerations for the choice of software in the second article, but a word of caution is useful here. If considering an off the shelf accounting package, beware of a "strait-jacket" system that requires you to fit your business to the software's mould. This kind of prepackaged system tends to be somewhat restrictive, however the substantial cost savings of "off the shelf" software may more than offset the inconvenience. In other situations, the restrictions may be so severe that the system becomes useless for your needs.

Additional applications may include: order entry, production control, budgeting, and cash flow planning. Many of these applications may be implemented through the use of generalized software for financial planning and modelling. This software falls into two categories: the electronic spreadsheet (Visicalc type), or the more formal modelling languages.

The above applications, if implemented in your business, could yield a number of benefits including: less overtime, more interesting work for employees, fewer errors, more timely reports, improvement in cash flow, increased sales, and reduced costs.

Here is a practical example:

The Z.A.B. Company Ltd (a hypothetical company) has annual sales of $2,200,000 and the value of goods in stock at any one time averages $350,000. As a result of purchasing a small business computer and implementing an inventory control system, the average inventory was reduced to $315,000. By eliminating some slow moving items and improving order points, this reduction of $35,000 in stock meant the company would reduce its borrowing by an equal amount. Since the loans were at 20 per cent interest the ultimate result was a saving on interest expense of $7000 in the first year.

This hypothetical situation points out the potential for savings through better asset control.

Does it still sound like your business is ready and could benefit from a small business computer? If so, then let us take a look at the first step in the selection process.

The feasibility study

At this point we need to do a "feasibility study" of your company. The feasibility study is intended to specifically discover if a computer would suit your business, and, if so, which particular areas would benefit from computerization. The study also provides the basis for the design of your system components (hardware, software, people, data, procedures).

During the selection process it is important to involve employees who will be the ultimate users of the proposed system. If they are not interested, unco-operative, or alienated for whatever reason, the best system in the world will not perform satisfactorily.

Call a meeting of all department managers, and explain to them your reasons for considering computerisation. Tell them your concerns, business needs, and plans for future growth. Then find out what concerns they might have, alleviating their fears right from the beginning. Openly communicate your concern for their welfare and desire for their help in the evaluation process. You now have a team.

The team's first task is to review each department's operations. They should list the different files, number of records on each file, number of accounts, reports required and frequency of these reports, personnel involved in the department and their responsibilities, quantity and nature of transactions. The manager should also define any needs or problems he is currently having regarding the information system.

Once this information is compiled it is possible to determine which

Continued on page 26

Advice from Accountants *-look at applications first*

By PAUL CROOKS

Don't look at microcomputers first - look at the application you see them fulfilling in the office.

That is the advice from Mr Denys Jones, Christchurch computer manager for chartered accountants Lawrence, Anderson and Buddle, to businesses intent on buying a microcomputer.

He and other staff have recently completed an extensive search for a microcomputer by purchasing an ICL Personal Computer.

They had two applications in mind during their search:

- In-house processing, some clients and particularly farmers wanted their accounts done on the spot in their home or office.
- Cash flow analysis, some clients and again particularly farmers required this.

Lawrence, Anderson and Buddle's (LAB) IBM 34 mainframe couldn't fulfill either of these requirements. On-line terminals were too expensive (the cost of a microcomputer, disk drives, printer and so on is about the same price as a screen for the IBM) and the software needed was either not available or too expensive.

So to fulfill its requirements, LAB began looking for a microcomputer that was:

- Portable
- Compatible with the IBM 34 diskettes or had a communications facility.
- Used an industry standard programming language
- Used an industry standard operating system and thus had access to a large amount of software including standard packages such as debtors, wages etc, that clients were likely to require.

After looking carefully at company backgrounds, LAB selected 10 companies on "the basis they wouldn't be gone tomorrow" and invited them to make proposals.

Nevertheless LAB was disappointed with some microcomputer sales staff.

"We got in touch with some micro people and the response was nil," said Mr Jones, "it could reflect we were moving down market and not making such a big purchase".

He also thought some salespeople would be very confusing for a new person looking for a microcomputer.

"They would baffle them with computereze".

After evaluating the responses, LAB decided the ICL Personal Computer fulfilled all the points they were looking for and commissioned Systems Software and Instrumentation of Christchurch to install it.

The ICL is portable, it could be linked while in the office to the IBM 34 via a cable, it runs as standard the popular operating system CP/M and programming language BASIC which means a large amount of software is already available for it.

LAB looked closely at the IBM personal computer but it's not being offered by IBM itself in New Zealand yet.

"As well it offers only 136 Kbytes of storage per floppy disk compared to the 250Kbytes per floppy on the ICL," explained Mr Jones, "we needed the extra capacity for larger clients."

The price of the ICL was also attractive at around $11,000 for the basic unit (with 64Kbytes of RAM memory), VDU, twin 5.25 inch floppy, disk drives and a dot matrix printer.

Mr Jones has found the ICL systems documentation to be no worse than most.

But the major reason for the purchase of the ICL was software; the quality, quantity and cost of the software available to run on it.

The CP/M operating system gives LAB access to a large amount of software including the popular financial planning packages such as VisiCalc, T/Maker and CalcStar which perform a number of statistical calculations and help prepare budgets and forecasts.

Continued on page 26

LAWRENCE ANDERSON AND BUDDLE'S MICROCOMPUTER EVALUATION

1. *Product Name*
2. *Manufacturer*
3. *New Zealand Distributors and basis of distribution*
4. *Delivery performance*
5. *Components*
   a. *processor type and speed*
   b. *bus S100 or other*
   c. *Standard memory size, maximum memory and unit of expansion*
   d. *Static or dynamic memory, blank select or not*
   e. *direct screen addressing or RS232 interface*
   f. *number of terminal interfaces*
   g. *printer interface*
   h. *spare slots on bus*
   i. *disk options capacity and average access time of each disk configuration.*
   j. *disk back up (streaming tape)*
6. *Operating systems*
   a. *systems available*
   b. *can it be CP/M compatible*
7. *Software*
   a. *any special software available*
   b. *ordinary software available*
   c. *who writes software*
8. *Costs of configurations*
9. *Costs of maintenance*
10. *Net working capability*
11. *Will it act as a terminal to a remote mainframe?*
12. *What is installed user base in New Zealand?*
13. *Documentation available and quality of documentation*
14. *Warranty period*
15. *General comments on componentry and expandibility.*

ICL is getting personal

ICL is already a well-known leader in large computers for large projects. Now ICL introduce a new range of multi-function PERSONAL Computers specifically designed for the small to medium sized business and the first time user.

Priced competitively, the range of ICL PERSONAL business systems are available for immediate delivery. See them demonstrated at ICL TRADER POINT outlets.

FULL ASSURANCE

The new compact computers from ICL offer assurance of back-up and technical assistance that only a large international computer company is able to provide. Right now . . . and in the future. An ICL assurance that is vital for the secure growth of your business.

BUSINESS EXPANSION

Every business needs to grow. The versatile ICL PERSONAL Business Computer has full capacity to expand with your needs, from increased data storage enhancements to multiple screens and printers. No need to change systems every few years — ICL PERSONAL Computers will grow as your business expands. Unlike many micro computers, ICL PERSONAL Computers can handle word processing concurrently with other business applications. Large data storage — minimum of 64 Kbyte memory store, flexi disks and 5 Mbyte hard disk options, allowing full flexibility for your expanding business systems.

SOFTWARE — READY, WILLING AND ABLE

A wide range of software packages eliminates need for expensive, specifically written programs. Packages are available to suit your business whether it's invoicing, stock control, general accounting or word processing. Simply explain your needs to the TRADER POINT specialists. They'll assist you in selecting a program that gets you straight down to business.

FITS IN . . . FAST

Learning how to use an ICL PERSONAL is quick and easy. No additional staff needed. No special installation. Just plug in anywhere and it fits your office routine immediately. Your staff will welcome the ICL PERSONAL for its speed, accuracy and the variety of tasks it can handle. A small business computer PLUS word processing options AND information processing.

ICL PERSONAL COMPUTERS

think fast for you!

To: ICL TRADER POINT
P.O. Box 394, Wellington

Please send me further information on ICL Personal Computers.

Name: ....

Business Name/Title: ....

Address: ....

....

....

Phone: ....

SEE FOR YOURSELF

Send this coupon now for more information. Check out how easy they are to use and how ICL PERSONAL Computers can help answer your business problems . . . in an instant.

ICL

trader point

This article discusses the way you can design and write a small business system once you have carried out the investigation outlined last month in the first part of this series.

By KERRY MARSHALL

Keep it Simple

This must be the guiding principle behind your early attempts to produce computer systems. Simplicity amounts to providing the software to handle 80-90 per cent of the processing and leaving the user to handle the exceptions manually.

Start with the OUTPUTS

Prepare samples of each output and describe how it is to be produced. This will indicate what stored data and input information is needed to accomplish the output.

For example a library system to record periodical circulation might have as its primary outputs:

* listing of all periodicals in alphabetical sequence.
* listing of all people showing which periodicals they receive.
* circulation lists showing the periodicals and the people receiving each one.

From this we can see that two main files are needed with a cross-reference method:

* Periodicals file - would have data about each periodical:
  title,
  topics,
  subscription date,
  publisher.
* Readers file - persons name
  location or number
  periodicals required.

Inputs and Processes

Now consider how each file will be handled. Both will need a program to add records, delete old records, and to amend data.

We will also need a program to produce each report, and the only problem is how to produce the cross-referenced list of periodicals and readers.

Experience

Here there is no substitute for experience. If you do not have access to suitable manuals, or a user group, then you just start from scratch and experiment.

There are numerous ways you could set up the files but the best way will depend on the disc operating system and the ability of the programmer.

Developers of software have given new meaning to the word experiment, with many a successful software package evolving from the trial and error programming of an inexperienced user. It is beyond the scope of this article to look into the details of file design, but perhaps future articles by specialists in different machines might give some advice.

Key points:

* Avoid excessive waste space (even if you have no constraint now, you may need the space tomorrow).
* Make sure security copies or back-up procedures are well thought out.
* Avoid storing extraneous data.

The key to a good system: Keep it SIMPLE

Program specifications

By now you will have broken the processing required into distinct steps. Each step will become a program. In some cases, steps may be combined, but only where the functions are compatible.

For example in our periodicals system, the periodicals file maintenance and listings routine may be combined into the same program. The advantage then is that after maintaining the file a listing can be done without changing programs (the program would of course display the functions as options). The disadvantage is that stepping from the system menu to the listing requires the loading of the now large maintenance program.

For each of the steps you should prepare a program specification which will have:

* An introduction - program objectives or purpose outline of how it works.
* Program name - a unique identifier.
* File Definition - full details of the file including name, location, block and record size, layouts of all the records.
* Report layout - showing how data is to be printed, and or displayed on the screen.
* Processing - a description of all steps to be followed including - error handling, control procedures, input checking, calculations required.

Writing the program

There are many text books and magazine articles which describe the process of writing a program and the principles of the various design methodologies. I will limit myself to the key items affecting the microcomputer user.

Inputs: this is where many systems fall down. A good input routine will:

* Only allow valid keys to be pressed (e.g. 0 to 9 when numerics are required).
* Prevent input past the maximum length allowed.
* Check the input for accuracy and ask for re-input if it is wrong.
* Not make assumptions (e.g. if No or Yes is required, don't accept any other input if it's not Yes).
* Display a default value wherever possible as a prompt and use that value if the return key is pressed without any other entry being made.
* Require all input to be completed by pressing return (even one-key entries!).

Screen

This is the interface with the user and should be designed with the user's help.

Use meaningful error messages, and indicate progress during periods of apparent inactivity. There is nothing more alarming to a user when he or she presses return - the screen goes blank and the machine sits there seemingly dormant.

When updating a file or doing a search, I show the current record number and key so that the user can

Continued on page 31

Office micro also linked to bureau

Annette, of the staff of Auckland accountant, Lex Davidson, using the Osborne computer.

By CATHY ARROW

Two accountants in the Auckland area were attracted to the Osborne 1 computer by overseas articles; subsequent investigation led to purchase of machines which have now been in use for several months.

Lex Davidson is a chartered accountant, with the Takapuna branch of Kenton Cox and Co. He first noticed the Osborne computer advertised in overseas computer magazines. He was attracted by its standard software, portability and RS 232 port.

He checked with Dr Jim Baltaxe of the Osborne user group in Palmerston North, to see if he found the system as good as advertised. Subsequently Mr Davidson purchased an Osborne for himself to use as an educational tool and has since written several games as exercises.

He has developed a program to account for time spent on clients behalf. All clients are listed on the machine; referenced by the first few letters of their name. Details of time spent are entered and time sheets compiled and printed off. Thus, Mr Davidson has a complete history of time spent and full details available for billing.

Currently Mr Davidson is the only person in his firm with a microcomputer. He is interested in programming to a degree, so he can talk more easily to people who program for his clients, but is not particularly interested in machine language.

The Osborne's small screen has clear definition and presents no problem for personal use.

"The keyboard feels good and as a non-typist the numerical keypad was important" says Mr Davidson. "A typist may tend to use the numbers on the main keyboard, but if you have used a calculator or are a processer operator, then the numerical keyboard is excellent."

Of the software provided with the Osborne Mr Davidson has found the Supercalc spreadsheet program the most useful so far. His applications for Supercalc have been cash flow projections, sales mix analysis and break-even analysis.

The office operates on a time share basis with mainframes at CBL and OBM. He was able to use the microlink communications program to convert the Osborne to a terminal and thus easily transfer files to and from the CBL mainframe computer.

As he has no letter-quality printer the Wordstar word processor is of interest only at this stage. He has however, plugged in and run LA120 and MX100 printers, driven by Wordstar with ease.

Meanwhile, in central Auckland a firm of chartered accountants who purchased their Osborne in July have been using it in conjunction with a Wang 2200VP computer which uses Hartley software.

This arrangement is proving extremely useful and time saving in audit and cash flow applications now that the initial setting up is done.

In audit work, statistical sampling and analysis can take a long time. Selection and sorting is much quicker by computer than by manual means. The computer is unequalled in speed and efficiency for selecting and sorting a random number of samples, as a basis of items to test in an audit situation.

In this office also, Supercalc is used for cash flow forecasts. The Wordstar word processor is utilised for confirmation letters, using a client data file to provide the information. If no reply is received within two weeks a second letter is generated. Form letters are easily personalised. They are currently setting up a standard audit program on Wordstar, tailored to print the individual client's circumstance. The Osborne's ability to operate on the spot in the clients office, eliminates the need to travel back to one's office to use a computer system, thus saving considerable time. The ease with which a microcomputer may be taken home is a distinct advantage.

From page 22

Advice from Accountants

"These programs cost a few hundred dollars whereas a financial package for the IBM 34 costs $13,000, explained Mr Jones.

"As well the IBM package is not as user-friendly which is important if it is going to be used by computer unintelligent end users at all."

The cash flow and farm accounts software LAB needed was provided in one $600 package, the Financial Recording System, developed by the Kellogg Farm Management Unit at Lincoln College.

With the right software available, inexpensively, it was decided to purchase the ICL and "migrate the data rather than the software." That is the ICL does the analysis then the results are communicated to the IBM which stores the data, sorts it and produces final accounts at the end of the financial year.

Mr Jones concedes another reason for the purchase of the ICL Personal Computer was that otherwise clients, wanting the advantages of a microcomputer might go out and buy of their own.

He sees some clients still doing this in the future and while LAB would rather see clients using its machines it is now advising clients on what microcomputers to buy.

This is because they will still use LAB for some analysis and accounts and it "gives us protection from clients coming in with disks from unfamiliar machines."

To help in this all LAB's New Zealand offices participated to compile a survey on most microcomputers available in New Zealand. They compared them on a number of points (see box).

Mr Jones thinks the survey is as good as any but because the market is so big and changeable, time will continue to be devoted to upgrade the survey.

From page 21

departments will benefit from the use of the computer and which will not. You may even conclude that there is insufficient need to justify purchasing a computer at this point in time.

Based on the data gathered, you will have available the information necessary for specifying the needs to be filled by the system that is ultimately selected. The information should be sufficiently detailed so that you can determine the number of work stations, disc storage size, and other system components in the system specification portion of the project.

Below is an example of specifications for a typical accounting application:

Accounts Receivable

Number of credit customers?	420
Number of credit invoices prepared per month	800
Number of cash receipts on account per month?	380
Number of credit/debit memos issued each month?	20
Maximum invoices open per customer at one time?	approximately 20 but average 3 to 5 invoices.
How often are statements produced for customers?	Monthly
How often is an ageing of debtors prepared?	Monthly but would like weekly
What other reports are required from the system?	Currently none but would like a weekly analysis of cash flow.
Does the quantity of transaction in the system have a peak load or other cylical feature?	Invoicing currently rises to a peak at month end and then tapers off for a week or so.
What is the estimated growth in the quantity of transactions over the next five years?	It is expected that sales transactions will double over the next five years.
Number of staff in department?	Three clerks and one manager
What needs or problems do you currently have in the system?	The department is often overworked at month end with peak load invoicing and sending out statements. Also need weekly reports mentioned above.
Current Debtors collection period	82 days.

When the details have been gathered for all departments or areas of responsibility, hold another meeting of managers and users to discuss the results. Evaluate the data gathered in the feasibility study, and come to a consensus on whether to computerise.

If the decision is to go ahead with automation, then the decision must also be made as to which applications within which departments justify the expenditure. It should also be established which additional applications will be brought on line over the five year time frame of the initial system implementation. This will establish the boundaries of the size of system required.

At this point, most of the groundwork needed to prepare a detailed list of your system specification and requirements has been done. Based on this detailed list, the process of selecting the right system, can proceed.

Now you can have a nice long talk with the salesman at the computer store, knowing you won't be swamped with his sales pitch. ptich.

Next month we will look at the process of selecting the right system for your needs.

1. "Computerisation: still a calculated risk", by Peter Issac, in *The Accountants Journal*, June 1982.

(John Vargo is a lecturer in accountancy at the University of Canterbury).

By R. HUNT

Sail your way through this

Minefield is a game that runs on the IK Sinclair ZX81. The object of this game is to score as many points as possible. To do this you must pilot your ship through a minefield.

The game begins after the background and mines have been plotted.

There is a bonus score if you manage to hit the "X".

You are able to move by using "B" for down and "H" for up.

If you collide with a mine the game concludes by displaying a score which is proportional to the amount of moves you have survived.

For those who are skilled enough to navigate the first field, another will be created, this will continue until you have been destroyed.

If you are able to score more than 3000, consider yourself an expert. If you wish you could use other graphic symbols, I have used simple ones.

The variables are:
K-score; W-horizontal position of ship; Q-vertical position of ship; A-B-positions of mines; R-PEEK result.

The loops are: 10-30 sets up variables; 40-60 draws background; 70-110 draws mines; 120 bonus object; 130-220 is the main program loop.

Program listing

```
10 LET K=PI-PI
20 LET W=PI-PI
30 LET Q=4
40 FOR N=0 TO 7
50 PRINT AT N,0;" 27 OF CHR$ 128"
60 NEXT N
70 FOR G=I TO 38
80 LET A=INT (RND*22)+4
90 LET B=INT (RND*8)
100 PRINT AT B,A;" "
110 NEXT G
120 PRINT AT RND*7, RND*26; "☒"
130 PRINT AT Q,W;
140 LET R=PEEK (PEEK 16398+256*PEEK 16399)
150 PRINT "▣"
160 IF R=0 THEN PRINT "CRASH ";K;O
170 IF R=189 THEN LET K=K+100
180 LET W=W+1
190 LET Q=Q+(INKEY$="B" AND Q<7)-(INKEY$="H")
200 IF W=27 THEN GOTO 20
210 LET K=K+10
220 GOTO 130
```

Note: ▣ and ☒ are inverse characters

By STEVEN DARNOLD

Running PET programs on your VIC

Converting programs from one type of computer to run on a different type is usually quite difficult. However, the VIC and the PET are so similar that most PET programs in BASIC will run on the VIC without too much difficulty. However, VIC owners who want to run PET programs will probably have to make several alterations.

Because the PET has nearly twice as many columns as the VIC, some PRINT statements will be broken in the middle of words and wrap 'round to the next line. Usually this is easily fixed by breaking up the PRINT statements. However, on some occasions, it's necessary that one PET line end up as one VIC line. For example a PRINTed chess board can not have the tail-end of each row wrapping 'round to the next line. In this case it is necessary to compress the PET line (usually by removing spaces). Sometimes the wrap 'round is so bad that the VIC user has difficulty figuring out what is supposed to be happening. In this case it may be helpful to discard the excess characters on each line until the pattern is clearer.

Other commands affected by the VIC's 22-column screen are POS, TAB, and SPC. POS is most often used to test for the end of a line, in which case it will need to be reduced by about 18. TAB, SPC, and comma are used to format printing on the screen. Their values will have to be reduced to fit the VIC screen, and some of the printing may have to be abbreviated.

Usually an INPUT in a PET program will occur on the same line as its prompt. However, the VIC does not like wrapping 'round on an INPUT, so it is usually best to start INPUTS on a fresh line. Therefore a statement like INPUT"prompt";A$ should be changed to PRINT"prompt":INPUT A$. Additionally, many PET programs have three spaces and three cursor-lefts at the end of a prompt in order to avoid dropping out of the program on a null input. Since the VIC does not stop on a null input, such extra characters at the end of prompts can be left out.

The further a PET program is from mainstream BASIC, the more difficult it will be to convert. Since PEEK, POKE, and WAIT address specific memory locations, it may be awkward to find the equivalent locations in the VIC. SYS and USR are even worse because they jump to machine language routines.

The accompanying chart lists the most frequently used addresses for PEEK, POKE, and WAIT and gives suggestions for conversions to the VIC. PEEKs and POKEs to the screen, however, are not so simply dealt with. Because the PET screen is laid out in rows of 40, vertical lines are POKEd by taking steps of 40; and diagonal lines, by steps of 39 or 41. On the VIC, however, vertical lines require steps of 22 and diagonal lines, steps of 21 or 23. VIC users will have to change screen offsets of 39, 40, 41, −39, −40 and −41. 32768 is the upper left corner of the PET screen, 32807 is the upper right, 33728 is the lower left, and 33767 is the lower right. VIC owners doing conversions are advised to rule off a 25 × 40 grid and number the squares accordingly.

Usually PET programs containing a SYS or USR are too difficult to convert. Even a knowledge of 6502 machine language may not be enough. However, sometimes the SYS or USR can simply be left out. For example, a popular SYS routine flashes the screen. Dropping such a SYS does no harm.

The PET and the VIC are very similar. They can read each other's cassettes and disks, they use the same graphics set, they control the cursor the same way, and they have roughly the same architecture. Therefore, with a little bit of effort, VIC users will be able to run most of the many PET programs.

Common Pet addresses for PEEK, POKE, and WAIT

40-53	pointers: indentical to VIC 43-56
141-143	clock: VIC 160-162
144	POKE 144 is used to disable the stop key: it can be left out
151	PEEK 151 is used to see which key is held down (64 = none)--values returned are different than those of PET
152	WAIT 152,1 waits until the shift key is pressed: VIC 653
158	number of characters in keyboard buffer: VIC 198
166	similar to PET 151: VIC 203
167	POKE 167,0 turns on cursor: VIC 204
623-632	Keyboard buffer: VIC 631-640
634-1017	cassette buffers (machine coded is often POKEd into this area—easy conversion unlikely): VIC 828-1019
32768-33767	screen memory (see article)
59464	POKE 59464 plays note (1 = hi, 255 = 1o)
59466	POKE 59466 sets octave for note (15 = 10, 51 = mid, 85 = hi)
59467	POKE 59467,16 turns on sound (0 = off)
59468	POKE 59468,14 sets upper/lower case (12 = graphics)

BBC

From page 5

2 handshake connections via a 26-pin plug.

- A/D converter: A 4-channel Analogue/Digital converter is made available via a 15-pin D socket. Apart from the analogue inpu channels there is a light-pen connection.
- TUBE: This is a special interface for a second processor. Which will be your choice of either another 6502 or a Z80 running CP/M both with 60K of user memory and in the pipeline is a 16-bit National 16032 machine capable of addressing 16MB of memory.
- Graphics: A range of eight modes from very high resolution in two colours to medium resolution in 16 colours. (Eight colours and eight flashing options) to just text. Modes use between 20K and 1K of memory.
- User port: This is an eight-bit parallel I/O port via a 20-pin plug. For PET users it is an unbuffered B side of a 6522PIA.
- Sound: This comes standard with models A and B. And is implemented via a small internal speaker. It can play three-note chords and make a variety of noises.
- Cassette interface: This is one of the best features of the BBC micro. Why? Because it can handle cheap cassette recorders and the volume and speed fluctuations that go with them. Speed is 1200 bits per second and 300 bps standard cuts format for compatability.
- Speech synthesiser: This option has two purposes, to give the computer a voice and to control special ROM packs that can be fitted. It has a built in vocabulary of 150 words and in addition you can program your own words.

Software

Memory in the BBC micro is divided into two 32K parts. The top 32K is used for ROMs and memory mapped I/O.

This contains a powerful Operating System, an Assembler, and an excellent BASIC. The only problem with having all this software in ROM is that it only leaves you with 32K of RAM, which depending on the graphics option leaves you between 27.5K and 8.5K for program space. I felt this was the worst feature of the machine.

> **'Exciting... for what it could become...'**

BASIC

The BASIC in the BBC micro is not Microsoft but a product of Acorn. It is just about standard Microsoft but with a number of improvements.

The best of these areProcedures and Functions. BBC BASIC does have the normal GOSUB, but in addition it has a DEF statement which you use to define functions (DEF FN name (parameter list)) and procedures (DEF PROC name (parameter list)). These are called by FN name (parameters) and PROC name (parameters) and for the first time a micro has structured BASIC.

Other goodies are REPEAT.... UNTIL loops (Pascal fans take note) and the assembler which can be mixed almost indiscriminantly with BASIC.

Sound as mentioned in the hardware section has four channels (three notes, one noise) these are controlled by a SOUND statement.

There is also an ENVELOPE statement, which can vary the amplitude and/or the pitch of a sound while it is playing. This lets you get on with the program while a complex sound is being produced.

Graphics is easily controllable from BASIC, with the PLOT statement being the graphics workhorse. It can plot a point, a dotted line, a solid line and even a solid triangle. The graphics on the BBC micro is far too complicated for me to do justice to it (it needs a separate article), but it is fun, powerful, and quite easy to use.

Editor

The screen editor is extremely easy to use. There are 2 cursors which separate when one of the arrow keys is pressed; they are the read cursor and the write cursor. The write cursor behaves normally and will accept text from the keyboard, but the read cursor can be moved anywhere on the screen and then text can be copied from the read cursor to the write cursor by pressing the COPY key.

Conclusion

Apart from a lack of memory (which can be remedied by a second processor) there are no major faults with this machine. It is probably the most exciting micro I have seen to date, more for what it could become than for what it is now.

Let's hear from you

"Bits & Bytes" wants publishable letters from readers. Please keep them within 150 words, and do not use a pen name. Any longer contributions will be considered for publication as an article. Technical questions, beefs about how home-computer users are treated by the bureaucracy, bricks and bouquets... all will be welcome. The editors.

From page 9

```
170 FOR I=1 TO 12 'do 12 jumps
180 PRINT@413,MN$(2) 'print first
  position
185 FOR XX=1 TO 90: NEXT XX
  'delay
200 PRINT@413,MN$(3)   'print
  second position
205 FOR XX=1 TO 90: NEXT XX
  'delay
210 NEXT I
220 PRINT@413,MN$(1)   'print
  resting position
230 PRINT@859, "PHEW!!";CHR$
  (30) clears to the end of the line
  being printed on
250 DATA 160, 190, 180, 32, 32,
  140, 174, 140, 132, 32, 152, 129
  137, 144, 32 REM data for first
  figure
260 DATA 136, 176, 190, 180, 152,
  32, 130, 174, 134, 32, 128, 152,
129, 137, 144 REM data for second
  figure
270 DATA 32, 160, 190, 180, 32,
32, 160, 174, 164, 32, 136, 161, 149,
181, 137:REM data for third figure
280 END
```

Experiment with the program:try changing the length of the delay loops, or eliminating them entirely to see how fast string graphics really can be!

This technique can be elaborated much more, for example by packing the graphics characters into string by POKE commands. But the basic idea is the same. Maybe we can look at string packing at another time. Those of you who have seen ANDROID NIM, DANCING DEMON, or DUEL-N-DROIDS will be aware of how good this method can be.

There are one or two traps - be careful to note where the cursor finishes after each PRINT statement especially. In the program above the cursor was placed at the bottom-left corner of each figure. Notice that I have used semicolons in most of the print statements to avoid scrolling problems.

You should be able to find a chart with all the graphics characters in your user manual - if not, send me a stamped addressed envelope,c/- the magazine, with a note saying what you want, and I will post you one - but no SAE means no chart!

A note for experienced programmers

Over the last month or so I have been avidly studying a very interesting and important book; "Basic Faster and Better", by Lewis Rosenfelder. This is one of the TRS-80 Mysteries" series, published by IJG Books.

It is packed with tricks and techniques for making your programs faster, better and easier to write. The techniques vary widely; they include methods for including machine-code routines in BASIC without reserving memory and an enormous number of subroutines and functions

Just to get the flavour, here are two useful functions which Rosenfelder mentions, just in passing:

```
10 DEF FN H2$(A%) = MID$("01
23456789ABCDEF", INT (A%/16)
  +1,1)+MID$("0123456789ABC
  DEF", A%-INT (A%/16)*16+1,
  1)
20 DEF FN H4$(A2%) = FN H2$
  (ASC(MID$(MKI$(A2%),2))) +
  FN H2$(ASC(MK1$(A2%)))
```

The first function, H2$(A%), converts an integer between 0 and 255 to Hex, while the second handles the conversion from -32768 to 32767.

The book has two faults. At $38.95 it is rather expensive (but all books are these days), and it is rather hard to read the program listings - they are all compressed. It is definitely not a book for the beginner, and you must have disk BASIC. And don't show it to any of the structured programming fanatics - they might burst a blood vessel!

I got my copy from MPS, 940a Colombo Street, Christchurch.

My slip!

Thanks to the many people who have bothered to write to me as a result of this column. Yes, there was an arithmetical error in the sound routine given in the first issue, which arose by my making an error in subtraction! The embarrassment this causes a maths teacher can be imagined. The program for 16K machines tries to POKE more numbers than are given as data. The easiest cure is to change 32767 (in line 40) to 32766.

Take 100 lines Gordon. I must use a calculator and not count on my fingers!

LETTERS

The Poly, etc

Dear Sirs,

It was with some distress that I noted N. M. Moir's letter in your last issue and feel that some points need to be clarified.

1. The in-service course in question was organised privately. I am sure the course committee concerned did what they felt was appropriate, in view of the limited number of machines available and the need to heighten the awareness of teachers with little computer experience. It is not the policy of Polycorp NZ, Ltd, to ban from courses or demonstrations, computer enthusiasts, and maths teachers. Rather the reverse.
2. Because Polycorp is more than interested in selling its wares, two letters were sent, last term, to every school in the country who had thus far, little comtact with the Poly system. One was sent to the principal and the other, surprise, was sent to the H.O.D. mathematics. The final paragraph invited any interested schools to contact Polycorp and arrange a demonstration.
3. The Poly has been demonstrated in small centres, such as Geraldine, Lumsden, Palmerston, Te Kao and Pleasant Point, primarily as a result of letters sent last erm.

If N. R. Moir likes to contact Polycorp, we will gladly arrange to visit his school and demonstrate the Poly system.

Yours sincerely,
Jan McKenzie,
Education Services Manager
Polycorp NZ, Ltd.

A 'beeper' to cut keyboard errors

Audible feedback for keyboards

This circuit may be used with any matrix encoded keyboard using up to 8 lines per side to detect a key closure. As shown it is very suitable for the ZX81 and is a great reducer of keyboard errors. Logic is negative, i.e. Or represents a keyboard enclosure.

With the ZX81 the inputs to the 14068B are taken from the five-way keyboard ribbon connector on the main ZX81 circuit board. The original built on "Veroboard" is mounted under the keyboard of the ZX81 beside the power supply regulator. The "beeper" unit may be mounted in any convenient position. Five volts can be taken from the output of the on-board regulator.

Both TTL level and "AC" "beepers" can be used, circuit 1(b) generating a square wave of approximately 4kHz for the AC variety. This section is not needed for a TTL level type. Ic2 a and b are used as a "Pulse Stretcher" to convert keyboard pulses into a 20m sec pulse for a suitable tone length.

This hardware improvement for the ZX81 is by CLIFFORD WRIGHT.

The values of R2/C2 are set to give a 4kHz square wave. Some "beepers" may give better results at other frequencies in the 1 kHz to 6kHz range.

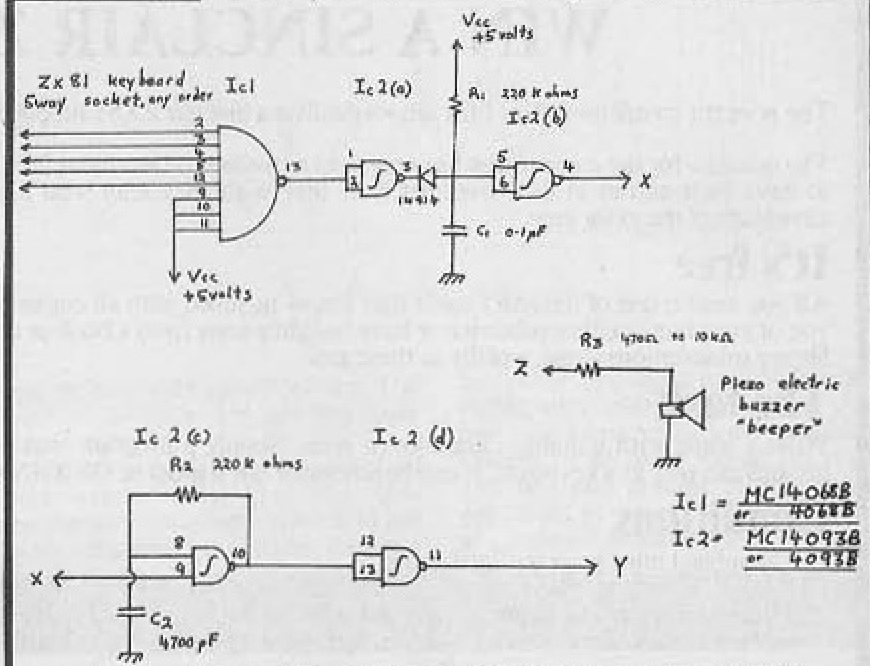
Volume is set by the value of the series resistor R3. Typical values range between 470 ohms and 12 k ohms depending on the transducer used.

It is important to ensure that the "beeper" is not of the type which requires a lot of drive power. Some small units on the New Zealand market need as much as 50 to 100 mA to operate and are not suitable.

Parts list

Ic1 1 only 14068B Integrated circuits
Ic2 1 only 4093B
R1/R2 2 only 220 k ohm ¼ watt 5% resistors
R3 1 only 470 ohm/12k ohm ¼ watt 5%
1 only 0.1 F 50 volt mylar film capacitor
1 only 0.0047 F 50 volt mylar film capacitor (4700 pF)
1 only audible alarm unit ("beeper") types PkB8 - 484AO, or PkB9 - 3A0, or similar

The total cost should not exceed $10 - $12.



For DC type "beeper" connect Z to X, for AC type, connect Z to Y.

BUSINESS

from page 24

see at any time that the machine is doing something and how far it has got to go.

Program design

* Plan your program first - set out the steps it will do and prepare a list of variables to be used. Use groups of names for common purposes e.g. C for counts, T for totals. Try to be consistent between programs so that in time you can easily look through a program and see what is going on.

* Include remarks in your program - the manuals will tell you that it will run faster without spaces or remarks but the small saving in time will disappear if you later have to look through a crammed program.

* Set out the code in blocks - e.g. one block for all the input, one for output, and so on. It again makes the program more legible in the future.

* Put frequently used blocks at the front of your program. Put those used only once or twice at the end. Many BASIC's search from the front of the program when a GOTO or GOSUB is encountered. If the routine is at the end there are that many more steps required each time.

Summary

the writing of programs is one of the more creative aspects of computers, but is also the most expensive. Good planning in this stage of the system will speed up the process and make the finished article a quality product that will stand the test of time. The key is to make it simple.

Next month we will look at getting the programs tested, setting up the system and implementing it. Also a look at the weak link - documentation.

SCHOOL COMPETITION

WIN A SINCLAIR ZX81!

The prize for competition No. 1 for school pupils is a Sinclair ZX81 supplied by David Reid Electronics.

The deadline for the competition has now been extended to December 20. Our apologies to those who rushed to have their entries in by November 5. If they wish they may send a subsequent, revised entry to take advantage of the extra time.

It's free

All you need is one of the entry cards that is now included with all copies of the magazine. This ensures that you or your family either subscribe or have bought a copy from a book or computer store (we can't survive on library subscriptions alone, worthy as these are).

The task

Write a game with minimum graphics or none. Supply a program that will generate a game one or more persons can play at a keyboard. It can be any game but it must be ORIGINAL!

Conditions

1. The subject must be as outlined above.

2. The program must be either entirely in BASIC or largely in BASIC. If any machine code is used, it must be well documented so that it is quickly evident to the judges what the code is doing.

3. A print-out containing a listing and a run on a continuous piece of paper is required. If the run can be performed adequately only on screen, then supply a written description of what happens in the run.
(Note: If you don't have access to a printer try contacting one of the micro clubs listed in "BITS & BYTES" for help. As a last resort send us a tape, at your own risk).

4. The entrant must send with the print-out a sheet of paper containing the entrant's:
•Full name • Address • School • Age • Form
• Name of program • Machine it's for • Memory it takes • Entry card (see above).

Deadline for entries

Entries for this first competition must be received by "BITS & BYTES" by Monday, December 20. Please send entries, the sooner the better, to:

Competition No. 1
"BITS & BYTES"
Box 827
Christchurch

What the judges will be looking for:

The outside judge and one from "BITS & BYTES" will select finalists largely from print-outs and the final group will be given runs. However, depending on time and number of entries it may be possible to run all entries. The judges will take strong account of comprehensibility of programs. Ease of understanding may well be more important than speed and elegance. Use plenty of REM statements, make it tidy, and structured. The judges will be looking at it just as you look at a program in a book.

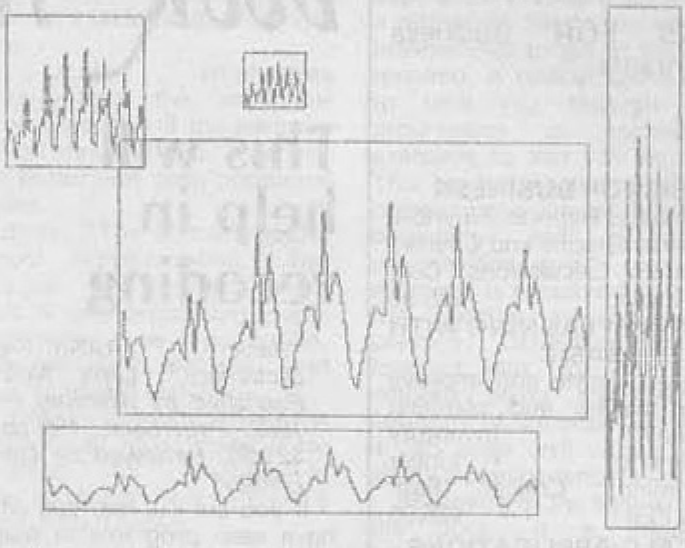
GET THAT ENTRY IN AS SOON AS POSSIBLE: More competitions coming!

From page 17

graphics screen to display text there: valuable but inflexible. With others a great deal of programming can achieve similar results. Only in the most recent releases is this facility well developed. The BBC microcomputer for instance allows it through its VDU command. The new 16-bit Corvus Concept also has a BASIC command for this which allows several windows for text and/or graphics to be operated on simultaneously.

In summary, the graphics ability of a machine in the hands of an average user is governed by how easily he can get results. If he is working with BASIC what he can achieve will be governed by the capabilities of the BASIC he is using, but each BASIC has greatly different graphics capability built into it. Assessing those capabilities is a quite daunting task requiring technical judgement and a clear idea of just what the programmer will want, at some time in the future, to achieve. For the computer buyer the advice is bother with software as well as hardware specifications. The built-in BASICs are getting ever more powerful.

A typical screen using viewports. Each graph displays the same information but has been scaled to fit the viewport.



Naturally BASIC cannot meet all your needs in the long run. One alternative increasingly open is to get extra software. You can for instance buy other languages such as Pascal, Forth, or LOGO. Alternatively you can buy some specifically graphics utilities to do extra things for you. The last two articles in this series will look at other languages and available software utilities.

POLY

Progeni takes control

As micro users will have read in the daily press, the Development Finance Corporation has announced that it has transferred its majority shareholding in Polycorp New Zealand, Ltd, to the software and systems, company, Progeni, of Lower Hutt.

Progeni had been a minority partner in the Poly venture, but now will be the majority partner, while the D.F.C. retains a 25 per cent shareholding.

"A major objective behind the establishment of Polycorp was for the company to handle the introduction of the POLY system to New Zealand schools," said Mr Murray Smith, assistant general manager of the D.F.C., of the change.

"Unfortunately, continuing delays in any decision on computers in education have affected the confidence which schools have in POLY, and raised some uncertainty about the future of this system.

"We believe New Zealand needs a national policy for the introduction of computer-assisted learning to ensure that the country maintains its reputation in the international league as a provider of quality education."

Mr Smith said the transfer was in line with the D.F.C.'s policy of transferring its equity investments in the private sector as early as possible in the development of any project.

"The D.F.C. has no doubts about Progeni's commitment to Poly and its determination to continue the work already done," he added.

Progeni was set up in 1968, the first software company in New Zealand, said Mr P. W. Harpham, the managing director of the firm.

Mr Harpham said that Polycorp had a clear understanding that Government would support the local system, whose development it had encouraged from the outset, by introducing the system and its purpose: designed software throughout secondary schools.

"Despite repeated assurances that Poly would be adopted because of its advanced specifications and proven in-school suitability, there still has yet to be any move from Government in this direction."

Mr R. J. Greenbank the new general manager of Polycorp, says a new marketing policy has been adopted. Markets other than domestic education will be developed.

"We must look to areas such as training in industry, word processing, and other applications in commerce. Most importantly, we must pursue international markets."

The drive for overseas markets has begun already and the firm is very encouraged by the interest shown by Australian educationists, Mr Greenbank says.

book reviews

This will help in recoding

"Basic BASIC-English Dictionary." Larry Noonan. Published by dilithium Press, 1982. Paperback, 150 pages. $21.95. Reviewed by Gordon Findlay.

If you are like me, you will often have seen programs in magazines which you would like to use, but which were written in some dialect of BASIC which as far as your machine is concerned may as well be Sumerian. How many really great programs have you seen with those strange graphics characters for the PET? What about peculiar commands like PRINT USING? With this book you can at least work out what they do, which gives you some chance of being able to recode the programs for your machine.

This is a dictionary, presenting an alphabetical listing of thousands of BASIC statements, operators, commands and special keys, used by Apple, PET, and TRS-80 computers. With its help you can translate from one of these dialects of BASIC to your own system. Of course, it will only tell you what the statement does in the original - it cannot make up for hardware differences, such as screen formats, graphics resolution, and memory size.

Each entry of a keyword, abbreviation, command or special key has an explanation of its meaning, and a description of its use in the Applesoft, Apple Integer, Pet, TRS-80 level 1 and TRS-80 level 2 BASICs.

There are a number of errors, which is only to be expected in a book with this amount of detail. Apple programmers may be surprised to learn that they have the PRINT AT statement available, and so will TRS-80 owners who find that POKE 32763, 1 to 255 will control the speed of a listing.

There are also a number of appendices. Appendix A lists ASCII codes for the three machines, but sometimes fails to make it clear which characters are relevant to each. Appendix B lists the basic facts about graphics for each machine - providing layout sheets for Apple and TRS-80, and a complete listing of the graphics characters for TRS-80 and PET. The printing quality in this appendix is a little poor - it is hard to distinguish some of the PET characters clearly. Appendix D provides full lists of reserved words for the three machines, and appendix E is a fairly elementary description of Boolean operators, which fails to make it clear that the numerical values for TRUE and FALSE are not the same in all machines.

Layout is adequate, although one or two passages become a little confusing. When using the dictionary, be sure to check whether the entry refers to a command, statement or special key - for example CLEAR could be any one of these!

It is tempting to compare this book with Dr David Lien's "BASIC Handbook". Lien's book gives examples from a wider range of computers, and helps you with the problem of recording by giving simple subroutines, but is more expensive, and does not include as much information on graphics, special keys, and so on.

This book is not perfect, but will be useful to programmers trying to cope with programs which need "translation". It is not cheap - nearly 15c a page - but what book is nowadays ("Bits & Bytes" is the only bargain in computing!). Used with caution, I can recommend it.

Correction

In a review in the September issue of "Bits & Bytes" the price for "Inventory Management for Small Computers", by Chuck Atkinson, was printed as $5.95.

The retail price is in fact $25.95.

In the classroom

"Microcomputers in the Classroom", by Alan Maddison. Hodder & Stoughton. 189 pages. N.Z. price $16.95. Reviewed by R. B. Clarke, senior master, Waitakere College.

This book is designed for teachers. No previous knowledge of computers is necessary. It could be considered a computer awareness book for any teacher who wants to know about this type of resource and use it in the classroom. No programming skills are taught. It assumes pre-prepared programs are available.

The reader is brought to the stage of being able to consider the following areas:

(a): Factors that could affect the decision whether to use a program.

(b): Information that should be available to a potential user of a program: how to assess the quality of a program; how to assess whether a program will improve the education of a class; how to decide if a program is suitable for educational administration; whether a program will run on the available computer; whether the accompanying documentation is adequate.

Part one, "Introduction to the Microcomputer," has chapters on computer hardware, computer software, communication with the user, and computer output.

The author in the preface states his aim to be comprehensive and points out that whole books have been written on each chapter. I found these chapters an easy-to-read summary.

A reader without a computer background may find the computer jargon a little overwhelming. Each new word and concept is written in italics as it is introduced. Teachers of computer awareness would find many of the descriptions used of value in the classroom.

Part two, "The Microcomputer in the Classroom," has chapters on introducing the computer to the classroom, the range of uses of computers, computer aided instruction, the classroom aid, randomness, models and simulations, discovery and investigative learning, and the computer and teaching materials.

Suggestions are made as to how computers could be used in the classroom in a wide range of school subjects.

The author emphasises throughout that the computer should be used only if the program is of a high standard and "teaches" in some better way than traditional techniques.

Part three, "The Microcomputer in School Administration," has chapters on the introduction to principles of data processing, pupil-based systems, class organisation, computer-managed learning, and general information handling.

I am most enthusiastic about the book and highly recommend it for consideration for anyone in computer education. It is certainly a must for our staff library.

For a grounding in Pascal

"Introduction to Pascal including UCSD Pascal," by Rodney Zaks, Sybex Inc. 420 pages. Price in N.Z. $26.50. Reviewed by Gerrit Bahlman.

Since the original definition of Pascal by Niklaus Wirth in 1970 the popularity of the language has exploded. The original conception of the langauge was for use as an education tool. Its two most significant features, program structuring facilities and data structuring facilities, quickly gave it much wider application. Because of its use of an intermediate p code phase of compilation an additional bonus became apparent. Microprocessors were able to cope with adaption to p code, which was relatively low level and thus they were capable of being programmed in a high level language. P code made Pascal readily transportable. The primitive subset of the language was clear and machine dependent extensions were a relatively simple matter.

These factors have combined to promote the language beyond the original parameters conceived by Wirth. This book provides tutorial approach to the language. It clearly distinguishes between Standard Pascal and the extensions provided by UCSD Pascal. The usual limitation of a tutorial text is that while suitable to a beginner the experienced Pascal user, who wants a reference, has to plough through irrelevencies to get at the specifics required. A tutorial text is designed to take you through a linear progression of material with exercises to test you on the way. This book does this. Each chapter presents the simple ideas at the beginning and develops the complexities quietly. A series of exercises is presented at the end of the chapter which encourages practice. Guidelines are clearly mapped out for the beginner without being patronising. The structure of the book was such that it can cope well with a range of ability and experience levels.

Returning to the tutorial nature of the book, it is imporant to emphasise, that while designed to teach the language, it is sufficiently well organised to make it a worthwhile reference once the rudiments of the language have been mastered.

An extensive appendix system is provided with a solid index and clear table of contents. A surprising number of programming examples are provided both in the text of the chapters and in a separate answer section - some 60 pages. The examples chosen are not all trivial and in themselves identify a number of useful techniques available in the language.

This book gives a thorough grounding in the language. Obviously there are areas which it fails to develop as fully as I might like. Recursion is covered but the examples are too trivial to underline the power of this facet of the language. Nevertheless, this failing is common to Pascal texts and this book at least provides more than simply the factorial example. The only facet of the text which could have been developed more fully was the area of program development. There is a need to develop an appreciation of structured programming as a design tool. It could, however, be argued that the book was written to teach the Pascal language and not to teach programming and that must carry weight.

In summary, an excellent tutorial text which is sufficiently well documented to make it a continuing asset to the Pascal programmer.

beginners

Basic BASIC No. 2

By GORDON FINDLAY

First, a word to those who think this series is too slow. You are well catered for already in books and magazines! This series is intended for those who are just starting to come to grips with programming - perhaps on their own computer, or perhaps in fleeting moments on somebody elses. It is deliberately at the lowest common denominator of BASIC - as machine independent as possible - and I am being very careful to avoid tricks and devious programming, in the hope that good habits can be fostered!

With these provisos, let's get on. The last exciting instalment left you with a program which printed some message over and over again, never stopping. Of course, you can stop the program from the keyboard, by pressing BREAK, RESET, RUN/STOP, or whatever. But it would be nice to program the computer to stop after the message had been printed a number of times.

Here I run into a problem: I don't know what sort of a computer you are using, it could be anything from a ZX80 to an IBM 370. So I don't know how many lines of text your screen can display. RIGHT NOW go and look that up if you don't already know it. Also look up the number of characters (letters and digits) you can fit on a line. Go on, do it now!

No matter how elementary the program you were left with last month, it does demonstrate the first of what I rather grandiosely call my Fundamental Rules:

FUNDAMENTAL RULE 1: Programs work from top to bottom, unless told otherwise. The "unless told otherwise" clause refers to GOTO statements just now, but will include other statements later.

To make our program stop, we need to count the number of times the message has been printed and instruct the computer to stop when the count reaches, say the number of lines on the screen. To count, we need to introduce a VARIABLE.

Variables

At a crude level of approximation, your beloved computer contains a whole lot of "pigeon holes". Never mind if the correct phrase is bytes, words, bits or whatever - you needn't worry (for now) what a "pigeon hole" is, or how big it is.

Each pigeonhole is big enough to put a number in. And once a number is in a pigeonhole, it will stay there until it is removed (or the power fails!).

For us to use the numbers in the pigeonholes we have to be able to refer to them. Each pigeonhole has a place for a label with its name. Let's call one of our pigeonholes C.

We can get at the number in the pigeonhole just by using its name - for example,

```
10 PRINT C
```

would tell us what was in there (and it would still be there after we had looked at it). At present we haven't put anything in any of the pigeonholes, so we would probably get the value 0 out - try it and see!

The names that we put on pigeonholes are called IDENTIFIERS, and they must conform to some rules. It would obviously be most confusing to have a pigeonhole called PRINT or GOTO. Again, the rules for identifiers vary a little from machine to machine - just as American English and New Zealand English differ.

RIGHT NOW go and find out exactly what the rules are for your machine. Go on, NOW.

Right, you're back. You will have certainly found that the names must start with a letter. Probably they can continue with a letter or a digit (0-9). But from there on, the rules differ. As an example, here are the TRS-80 rules, which are also the same for the Apple:

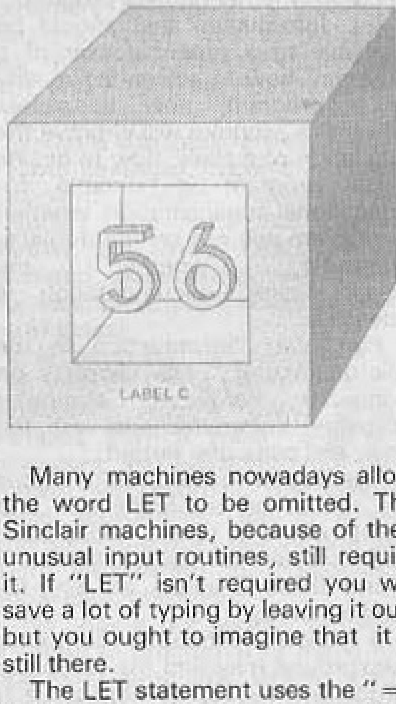
The name must start with a letter, and contain only letters and digits (and possibly a dollar sign at the end for a special purpose). Only the first two characters of the name count. The name cannot contain any of the words which are reserved for BASIC instructions.

Some examples: BITS is O.K. So is BYTES. But GORDON isn't - the word ON means something special to the machine. Since only the first two letters count, CHRISTMAS and CHRYSANTHEMUM refer to the same pigeonhole - so save yourself some typing!

O.k., we can label the pigeonholes. We even know the rules for our own machines. How can we get numbers stored in them? In the original BASIC the instruction used was LET, like this:

```
10 LET C = 56
```

This statement would put the number 56 in the pigeonhole labelled C. If we hadn't used the name C before in the program, the computer would automatically find an empty pigeonhole and label it C for us before putting 56 in it.



Many machines nowadays allow the word LET to be omitted. The Sinclair machines, because of their unusual input routines, still require it. If "LET" isn't required you will save a lot of typing by leaving it out, but you ought to imagine that it is still there.

The LET statement uses the "=" sign. The choice of "=" is a little bit misleading. Look at these statements:

```
10 A = 10
20 B = 30
30 C = A + B
```

The effect is to put the value 10 in a pigeonhole labelled "A", the value 30 in B, and to take the two values found in A and B, add them together, and put the result in C. This means that C has the value 40. If the program continues

```
40 A = 65
```

The C doesn't change! C still has the value 40, and mathematically, C = A + B is no longer true. The best thing is to think of the equals sign as a storage instruction. Technically it is called an "assignment operator", and is sometimes read "becomes" or "sets".

In assignment statements you can use the arithmetic operators we met last time:

+ for addition
- for multiplication
* for multiplication
/ for division

and ^ for powers (x^2 means x-squared).

The normal rules apply about the order operations are done in - multiplication and division before addition and subtraction. Brackets may be used to decide the order things are done in:

```
10 C = (10 + 2) / (2*3)
```

gives C the value 2.

Now it's time for another of the fundamental rules:

FUNDAMENTAL RULE 2: Everything you calculate must be stored, by doing the calculation in an assignment statement.

Like all rules, this one has an exception - occasionally you might do a calculation in a PRINT statement.

Now, how can we count the number of times a message is printed on the screen? Obviously, before we start the count is zero. Let's store the count in a pigeonhole labelled "C". Every time the message is printed, the count goes up - by one. This can be accommodated by the statement:

```
(LET) C = C + 1
```

All those with mathematical backgrounds will cringe! How can C be equal to C + 1? But remember, the equals sign doesn't mean that! The statement above means "work out the value of C + 1, and store that value in the pigeonhole labelled C'. The effect then is to replace the value in C by a new value, which is one greater.

Other examples of this sort of thing:

W = W + 4 increases W by 4
V=V*2 doubles the value of V
X = X/2 + 1 halves the value of X, then adds one to the result.

Our program is beginning to take shape: start count off at zero, and every time we print the message, increase it by one: something like this:

```
10 C = 0
20 PRINT "ONCE A JOLLY SWAGMAN"
30 PRINT "CAMPED BY A BILL-ABONG"
40 C = C + 1
50 GOTO 20
```

(This time, just for a change, I've printed a two-line message.)

Now let's make it stop. Since the pigeonhole C has the count in it, we can interrogate C using another important statement: IF.

```
50 IF C < 10 THEN GOTO 20
```

The IF statement is just about self-explanatory: if the value found in C is less than 10, go back to line 20 (and carry on printing). IF statements can generally use several kinds of relations: > (greater than) < (less than) >= (greater than or equal to) <= (less than or equal to) = (equal to) and <> (not equal to). Notice that some of the symbols are made up from two keys - because of the limited nature of the keyboard.

One important thing - what happens if the "question" part of the IF is not true - C is not less than 10? The program 'falls through' to the next numbered statement. In this case, we don't want anything in particular to happen, just to stop the program. We can use STOP or END for this.

```
10 C = 0
20 PRINT "ONCE A JOLLY
   SWAGMAN"
30 PRINT "CAMPED BY A BILL-
   ABONG"
40 C = C + 1
50 IF C < 10 THEN GOTO 20
60 END
```

An abbreviation of the IF statement is often met with: either the GOTO, or the THEN or both may sometimes be omitted. Check to see if your machine allows this. I usually prefer to leave them in to make programs more readable.

With the statements we have discussed so far, PRINT, GOTO, assignments and IF-THEN, it is possible to do quite a lot of simple programming. To try yourself out, here is a program which produces some output on the screen. Read it, and try to sort out what! Then try the program on a machine to check. Try some similar programs - and don't be afraid to experiment.

```
10 S = 1
20 C = 0
30 PRINT "*"
40 C = C + 1
50 IF C < S THEN GOTO 30
60 S = S + 2
70 IF S < 10 THEN GOTO 20
80 S = S - 2
90 C = 0
100 PRINT '*'
110 C = C + 1
120 IF C < S THEN GOTO 100
130 IF S > 3 THEN GOTO 80
140 END
```

Be careful! Make sure you sort out exactly how many things happen at each stage.

Next month we will investigate input and output (I/O) and how to handle words as well as numbers.

An ABC of computer complexities

Algorithm: A list of instructions for carrying out some process step by step.

Applications program: A program written to carry out a specific job, for example an accounting or word processing program.

BASIC: An acronym for "Beginners' All-purpose Symbolic Instruction Code." The most widely-used, and easiest to learn, high level programming language (a language with English-like instructions) for microcomputers.

Binary: The system of counting in 1's and 0's used by all digital computers. The 1's and 0's are represented in the computer by electrical pulses, either on or off.

Bit: Binary digit. Each bit represents a character in a binary number, that is either a 1 or 0. The number 2 equals 10 in binary and is two bits.

Boot: To load the operating system into the computer from a disk or tape. Usually one of the first steps in preparing the computer for use.

Buffer: An area of memory used for temporary storage while transferring data to or from a peripheral such as a printer or a disk drive.

Bug: An error in a program. The term allegedly arises from an insect that caused a malfunction in a pioneer computer.

Byte: Eight bits. A letter or number is usually represented in a computer by a series of eight bits called a byte and the computer handles these as one unit or "word".

Character: Letters, numbers, symbols and punctuation marks each of which has a specific meaning in programming languages.

Chip: Common term for an integrated circuit etched on a tiny piece of silicon. A number of integrated circuits are used in computers.

Computer language: Any group of letters, numbers, symbols and punctuation marks that enable a user to instruct or communicate with a computer. See also Programming languages ano Machine language.

Courseware: Name for computer programs used in teaching applications.

CP/M: A disk operating system available for microcomputers using a particular microprocessor (that is the 8080 and Z80 based microcomputers such as the TRS 80 and System 80). See also Disk Operating Systems.

Cursor: A mark on a video that indicates where the next character will be shown.

Disk: A flat, circular magnetic surface on which the computer can store and retrieve data and programs. A flexible or Floppy disk, is a single 8 inch or 5¼ inch disk of flexible plastic enclosed in an envelope. A hard disk is actually an assembly of several discs of hard plastic material, mounted one above another on the same spindle. The Hard disk holds much more information - up to hundreds of millions of bytes - while floppy disks typically hold between 140,000 and three million bytes.

Disk drive: The mechanical device which rotates the disk and positions the read/write head so information can be retrieved or sent to the disk by the computer.

Diskette: Another name for a 5¼ inch floppy disk.

Disk Operating System: A set of programs that operate and control one or more disk drives. See CP/M for one example. Other examples are TRSDOS (on TRS 80) and DOS 3.3 (for Apples).

DOS: See Disk Operating System.

Dump: Popular term for sending data from a computer to a mass storage device such as disks or tape.

Execute: A command that tells a computer to carry out a user's instructions or program.

File: A continuous collection of characters (or bytes) that the user considers a unit (for example on accounts receivable file), stored on a tape or disk for later use.

Firmware: Programs fixed in a computer's ROM (Read Only Memory); as compared to software, programs held outside the computer.

Floppy disks: See Disks

Hard disks: See Disks.

Hardware: The computer itself and peripheral machines for storing, reading in and printing out information. The parts of the computer which you can kick.

High-level Language: Any English-like language, such as BASIC, that provides easier use for untrained programmers. There are now many such languages and dialects of the same language (for example MicroBASIC, PolyBASIC etc).

Input: Any kind of information that one enters into a computer.

Input device: Any machine that enters information into a computer. Usually done through a typewriter like keyboard.

Interactive: Refers to the "conversation" or communication between a computer and the operator.

Interface: Any hardware/software system that links a microcomputer and any other device.

I/O: Acronym for "input/output".

K: Represents 1024 bytes. For example 5K is 5120 bytes (5 x 1024).

Kilobyte: See K.

Modem: Acronym for "modulator-demodulator." An instrument that connects a microcomputer to a telephone and allows it to communicate with another computer over the telephone lines.

Machine language: The binary code language that a computer can directly "understand".

Mass storage: A place in which large amounts of information are stored, such as a cassette tape or floppy disk.

MB: Represents a million bytes.

Megabyte: See MB

An ABC

Memory: The part of the microcomputer that stores information and instructions. Each piece of information or instruction has a unique location assigned to it within a memory. There is internal memory, inside the microcomputer itself, and external memory stored on a peripheral device such as disks or tape.

Memory capacity: Amount of available storage space, in Kbytes.

Menu: A list of options within a program that allows the operator to choose which part to interact with (see Interactive). The options are displayed on a screen and the operator chooses one. Menus allow user to easily and quickly set into programs without knowing any technical methods.

Microcomputer: A small computer based on a microprocessor.

Microprocessor: The central processing unit or "intelligent" part of a microcomputer. It is contained on a single chip of silicon and controls all the functions and calculations.

Network: An interconnected group of computers or terminals linked together for specific communications.

Output: The information a computer displays, prints or transmits after it has processed the input. See Input and I/O.

Pascal. A high-level language that may eventually rival BASIC in popularity.

PEEK: A command that examines a specific memory location and gives the operator the value there.

Peripherals: Any external input or output device that communicates with a microcomputer, for example disk drives.

Personal computer: A small computer for one's own use, whether in the home, school or business.

Pixel: Picture element. The point on a screen in graphics.

POKE: A command that inserts a value into a specific memory location.

Printer: Device that prints out information onto paper.

Program: A set or collection of instructions written in a particular programming language that causes a computer to carry out or execute a given operation.

RAM: Acronym for "random access memory". Any memory into which you "read" or call up data, or "write" or enter information and instructions.

REM statement: A remark statement in BASIC. It serves as a memo to programmers, and plays no part in the running program.

Resolution: A measure of the number of points (pixels) on a computer screen.

ROM: Acronym for "read only memory". Any memory in which information or instructions have been permanently fixed.

Simulation: Creation of a mathematical model on computers that reflects a realistic system.

Software: Any programs used to operate a computer.

Storage: See Mass storage.

System: A collection of hardware and software where the whole is greater than the sum of the parts.

Tape: Cassette tape used for the storage of information and instructions (not music).

VDU: Acronym for "visual display unit". A device that shows computer output on a television screen.

Word: A group of bits that are processed together by the computer. Most microcomputers use eight or 16 bit words.

CLASSIFIEDS

Acorn Atom: 12K RAM, hires colour, cassette, software, books, power supply, VHF modulator. Roger Altena, Box 8914, Symonds St, Auckland.

ANYONE out there an expert on parlour video games? "Bits & Bytes" is looking for someone who can keep the rest of us up to date on them. Drop a line to "Parlour Video", Bits & Bytes, Box 827, Christchurch.

CALCULATOR users. "Bits & Bytes" is looking for someone to write and review calculators, especially the advanced programmable types. Write to "Calculators", Box 827, Christchurch.

For Sale ZX80 computer, 3K RAM, p/s, software (on cassette and SYNC magazines). $150 o.n.o. P. Johnson, 669 Pukehina Pde, RD9 Te Puke.

FOR SALE: Commodore VIC 20. Free software, free introductory lessons, and a free Lions Xmas cake with every computer purchased. Only $899. Phone AK 278-3185 for appointment to view.

SELLING a micro or a peripheral? "Bits & Bytes" classifieds can help, and for up to 20 words, they are free. Write to "Classified", Box 827, Christchurch.

System 80 32K on board, in built sound, lots of software, $1020, monitor $95, phone Wn 269-056

FREE! Classified advertisements are free to hobbyists and students and non-commercial micro users generally. However, this applies only to the first 20 words; each word above 20 costs 20c. Commercial users' rates on request.

Wanted to buy (or swap) **ZX81** software (games)
Write M. Loughnan, Box 492, Palmerston North or phone 68-481 (PN).

CALLING ALL HAND-HELD USERS!

Micros don't finish at those of desk-top size. "BITS & BYTES" is interested in hand-held computers, too.

We would like to hear from any enthusiasts of these mini micros . . . Sharps, Hewlett-Packard, the brand is immaterial.

We will pay for good contributions for publication.

BITS . . & . . . . . BYTES . .

NEW ZEALAND'S PERSONAL COMPUTER MAGAZINE

If you want the latest micro news on

* education ✓
* business ✓
* clubs ✓
* hardware ✓
* software ✓

If you want micro

* games ✓
* hints ✓
* help ✓
* competitions ✓

.... then this is your magazine. There is a small subscription charge (to cover postage costs). Just fill out the card enclosed in the magazine or the coupon below.

No stamp is required if you
write this address on your envelope

Freepost No. 125
Bits & Bytes
P.O. Box 827
Christchurch

POST YOUR SUBSCRIPTION TODAY!!

Please tick appropriate boxes

NAME ....

ADDRESS ....

....

....

Please mark which field interests you

Education ☐
Competitions ☐
Business use ☐
Hobby use ☐
Farming ☐
General news ☐
Programs ☐
Hints ☐

School ☐ Adult ☐

If school name it ....

Form ....

I enclose payment for one year's subscription to Bits & Bytes Adult $8 ☐ School $6 ☐

Payment
Visacard ☐
Bankcard ☐

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Card holder signature ....

Date card expires ....

Cheque ☐ Cheque No. ....
Postal Order ☐ Postal note ☐

MICRO CLUB CONTACTS

APPLE Users Group: Bruce Given, 12 Irirangi Rd., One Tree Hill, Phone (09) 667-720 (h).

ATARI Microcomputer Users Group: Brian or Dean Yakas, Phone (09) 8363-060 (h).

COMMODORE Users Group: Doug Miller, 18 Weldene Ave, Glenfield, Phone (09) 444-9617 (h), 497-081 (w).

CP/M Users Group: John Lamb, 11 Martin Ave, Remuera, AK5, Phone (09) 546-192 (h), 771-729 (w).

DREAM 6800 users: Peter Whelan, 22 Kelston St, New Lynn, Auckland, Phone (09) 875-110 (h).

KIM users: John Hirst, 1A Northboro Rd, Takapuna, phone (09) 497-852 (h).

LNW users: Ray James, phone (09) 30-839 (w), 585-587 (h).

SORCERER Users Group (NZ): Selwyn Arrow, phone (09) 491-012 (h).

ZX80/81 users: Doug Farmer phone (09) 567-598 (h).

1802 Users Group: Brian Conquer, phone (09) 655-984 (h).

2650 Users Group: Trevor Sheffield, phone (09) 676-591 (h).

SYMPOOL (NZ SYM User Group): J. Robertson, P.O. Box 580, Manurewa, Phone (09) 266-2188 (h).

ACES (Auckland Computer Education Society): Ray Clarke, 1 Dundas Pl, Henderson, Auckland, Phone (09) 836-9734 (h).

OSI USERS GROUP (AK) Vince Martin-Smith, 44 Murdoch Rd, Grey Lynn, Auckland.

NZ TRS 80 USERS GROUP (AUCKLAND): Olaf Skarsholt, 203a Godley Rd. Titirangi, Phone (09) 817-8698 (h).

ATARI 400/800 USER CLUB: Dave Brown, P.O. Box 6053, Hamilton, Phone (071) 54-692 (h).

GISBORNE MICROPROCESSOR USERS GROUP: Stuart Mullett-Merrick, P.O. Box 486, Gisborne. Phone 88-828.

GISBORNE MICRO COMPUTER GROUP: Ron Taylor, 17 Byron St., Gisborne. Phone: (079) 81-450 (h).

WHANGAREI COMPUTER GROUP: Tom Allan, 3 Maunu Rd, Whangarei. Phone 83-063 (w). Meets every second Wednesday of the month at Northland Community College.

TARANAKI MICROCOMPUTER SOCIETY: P.O. Box 7003, Bell Block, New Plymouth: Francis Slater, Phone (067) 84-514.

MOTOROLA USER GROUP: Harry Wiggins, (ZL2BFR), P.O. Box 1718, Palmerston North, Phone (063) 82-527 (h).

OSBORNE USER GROUP: Dr Jim Baltaxe, 18 Matipo St, Palmerston North, Phone (063) 64-411.

HAWKES BAY MICROCOMPUTER USERS GROUP: Bob Brady, Pirimai Pharmacy, Pirimai Plaza, Napier, Phone (070) 439-016.

UPPER HUTT COMPUTER CLUB Shane Doyle, 18 Holdsworth Avenue, Upper Hutt. Phone 278-545. An all-machine club.

SUPER 80 USERS GROUP: Bruce Stevenson, Peanut Computers, 5 Dundee Place, Chartwell, Wellington 4. Phone (04) 791-172.

WELLINGTON MICROCOMPUTING SOCIETY Inc. P.O. Box 1581, Wellington. Meetings are held the 2nd Tuesday each month at 7.30 p.m., Block 2, VICTORIA UNIVERSITY.

CHRISTCHURCH MICROPROCESSOR USERS GROUP: J. D. Mann, 330 Centaurus Rd, Cashmere, Christchurch. Phone (03) 325-652.

CHRISTCHURCH '80 USERS GROUP: David Smith, P.O. Box 4118, Christchurch, Phone (03) 63-111 (h).

CHRISTCHURCH PEGASUS USERS GROUP: Don Smith, 53 Farquhars Rd, Redwood, Christchurch, Phone (03) 526-994 (h), 64-544 (w), ZL3AFP.

CHRISTCHURCH APPLE USERS Group: Paul Neiderer, C/- P.O. Box 1472, Christchurch, Phone (03) 796-100 (w).

OSI USERS GROUP (CH): Barry Long, 377 Barrington St., Spreydon. Christchurch. Phone (03) 384-560(h).

LEADING EDGE HOME COMPUTER CLUB: Elaine Orr, Leading Edge Computers, P.O. Box 2260, Dunedin. Phone (024)55-268(w).

NEW ZEALAND MICRO NEWS

In New Zealand's own personal computer magazine