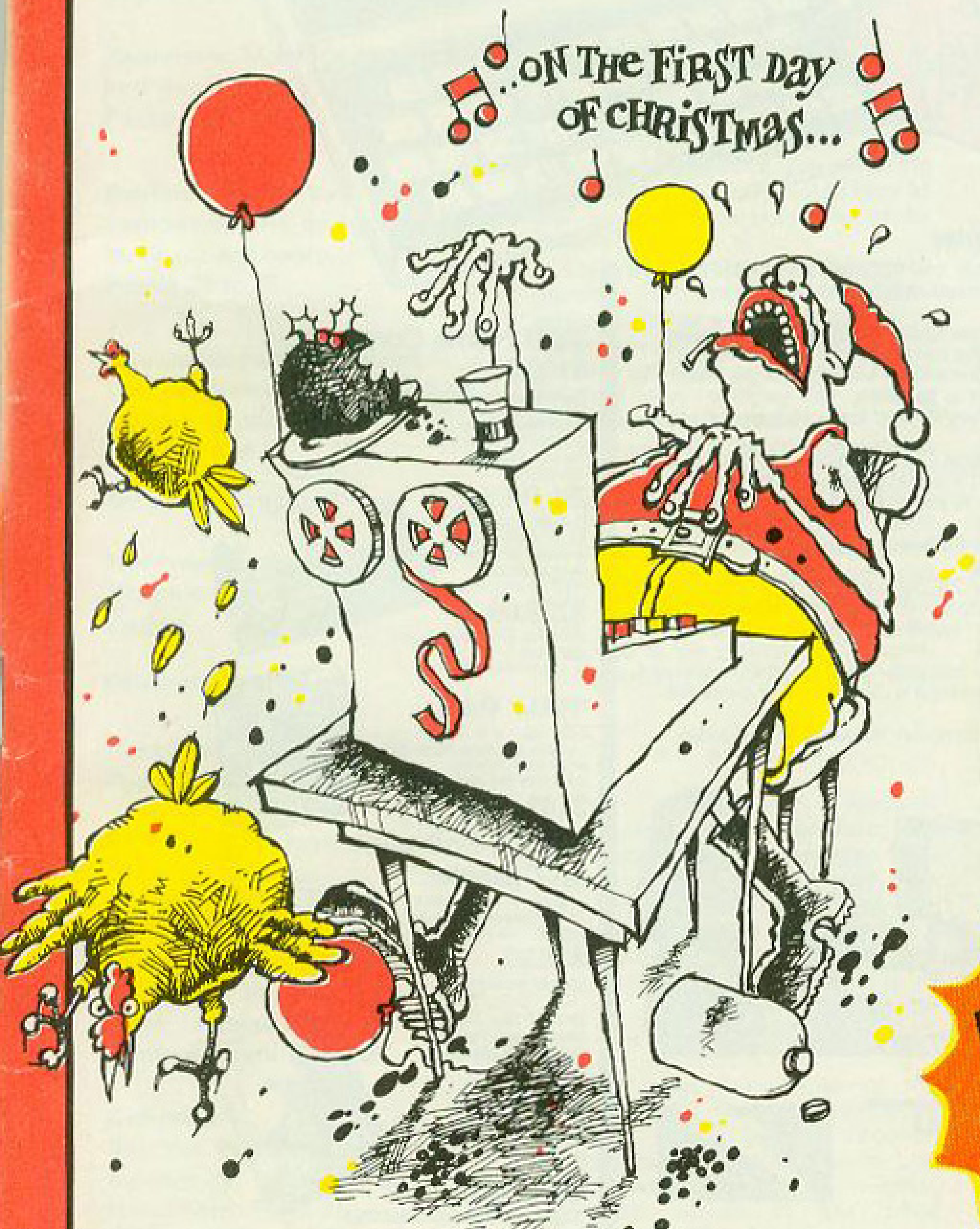
ZEALAND'S PERSONAL COMPUTER MAGAZINE

BITS & BYTES

Issue No. 4, Dec/Jan 1982-83: $1.00

A taste of COCO — a colour computer from Radio Shack



An adventure in computer games

Columns for ★ 80 Users
★ VIC
★ Sinclair
★ Atari

BASIC and binary for beginners

A program to check your homework

Micros on the farm

HAVE FUN WITH ZX81

New Sinclair ZX81 Home Computer

ONLY $199

ZX81 Home Computer

Mum, Dad. The whole family can have fun with ZX81! It's great for games, education, business, household budgets and calculations.

It's also an investment in the future . . . today's ZX81 helps prepare your children for the computer age of tomorrow.

It comes with a very comprehensive Manual that will teach you BASIC* programming skills at home!

ZX81 – Very Compact. Very Smart! Easy to Store! Easy to Run!

Starting is easy – ZX81 plugs into the aerial socket of your black and white or colour TV set and you can use your own portable cassette recorder to make and store your own programmes.

Our exciting range of ZX Software is growing every day – it's available on cassette!

Choose cassettes from our ZX Software Library – it currently includes Fun Games, Education (Primary, Intermediate and Secondary School Levels) Household Management, Games of Skill, Business, etc!

Kids have grown up in the 'computer age', so you'll find they'll be **showing you** how to arrange a 'Shoot Out' or cross the 'Perilous Swamp'!

*BASIC the most popular computer language in use today.

sinclair

ZX81

HOME COMPUTER

You'll see the sign

ZX81 Home Computer.

$199.00

Power supply not included.
(Power Supply $29.95)
If you have your own
9 V DC 700 mA Adaptor,
you're in business!

ZX Printer.

Connect a ZX Printer for an instant printout of what's on screen! Keep a record of anything you wish!

$239.00

Complete with Power Supply & 65ft roll of printout paper.

'RAM' Pack.

ZX81 has 1 K of memory but you can plug in a RAM Pack for 16 K times more memory.
It's a powerful combination!

$149.00

Sinclair Software on Cassette.

Available in 1 K and 16 K.
The 1 K cassettes use the memory built into the ZX81

$19.95

Ideal for starters!

This 212 page ZX81 Manual gives you a step by step course in BASIC programming and tells you everything you want to know about home computing!

DAVID REID ELECTRONICS LIMITED

THESE PLUS HUNDREDS OF OTHER ELECTRONIC PRODUCTS ARE AVAILABLE NOW AT YOUR NEAREST ELECTRONICS STOCKIST!

BITS & BYTES

Contents

Micro News... Micro News...

The retail price of Commodore's VIC 20 colour computer is being cut by $100 in December to $795.

Commodore in New Zealand say this is due to economies from the huge productior runs of VIC 20s now needed to meet world demand (they say it is now the No. 1 selling microcomputer of its type in the United States where prices range from $200-$300).

However the price cut is also helpful in meeting the stiffening competition on the New Zealand market.

* * * *

One example of this competition is the newly released Dick Smith Wizzard colour computer, retailing for $495. It will be promoted here as the cheapest home colour computer in New Zealand.

The Wizzard comes with a touch sensitive keyboard and two paddles, which together with the high picture resolution (49,128 pixels) and plug-in game cartridges have made the Wizzard very popular for playing video games in Australia.

However Dick Smith Electronics emphasise the Wizzard is a true computer with 1K user RAM and a BASIC programming cartridge supplied. As well the touch sensitive keyboard can be easily replaced with a typewriter-style keyboard and a tape cassette unit can be attached to the side of the Wizzard. More about the Wizzard in a later issue.

* * * *

The release of National Panasonic's JR 100 computer has been delayed until early in the New Year and New Zealand agents, the Microcomputer Electronic Company, are now talking about a retail price of $390-450 (compared to the $595 originally mentioned).

* * * *

Soon after the release of the Epson HX-20 portable computer, Epson's New Zealand agents, Microprocessor Developments Ltd, are releasing another Epson computer aimed at the small business market.

The QX-10 should be available here early next year for around $4600. That price includes 128K RAM, 128K Video RAM and two disk drives.

The QX-10 has a picture resolution of an incredible 260,000 pixels which should mean very impressive graphics.

Space for up to 25 optional cards, including expansion to 256K RAM, is included in the QX-10 which runs the CP/M operating system. A software package called Veldox which includes a word processor, database manager, personal filing program and financial calculations program aimed at first time users, will also be available.

* * * *

January 19. That is the date for the world wide release of the new Apple computer and a modified Apple II computer.

The new Apple is reported to be aimed at large computer users and a radical departure from the personal computer market.

Overseas reports indicate the new Apple, code-named Lisa, is a 16 bit executive work station.

Of more interest here, initially anyway, will be the new Apple IIe of which BITS & BYTES has had a sneak preview but at this stage we can only report it features several enhancements on the present Apple II. The price is not yet known.

A new disk drive unit will also be announced by Apple on January 19.

Meanwhile CED Distributors, who have just been appointed sole Apple agents in New Zealand for another two years, say the Apple III is definitely not dead and it will soon be announcing a special promotion on that model.

A review of the Apple III will appear in our March issue.

* * * *

CED distributors is also behind moves to establish a microcomputer database in New Zealand similar to "The Source" in the United States or "The Australian Beginning" in Australia, although on a much smaller scale.

Continued on Page 3

One way micro computers can help N.Z.

Microcomputers are for more than fun and education. That's why this fourth edition of BITS & BYTES has a special section for farmers.

Agriculture is still our lifeblood in New Zealand. The feeling of economic pessimism that spreads when wool prices slide and freezer storage is choked up with unsold lamb reflects this.

So it is in the interests of all New Zealanders that our farmers gain, as quickly as is efficiently possible, the benefits and help that microcomputers offer them. The country has not been lagging: pioneering work by the Kellogg unit at Lincoln College (featured in BITS & BYTES in September), is now part of a mushrooming farm software industry as private enterprise enters the picture.

The software being offered to farmers today will be supplemented by much greater things in the years ahead. Farmers will undoubtedly have network links so that their home machines can process data from stock and wool sales of the day. Data from producer boards, accountants, computer consultants, and Government departments will flow to and from microcomputers on isolated farms.

But that's tomorrow. The benefits of today are outlined in the special articles in this issue.

As well as help in running farms, micros offer the same attraction to country folk as to city enthusiasts: intriguing games, intellectual challenge, and scores of uses from writing letters to storing household accounts. They are fun and mind extending.

For the city enthusiasts, a thriving and computer-aware country population will help get a fairer tax deal for microcomputer imports, and, if farm efficiency is widely lifted, more overseas funds for hardware.

Coming up in BITS & BYTES

Hardware Reviews

We are having trouble keeping up with the spate of recent microcomputer releases in New Zealand but in our February issue we hope to have at least three reviews;

- © The Epson HX20
- © Hitachi Peach
- © Sirius 16-bit

Hand-held

Our focus will be on hand-held micros next month and we will be featuring reviews of the latest models as well as program and hints for established models.

Farming

A look at some farm software developed in the deep south, including programs for deer farmers, and Dr P. L. Nuthall of the Kellogg unit gives his views on farming and computers.

Business

John Vargo continues his series on selecting a small business microcomputer.

BITS & BYTES, No. 4, December 1982 and January 1983. The next issue will be February 1983. BITS & BYTES is published monthly, except January, by Neill Birss, Dion Crooks and Paul Crooks. Editorial and subscriber inquiries to Post Office Box 827, Christchurch.

ADVERTISING:
please note the following advertising contact addresses.

Auckland - Cathy and Selwyn Arrow: Phone 491-012
Wellington - Annie Carrad Phone: 723-431
Elsewhere - Graham Beecroft: Phone 554-265 Chch.
EDITORIAL
Auckland - Cathy and Selwyn Arrow: Phone 491-012

Production: Roger Browning, Graeme Patterson, Lee Tecksui, Janine Morrell
Typesetting: Gail Rarkin
Cover: Trace Hodgson
Graphics: Sally Williams

Subscription rate: $8 a year (11 issues) adults
$6 a year for school pupils

Technical Editor: Chris O'Donoghue

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.

TELL US! Write to us

From Page 1

Access to the database, which initially will have an educational flavour will be via modems or acoustic couplers.

CED says it has now found a low cost coupler and is having discussions on the hardware for the mainframe computer to store the programs.

There will be a yearly subscription charge to the service as well as a charge each time the database is used and CED says that database will be open to any brand of microcomputer not just Apples.

Again an announcement will be made early in the New Year.

* * * *

Japan's largest selling personal computer is now on sale in New Zealand.

The NEC PC-8000, described as a personal computer for professionals, sells here for $1690.

Based on a 280A compatible microprocessor, the PC-8000 has 32K RAM (expandable to 64K) and 24K ROM including Microsoft BASIC.

The PC-8000 runs NEC DOS but CP/M is available for an extra $200.

Interfaces for colour or black and white screen, audio cassette and printer are built in. An expansion unit costing $1280 or $1460 is required for disk drives (single sided $2110, double sided $2340).

* * * *

The Australian designed and built Microbee Computer was officially released in New Zealand at a function in Wellington recently.

The Microbee is finding increasing acceptance among Australian schools and the manufacturers, Applied Technology Ltd, say there are now more Microbees installed in Australian schools than Apple computers.

Judging by its price the Microbee seems certain to be equally popular here among schools and home hobbyists.

Micro News...

The 16K version sells for $829 ($525 for schools) while the 32K version is $990 ($660).

BITS & BYTES will have a full review of the Microbee in a coming issue.

* * * *

AWA New Zealand Limited has announced the release of the Hitachi MB6890 Microcomputer in New Zealand. Hitachi have long been active on the mainframe computer area but this machine signals their entry into the microcomputer market.

The machine, the Hitachi Peach, has achieved outstanding success in Australia where it has earned a reputation for reliability and performance.

The Peach is based around powerful 6809 CPU and has a large number of options fitted as standard. These include light pen, cassette, RS-232C, Centronics, monochrome and RGB video interfaces, separate numeric key pad, 40/80 column screen under both hardware and software control, full eight colours and inverse, in-built loud speaker with volume control, six internal expansion slots and two memory expansion slots.

A wide range of software is available to run on the machine including the Hi-Writer Word Processing package, Hi-Finance Accounting System (written especially for local conditions), Procalc electronic spread sheet and many others including a wide range of games.

In announcing the release of this product, Mr R. V. Johansen, manager of the Data Systems Division of AWA New Zealand Ltd, said that thecompany was establishing dealers throughout N.Z. and that AWA would be providing both hardware and software support for the product.

BITS & BYTES will feature a review of the Hitachi Peach in February.

BBC Series

The 10-part television series screened by the BBC as its computer literacy project will be broadcast in New Zealand by TV1, beginning in March. Pitman Publishing, NZ, Ltd, and Access Data, Ltd, will market a package on the series comprising an introductory book on computing, a video series, and a microcomputer and software.

Micro News... Micro News...

COCO 'a good fun machine'

By L. A. BISMAN

After a very long time coming to New Zealand, the Radio Shack Colour TRS-80 Computer is here and available.

The Colour Computer (COCO) is available either as a 4K RAM machine or as a 16K machine with a more powerful BASIC language ROM. The model used for this review was lent to me by TRIAD COMPUTERS of Colombo Street, Beckenham, Christchurch. I decided to use the 16K version. The price difference is about $300, the more powerful machine selling for about $1500.

Opening the box I was confronted with the following items: an operating manual; a colour BASIC manual; and, continuing the Radio Shack tradition, a quick reference fold-out card; an extended colour BASIC manual; a television-set-to-computer connecting cable; and, of course, a Colour Computer.

After unpacking I found that the directions were very easy to follow and in short order had the computer hooked up to my colour television set and a tape recorder (neither included in the $1500 price).

The computer is in the grey-plastic common to all Radio Shack computers and peripherals. There are not a lot of leads "hanging" from the computer, as the power supply is inside the case (after running the machine all day, no overheating was apparent). The whole unit measures some 37cm by 34cm and rests quite securely on rubber feet.

The COCO has a built in real-time clock and RS-232 port. At the back are din sockets and other connectors for tape recorder,

joysticks, modem, and ROMPAKS. Also at the rear are the on/off switch and the RESET button. The keyboard at first glance looks a little like a toy typewriter but it is in fact very sturdy. The keys appear to stick up more and be separated more than is usual on keyboards. But after a thorough work-out (I am an 80 w.p.m. touch-typist), I decided that it was really quite easy to use. There were none of the "old" problems such as key-bounce, etc. In fact, the keyboard lends itself quite nicely to a template, (no more sticking things on the edges of the keys for special functions).

The screen display is very clear, and easy to read. Only upper-case letters, though I believe there is some software, (yes, I said software) available to re-design the character set so that 50 characters per line and 25 lines upper and lower case with true descenders are available. This is achieved by utilising the high-resolution graphics which are available. Without this modification, or indeed some hardware equivalent, we must make use of the 32 characters per line and 16 lines of upper case only characters. This I found to be the COCO's worst feature.

At power-on, the screen displays black on green, but this is under software control and may be altered quite easily. Nine colours are available (the manual says eight, but it doesn't count black). These are black, yellow, green, blue, red, orange, magenta, buff, and cran. These colours may be used in any of five resolutions of density. From 32 x 64 up to 192 x 256. However at the highest resolution only four colours may be used.

Although the COCO has branched away from the familiar Z80 series chip to a 6809 series, the powerful dialect of BASIC as used on other TRS-80 models has been retained and added to. The familiar "EDIT" facility remains almost unchanged and users of "LEV3 BASIC" or its copies will recognise powerful graphics commands for drawing lines, boxes, filled in boxes, etc. New commands have been added such as "CIRCLE" and "PAINT", which automatically paints a whole area (any size or shape) a specified colour. Another command, "DRAW", is so versatile that it is possible to draw lines in eight different directions at once and to scale your drawing either up or down.

The COCO has built in "SOUND" and "PLAY" commands. Pitch and note length are easily determined. Using very simple programming techniques and about 10 program lines I was able to write a Morse-code generating program with the aid of these "music" commands.

In the time I spent with the machine I saved and loaded many tapes and despite the fast loading speed of 1500 baud, did not experience a bad load. Two features that I particularly liked were being able to use an 8-byte name on tape and the ability to load and save machine-code programs. It is necessary to know the start, end, and entry point if you want to save a machine-language program to tape. It isn't necessary for BASIC of course.

ROMPAKS are definitely the way to load programs. Plug one into its slot and almost instantly the program begins execution faster than a speeding bullet! All of the ROMPAKS I used were of games. Aren't there a lot of them, too! The quality of the high resolution graphics and sound was enough to at least rival the machines found in burger bars, and in some cases, I am sure, surpass them. All the old favourites are available. There I was being blown to bits by almost everything and loving every minute of it.

To sum up, the COCO turned out to be everything the advertisements said it would. The price is very competitive and the availability of software already is overwhelming. It's a pity that all those MOD I and MOD III programs won't transfer straight across, but I have read that Radio Shack is re-writing all the good software such as EDITOR ASSEMBLERS and so on for the COCO. If you want to run a business, this isn't the computer for you, but if you want to learn computing and have a lot of fun doing it, this computer would fit that bill perfectly.

Write to us

Blaze of sound and colour at show

AUCKLAND REPORTER

Trillo's, was alive with hi-fi sound at all levels from every direction, and video in every colour on large and small screens, at the Consumer Electronics show in Auckland recently.

Organised and promoted by the Guild of Electronic & Appliance Retailers (Inc), the four day show enabled both trade and public to make realistic comparisons of both new and established electronic products - including video recorders, stereos, home appliances and computers.

TELETEXT

While TVNZ may be having a difficult time negotiating the introduction of Teletext, this proposed television public information service was enthusiastically viewed at the show.

Using a spare section of the television signal to transmit "pages" of information displayed on specially equipped home receivers in written and graphic form, the service is free to the viewer.

Selected programmes can also be transmitted with captions for the hard of hearing These would appear on the screen over the picture. For normal use the Teletext pages would replace the usual pages.

A special decoder is necessary to enable the information to be displayed, and these will be available on the New Zealand market when teletext commences.

Initially the decoder will be a separate unit. However, Thorn is building decoders into sets for export to Australia.

While TVNZ is at present transmitting only eight test pages of teletext as a service to the trade, hundreds of pages could soon be available. Britain has approximately 800 pages on call 24 hours a day and information can also be relayed from screen to home computer for later retrieval.

VIEWDATA

Also on demonstration was the marriage of three technologies - the computer, the telephone and the television set - which together will provide a low cost, simple and fast electronic information system known as viewdata.

It will be available by simply dialling a telephone number and accessing any of thousands of screen pages of information stored in a main-frame computer. The subscriber can choose the exact information required by pushing the buttons on a 12-key pad. At present in Britain, there are more than 200,000 pages available on over 450 subjects.

ROBOT ARM FOR THE COMPUTER HOBBYIST

Freshly arrived in New Zealand from Mitsubishi, the RM-101 Move Master can be easily controlled by many of the widely available home computers. With five degrees of freedom and six axes of rotation under simultaneous microcomputer control, it uses the same principle of operation as an industrial robot.

Just the device for the experimenter, classroom or laboratory, it comes complete with three different hands and off-the-shelf software. It can also be programmed in BASIC or assembly language and is easily connected to most computers via the standard centronics interface.

The Mitsubishi MX-6000 computer with its CP/M operating to control two RM-101's for operation. In particular it has a new sophisticated two-handed operation, in particular it has a new robot oriented language M-Roly.

Complete with a built in CRT display screen, 58K user RAM, floppy disk drive and a printer, it's a computer in its own right, apart from its advantage of dual RM-101 control.

INTELLIGENT GAMES

With its first public release at the show, Compu game utilises the very latest micro chip technology. UVI -

Continued on next page

From previous page

user video interface - allows the microprocessor in the unit to intelligently interpret the video information.

Its features include more than just another space invader game. Already available are 8 plug-in game cartridges with ten more due soon. It has high definition pictures, more realistic sounds and self-centring joysticks with associated keypads. One example of this product's ingenuity is that the two keypads can be clipped together to form a 30 button keyboard for use with a foreign language/English translator cartridge. This is a New Zealand designed and buit product and is marketed by Orbit Electronics.

On the AWA stand one Hitachi Peace computer running a business program attracted serious inquiries

Meanwhile right alongside was New Zealand's first working display of a digital audio disk system. The shiny silver disk will never wear out as there is no direct contact between disk and its laser type pick up head. A microprocessor is utilised to search and select any track by simply pressing buttons.

WELLINGTON REPORTER

We need people to review new products and keep us informed on other microcomputer developments in the Wellington area.

If you are interested in earning some extra money please send details of microcomputer experience, writing experience, availability and so on to WELLINGTON REPORTER, BITS & BYTES, BOX 827, CHRISTCHURCH.

Back copies

Unfortunately no more copies of Issue No. 1 are available. Copies of issues 2 and 3 are available for $1 plus 50c postage and packaging. Send your cheque or postal note to Back Copies, BITS & BYTES, Box 827, Christchurch.

Fantastic world of Adventure games

By STEVEN DARNOLD

You are standing at the end of a road before a small brick building. Around you is a forest. A small stream flows out of the building and down a valley.

So begins the original Adventure program, written in the mid-70's for the PDP-11 computer. Many thousands of people have searched for the entrance to Colossal Cave and explored the wondrous caverns within. Many have fallen in pits, many have been attacked by dwarves, many have lost their way in mazes. Some have even crossed the Troll's Bridge and seen the breath-taking spectacle beyond.

The original Adventure program has been so popular that, not only has it been adapted to run on most micro-computers, but it has also given rise to over a hundred other Adventures. Some of these Adventures match the quality of the original; others, however, fall short. The purpose of this article is to examine some of these Adventures and distinguish between the good and the not so good.

ORIGINAL ADVENTURE

I recommend this Adventure very highly. It is particularly effective in the way it uses detailed descriptions to weave a rich tapestry of images. However, since these detailed descriptions take over 60K of memory, most versions of this program require a disk drive.

SCOTT ADAMS' ADVENTURES

Scott Adams began writing

Adventures so long ago, that he must be considered one of the founding fathers of the genre. At last count, he had written 12 Adventures: Adventureland, Pirate Adventure, Mission Impossible, Voodoo Castle, The Count, Strange Odyssey, Mystery Fun House, Pyramid of Doom, Ghost Town, Savage Island (two parts), and Golden Voyage.

Unfortunately, only the first two of these Adventures are available for my computer - they are both excellent. Adams plots his Adventures very tightly and injects them with a delightful sense of humour. Highly recommended.

GREG HASSETT'S ADVENTURES

Greg Hassett has written six Adventures: House of Seven Gables, Sorcerer's Castle, King Tut's Tomb, Trip to Atlantis, Enchanted Island, and Journey to the Centre of the Earth. I have played the first four. Unfortunately, I found them tedious. Hassett had, perhaps, one or two clever ideas in each Adventure. If he had put them all into just one Adventure, it might have been a good one. The most interesting aspect of the Hassett Adventures is that the actual programming is very soundly done. Anyone interested in writing their own Adventure should consider getting one of the Hassett Adventures and using its structure. In fact, a little bit of reworking could turn a Hassett Adventure into something quite nice.

LORDS OF KARMA

I have mixed feelings about this Adventure. It is nicely presented and it incorporates a number of good ideas; however, it is too loosely plotted. The player wanders around, doing a bit of this and a bit of that. There is no sense of progress through the Adventure and no sense of finality at the end. Is 100 karma points enough? Is 1000 too many?

AFRICAN ADVENTURE

This is quite a nice Adventure. The environment is imaginatively designed and there are several interesting obstacles to overcome. There are not enough hints for beginners, but more experienced Adventurers should have no problem.

DOG STAR ADVENTURE

I enjoyed this Adventure, although it is somewhat shorter than normal. The geography of the space ship is nicely described and there are several interesting events. I was, however, perplexed by some of the plot elements. For example, it does not seem reasonable to feed a hamburger to a robot.

Continued on next page

From previous page

LOST DUTCHMAN'S GOLD

This rather ordinary Adventure is enlivened by some nice touches of humour. I enjoyed playing it, although it would have been much better had it been longer and fuller.

HITCH-HIKER'S GUIDE TO THE GALAXY ADVENTURE

So far I have dealt only with Adventures that run on several different brands of computers. However, I would like to mention one Adventure that is available only for Commodore computers. The Hitch-Hiker's Guide to the Galaxy made interesting reading, interesting radio, and interesting television. It is not surprising that it makes an interesting Adventure.

The problem is that the Adventure's author has made the obstacles so difficult to overcome, that few players ever get past square one. I would recommend this Adventure only to very experienced players and only if they are masochists.

AUTOMATED SIMULATIONS

Although this company does not claim to produce Adventures, I often see their games listed as such. I have played Datestones of Ryn, Hellfire Warrior, and Dragon's Eye - they are definitely not Adventures. All of them involve moving a graphic character around the screen using one-key commands. A player's actions are very limited, and there is little scope for puzzle-solving or humour. Not recommended.

Small Business Users Group

With the increasing number of microcomputers being used in small business, a small business user group, using the facilities of the N.Z. Micro Club is now being formed.

The nature of business conducted by members will naturally vary, but there are many common business procedures and mutual interests in microcomputing to be shared.

If you are interested in this group, we would appreciate you contacting Selwyn Arrow at the address below, at any NZ Micro Club meeting, or call on (09) 491-012 (between 9 a.m. and 9 p.m. please).

As sufficient interest has already been shown, a meeting will be held on Monday 13th of December, starting at 7.30 p.m., at 30a Bracken Ave, Takapuna

The postal address is: Small Business User Group, P.O. Box 6210, Auckland.

Group disbands

The Christchurch Microprocessor Users' Group has disbanded.

Gisborne Group

Gisborne has a newly formed computer group. The initial gathering of interested bods was in August, with 47 present - quite a few interested had other commitments that evening.

The group meets on alternate Wednesday evenings at the Tairawhiti Community College and is now formally constituted as:

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

Types of processors possessed or accessed total at least 14 at present, of which Sinclair has the most users at about 18, down to at least nine people who have none yet. Most are listed as "hobby", but quite a few are "professional, technical, or educational" - one is "farm management".

Kid-glove treatment for hard disks

Handled like "new born babies". That's the treatment given to hard disks now they can be sent to Auckland for servicing.

Having nothing at all to do with Olympic discus throwers, these hard disks are large-capacity data-storage devices used in many of today's microcomputer systems by both small and medium sized businesses.

Otherwise known as a Winchester disk drive, this device provides a microcomputer with the data storage and power of a small mainframe computer. Typically each drive provides fast access to five or ten million bytes of data (one byte equals one character).

This data is recorded on the surface of an aluminium disk (like a 45 RPM record) that has been coated with magnetic recording material similar to the red-brown material on a tape recording. This disk along with its record/playback head, is enclosed in a hermetically sealed compartment along with its drive motor and control electronics.

These hard disks are somewhat different from the "floppy" disks which are constructed of flexible mylar, coated with a magnetic oxide and enclosed in a square jacket. Each floppy disk can hold from one hundred thousand to over one million bytes of data depending on size and type. Each floppy disk is inserted into a suitable drive unit as required so that several disks can hold a lot of data, but any specific item is not always available on call as is the case with the hard disk.

Dust is one of the greatest enemies of all types of data storage disks. As the head "flies" on a tiny cushion of air immediately above the magnetic surface, a single particle of smoke or dust could become lodged under the head thereby scratching the magnetic surface. This would severely damage the disk and possibly the head, causing a "head crash" resulting in serious loss of data and perhaps costly repairs to the drive.

Winchester-type hard disk units have been readily available for the last two years in New Zealand. The very latest models now on the market have two or more disks stacked together in the one drive to increase the drive's total capacity two or three fold.

Until recently each hard disk was permanently fixed in the drive at manufacture, but the latest news is that a removable platter type should be available soon.

By SELWYN ARROW

Most manufacturers guarantee their hard disks for ten thousand hours of operation and up until October the only way to obtain service under this warranty was to despatch it overseas for servicing at the manufacturer's special clean room facilities, usually in the United States or Japan.

An uncontaminated working atmosphere is an absolute necessity for servicing sealed hard disks. These conditions have now been achieved for the first time in New Zealand by an Auckland company, Advanced Control, Ltd, of Newmarket.

It has set up an "incubator" type desk top clean-room which is pressurised by an absolute three stage air filter system to keep out dust and contaminants. The incubator is enclosed in a semi-clean working cabin

Its use effectively eliminates the costly overseas return freight costs and loss of valuable computer time is kept to an absolute minimum as the unit can be back in full service in a week or so, not months as before.

The Advanced Control, Ltd, facilities also enable them to service most brands of 5¼" and 8" floppy disk drives. They also market the Tandon range of disk drives, Xebec intelligent hard disk systems, as well as a range of Zeda and Zenith computers.

In the belly of the beast

-explaining binary code

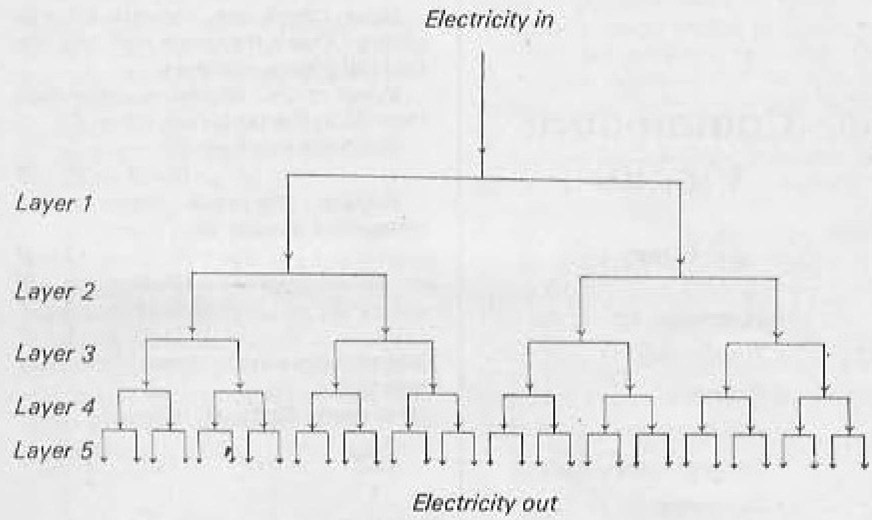
By GERRIT BAHLMAN

We have discussed the role of high and low-level language and the interpretation process called compiling. To appreciate what machine code is we need to look at the binary number system and see how it works. A BIT is a BInary digIT and obviously machines must be operating using binary arithmetic. Incidently a BYTE is 8 binary digits or bits. A computer WORD is the number of bytes that a particular machine can use at once. When people talk about 8, 16, or 32-bit machines they are talking about the word size of that machine. One rule of thumb is that the bigger the word a machine uses the faster it works.

The "innards" of a computer (the inside of the chips) are just a form of stone or crystal. The composition of the crystal is very special and has been built up very carefully. It has some strange properties when electricity is passed through it. These properties vary from chip to chip according to the particular job they have to do. Some parts of the chip let electricity pass through quickly and easily, other parts take a great deal before electricity will pass through. The flow of electricity can be changed by technically complex means, just as a switch can stop or start a light bulb. It would take too much specialist explanation to describe exactly how these properties are achieved and we will not go into that here. The important point is that electricity can flow through in particular ways and those directions can be switched. It is because of this ability to switch that we are going to look at binary arithmetic.

The base 10 number system

Humans use 10 digits to represent numbers, presumably because we have 10 digits called fingers. But the computer has two directional switches. So it only uses two digits for its number system. We use 0 1 2 3 4 5 6 7 8 9; computers use 0 1. If you look at the diagram you will see how quickly switches can be used to generate large numbers of alternative pathways. The more pathways - the bigger the number being represented.



In just five layers of switches we can send an original single line of electricity out in 32 different directions. How many would you have if there were eight layers? 16? While this might be interesting, how does it relate to computers. .

Remember, computers are made of crystals that can "switch" and not just once but thousands of times. Each switch can be turned one way or the other. Either on or off. So each switch can represent one binary digit. How many binary digits or bits do you think there are in a machine with a memory of 16K? Recall that 16K means 16 thousand bytes and each byte is eight bits. Would we use all those switches to represent one number?

The two digits used are 0 and 1 and they represent a switch being off or on. Examine the following table:

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7
1	2	4	8	16	32	64	128
2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	
256	512	1024	2048	4096	8192	16384	

Actually we are used to using a table like this already. What does 78693 mean?

78693

7x10000 8x1000 6x100 9x10 3x1

When we expand numbers into their units, tens and hundreds etc. we are using a table much like the binary one. i.e.

10^0	10^1	10^2	10^3
units	tens	hundreds	thousands
10^4		10^5	
tenthousands		hundredthousands	

We will use the binary table to convert base ten numbers to base two numbers. e.g.

78693_{10}-10011010011110101_2

The binary number system

We only use two digits, 0 and 1. But all the normal arithmetic operations like addition and subtraction still work e.g.

$$\begin{array}{r} 12_{10} \\ +5_{10} \\ \hline 17_{10} \end{array} \qquad \begin{array}{r} 1100_2 \\ +101_2 \\ \hline 10001_2 \end{array}$$

Continued on next page

The addition rules for binary arithmetic are simple:

(i) 0 + 0 = 0
(ii) 1 + 0 = 1
(iii) 0 + 1 = 1
(iv) 1 + 1 = 10

This is the strange one!

How can 1 + 1 = 10?? If you look back at the binary table recall that we have units and a "twos" column instead of a tens column. So whenever you get 2 you add one to the "twos" column. The same thing would happen if you got two 2's. You would add one to the "fours" column.

Converting from denary to binary

How could we convert 93_{10} to binary? Once more we will use the table of binary numbers.

What is the largest number less than 93 in the table? i.e. $64 = 2^6$

Subtract this from 93.

$93 - 64 = 29 \quad 2^6$

Repeat this until there is no remainder greater than one:

$29 - 16 = 13 \quad 2^4$
$13 - 8 = 5 \quad 2^3$
$5 - 4 = 1 \quad 2^2$
$1 - 0 = 1 \quad 2^0$

We can now write down the binary number.

How many digits will it have?

1×2^6	$+ 0 \times 2^5$	$+ 1 \times 2^4$	$+ 1 \times 2^3$
1	0	1	1
$+ 1 \times 2^2$	$+ 0 \times 2^1$	$+ 1 \times 2^0$	
1	0	1_2	

Notice that there is a '1' in each of the columns for which there was a subtraction.

As an exercise try doing the following addition problem in binary. You should convert all the parts to binary first and then do the adding. Convert your answer back to denary to check.

125 + 43 = 168

Converting from binary to denary

There are several different ways but we will use our tables to interconvert numbers. Let's take an example:

$1 \quad 0 \quad 0 \quad 0 \quad 1_2$

if we expand this number we get:

$1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 0 \times 2^3 + 1 \times 2^4$

units twos fours eights sixteens

$1 0 0 0 1_2$

Working this out we get:

1 + 0 + 0 + 0 + 16 = 17

Let's take another example: Convert 101010_2 to a base ten number.

Try it and see if you can do it.

When a computer is given a base 10 number it has to convert it to binary before it can do any arithmetic on it. It also has to reconvert it to base 10 before we can understand it. Naturally, it does this very quickly and we are never conscious of it doing this.

In fact, alphabetical letters are also converted to binary numbers with a particular number in binary representing each character. These special binary numbers that are used as codes for character are often called bit patterns. There are a number of different bit patterns systems used. You may have heard of them.

ASCII - American Standard Code for Information Interchange.
BCD - Binary Coded Decimal
EBCDIC - Extended Binary Coded Decimal Interchange Code.

The computer uses a conversion table whenever it is instructed to read or write information so the interconversion of letters to binary numbers is very quick.

In the next article we will continue our examination of the innards of the beast by looking at the way in which a computer is arranged. We will look at its parts and begin to appreciate what it does when a program is run.

School news

BITS & BYTES would welcome more general news about computers in schools. Other schools are interested in hearing what sort of computers you have, special projects, how they are being used in curricula, and any special achievements by pupils.

We are specially interested in hearing about how country schools are getting on with computers. From our subscriptions we know there is a high level of interest in many country areas, such as Western Southland.

So hook up the printer, get out the typewriter, or even the pen, and drop a line to "School News", BITS & BYTES, Box 827, Christchurch.

Computer Conference

The Computer Society is inviting papers for its eighth national conference in Wellington in September, 1983. The theme of the conference is "Entering the Information Age".

Basic BASIC No. 3

By GORDON FINDLAY

Last month we met variables, and ways of saving and retrieving, values. We also saw one way of constructing a loop - making some action happen a number of times. First off this month, let's look at that again - here is a simple problem, which I want you to try BEFORE you read on:

Write a program to print a table of discounted prices - making the discount 10 per cent, and prices 10, 20, ...100 dollars.

Done it - and got it to work? Let's try to write down the thinking that is needed to construct this program. First we know the way to get the discounted price is to multiply the original price by 0.9. This means we need two variables, one for the original price, one for the discounted price. I will call them OP and DP (Original Price and Discounted Price - tricky huh?). The heart of the program will be the statement:

```
DP = OP * 0.9
```

This statement calculates the discounted price all right - but needs the original price to have a value already. So, give it one:

```
OP = 10
```

and we need to change the value of OP every time we go around the loop loop:

```
OP = OP + 10
```

The loop is easy - just a GOTO statement. We should also print the results out:

```
PRINT OP,DP
```

Putting these together, we have:

```
10 OP = 10
20 DP = OP * 0.9
30 PRINT OP,DP
40 OP = OP + 10
50
60 GOTO 20
```

The missing line 50 must somehow stop the loop. The statement required is obviously an IF:

```
IF OP> 100 THEN STOP
```

STOP is a statement whose action is very easy to follow! This is what our completed program looks like:

```
10 OP = 10
20 DP = OP *0.9
30 PRINT OP,DP
40 OP = OP + 10
50 IF OP >100THEN STOP
60 GOTO 20
```

Incidentally, there is another way of stopping the loop. Look at the two statements 50 and 60 above. They can be interpreted in a slightly round-about way as 'If the value of OP is less than or equal to 100, then goto line 20, otherwise stop'. This translates into the BASIC statement:

```
50 IF OP < = 100 THEN GOTO 20
```

Line 60 then isn't needed. This is often worth checking for - can a program be improved by 'reversing' the question?

REMARKS

Programs are notoriously hard to follow. Many a programmer has found that he cannot recall what a particular statement is for even in a program he wrote a few weeks ago. And as for trying to modify a program somebody else wrote! Yet often a program has to be modified to fix a bug, or because some new facility is wanted, or to make it more efficient. The REM (short for REMark) statement is intended to make it possible for a programmer to include notes in the program as to what the program is doing, the conditions required for its use, or anything else which the user ought to know. Very often the REM statement is used to indicate the author's name, copyright information, etc.

The statement is simplicity itself. After the line number place the word REM, then anything you like. The computer will ignore anything after the word REM until the next line number. Here is the little program above, with a few remarks included:

```
5 REM PRINT DISCOUNTED PRICES
10 OP = 10
15 REM ALLOW 10 per cent DISCOUNT
20 DP=OP * 0.9
30 PRINT OP,DP
```

Continued on next page

Basic BASIC 3

(Continued)

```
40 OP = OP + 10
45 REM CHECK TO SEE IF DONE ENOUGH
50 IF OP > 100 THEN STOP
60 GOTO 20
```

There is absolutely no restriction on what may be included in a REM statement - after all, the computer completely ignores the statement apart from the first word!

Be sensible in your use of REMs. The most common fault is omitting them, but make sure that the ones you include are useful. Especially include REMs in parts of the program which are hard to follow or were hard to get working - these are the bits which will be difficult to maintain. One of the most useful things to include in a REM statement is a careful definition of exactly what each variable is used for. The only reasons for not using REM statements are firstly laziness - they do have to be typed - and secondly they do take up room in memory.

Input

A program which has all of the information it works on built in is not very versatile. Here is a program which calculates the sale price on an item whose normal price is $789.95, with a reduction of 15 per cent.

```
10 REM OP IS THE ORIGINAL PRICE, SP THE SALE PRICE
20 OP = 789.95
30 SP = OP * (1-15/100)
40 PRINT OP, SP
50 END
```

Now this is not the most elaborate piece of software ever written. It does the job - but what happens tomorrow, when the item costs $452.19 full price, and the discount is only 12.5 per cent? The whole program would need to be rewritten. Why not let the user of the program TELL the program the discount rate and the full price as it is running? To do so, we need to make the computer pause, and wait for values to be typed in from the keyboard. This is done by the INPUT statement.

The word, INPUT, is followed by a list of variables - one or more, and the computer will, when it reaches this point, pause, and wait for values of the variables to be typed in through the keyboard. If more than one variable is to be input, the user should separate them with commas. Usually the computer will print a question mark, or some other prompt.

Good programmers will also add some text to explain to the user what to do. This may be done with a PRINT statement just before the INPUT: here is an example, to calculate the sale price given the full price and discount as inputs.

```
10 REM 'OP' IS THE ORIGINAL PRICE, SP THE SALE PRICE.
15 PRINT "TYPE ORIGINAL PRICE, DISCOUNT PER CENT"
20 INPUT OP, DI
30 SP = OP *(1-DI/100)
40 PRINT OP,SP
50 END
```

Most dialects of BASIC allow you to combine the PRINT and INPUT statements by placing some text in the INPUT statement. This text is printed before the values are read from the keyboard. If this is allowed in your variation, the statements 15 and 20 in the program above could be combined:

```
20 INPUT"TYPE ORIGINAL PRICE, DISCOUNT PER CENT";OP,DI
```

(There may be minor punctuation differences between machines.)

If the user makes a mistake, and types in something other than a number, the computer will complain - saying"REENTER", "REDO" or something. If he doesn't type enough numbers, usually the computer will just sit and wait for the rest. There are other mistakes that can be made, too - making programs 'bulletproof', so that NOTHING the user can do will upset them, is a major task for software writers.

Strings

Thus far we have handled only numbers. Sure we have put messages on the screen, but that's all. But most computer applications need to handle words, letters, and other characters. For example, last time you got top score in a game, what did you record? Your name or initials, which involves letters. Of course, the word-processing program which I am using to write this article will handle letters and words, and so do all the programs which work with names, addresses, and so on.

A CHARACTER is any symbol which the computer can record internally. This includes letters, digits, punctuation marks, and sometimes other symbols - graphics. A STRING is any collection of characters "strung" together. Here are some examples: "BIT", "NAME:", " $%&'&%$ ", "123". To distinguish them in print we often enclose strings in quotes - sometimes we have to do that in a program as well.

Computers handle strings in just the same way as they handle numbers - by storing them in a variable. But the computer needs to be aware that the variable represents a string, rather than a number. We tell it so by putting a dollar sign on the end of its name - like G$, NP$ and so on. Usually you will find that the same name may be used WITH a dollar sign to represent a string, and WITHOUT one to represent a numerical value. But that is a sure way to cause confusion; don't do it!

Strings can be used in almost all the BASIC statements we have met so far: PRINT, LET, IF, INPUT. Here is a trivial example of inputting a string, comparing it with a string constant, and using it in an output statement:

```
10 INPUT "WHAT IS YOUR NAME";NM$
20 PRINT "HELLO"; NM$
30 IF NM$="GORDON" THEN X$= "NICE"
40 IF NM$ <> "GORDON" THEN X$="SILLY"
50 PRINT "THAT IS A ";X$;" NAME"
60 END
```

You shouldn't have any trouble following the operation of that terribly useful(!) program. Notice that strings in some statements need quotes around them, like "GORDON", "NICE", and "SILLY" here. If there were no quotes the computer may well think that they were identifiers (names of variables). Try to change the program to get both first and last names, and print them out in reverse order, as well as a message.

Strings cannot be used for arithmetic. Even if you have two strings which LOOK like numbers, say X$= "123" and Y$ = "34.8", you cannot multiply them together!

Continued on Page 34

Low cost farm computing offered

By PAUL CROOKS

A Timaru based company has taken the cost of farm computing by the horns and slashed it to under $1800.

For many this figure may raise questions about the quality of the hardware and software that the company, Primary Software Ltd, is using.

These doubts may be further heightened when it is revealed that the company's "Primesoft" programs run on the Commodore VIC computer, one of the lowest priced brands on the New Zealand market and previously much of the software available for it was recreational.

But the VIC is also becoming known for its versatility and the key to any computer is the quality of the software available to run on it.

In this regard Mr Doug Hanna, the man who wrote Primesoft's programs, maintains they are not only comparatively cheap but also produce necessary information for farmers in minutes compared to the hours previously required.

As well Mr Hanna, a Lincoln College graduate, has designed the programs so that even the most computer unintelligent farmer should be able to use thém after a minimum of instruction.

By the end of the year Primesoft will have three major program packs available; for dairy farmers, sheep and beef farmers, and horticulturalists. Each pack costs $495 while a crop module can be added to the first two (it comes as part of the horticulture pack) if required for $245 and a cash book program will also be available for $145.

To this must be added the cost of the computer hardware which is $1298 giving a total cost for most farmers of $1792. A television set (either the family set or a cheap black and white model) is also required to display the programs.

But what advantages do Primesoft's programs have for the farmer?

Firstly the program packs in fact contain five programs designed to be run in a certain sequence.

Taking the dairy pack as a typical example here is the information that each program in that pack is designed to provide.

- Calving Schedule: Calving dates for each cow and heifer after details of cow numbers, service dates, return dates and so on are entered.
- Annual financial forecast: Prepares a profit and loss for the year after physical and financial data are entered.
- Cash flow: Gives monthly income and expenses and bank account balance after allocating the yearly totals from the annual financial forecast across the different months.

Because of the increasing interest among farmers and the rapid growth of farm software, BITS & BYTES will feature a regular farm column from next month.

We will talk to farmers using different microcomputers on their experiences and review new farm computing products as they are released.

A lot of claims are being made about the advantages to farmers of computers — this column will give you the chance to keep abreast of developments and decide if they will aid the efficiency of your farm. Remember a subscription to BITS & BYTES costs only $8 a year.

- Dairy cow profitability margin: Gives four different profit margins rather than total profit. The margins are $/cow, $/stock unit, $/hectare, $/acre. Also included is a sensitivity analysis showing the effect of a 10 per cent change in milk yield and/or milk price.
- Dairy beef profitability margin: Similar to dairy cow above but profit is also expressed as $/beast/week and the sensitivity analysis can measure any percentage change in the purchase and selling price.

The real advantage of computer power can be seen in all these programs if the farmer wants to look at different options or combinations.

For example if the initial financial forecast shows an unhealthy situation at the end of the year, the farmer may want to reduce development expenditure. When this figure is changed all the other figures it affects (such as total expenditure and the profit or loss) are automatically updated in a second.

"The farmer can look at dozens of different combinations that would take hours by hand," said Mr Hanna.

All the programs are menu drive (a list of options appears on the screen and the operator selects one), and a series of one letter commands (for example press 'C' to continue, 'T' to go to the top of the screen and 'M' to return to the menu) allow the operator to quickly and easily move about a program.

If the operator does become lost for any reason each program has a menu relocation number which when typed in takes the operator back to the reassuring sight of the menu.

All the programs use familiar farm headings and terminology suggested by the other two partners in Primary Software Ltd, Timaru farm consultants Mr Eric Etwell and Mr John Gillespie.

The manual provided with the programs is comprehensive, well laid-out and above all understandable.

There are disadvantages with the Primesoft approach. Most of them centre round the use of cassettes to store the programs and data and the amount of RAM available on the VIC.

Mr Hanna has devised several techniques to lessen the major drawback of cassettes - the time it takes to find programs and data.

Each of the five programs in a pack comes on a separate C15 cassette and is stored at the beginning of the tape, making it easy to find.

For storage of data another five C15 cassettes are provided and Mr Hanna recommends that only one group of information or file be stored on each side of the cassette, again at the beginning of the tape.

It still takes time to load and unload the program and data into the computer (up to four minutes each) but each of Primesoft's programs can fit into the fast working RAM of the VIC (no mean programming feat) so slow cassette searches of data are not required.

But the limited size of the VIC's RAM means the programs can't be linked together so, for example, the totals from the financial forecast can't be automatically transferred to the cash flow program. They have to be written down (Primesoft supplies stationery which matches the screen display of its programs so that hard copy can be produced by hand without the necessity for a printer) and then typed in again after the cash flow program is loaded.

The screen display of the VIC is another disadvantage. It is only 22 characters wide by 23 lines which can make the figures and words cramped. Nevertheless it is something the operator would become used to.

As can be seen most of the disadvantages relate to convenience and this factor has to be weighed against the advantages of cost, usefulness and ease of use outlined earlier.

One thing Mr Hanna is careful to stress is that Primesoft's programs won't prepare a farmer's accounts mainly because of the many different tax factors.

"They won't replace your accountant but they should help you relate to that person better."

So far only a few of the first Primesoft package available, the dairy pack, have been sold, but the company has and will be holding seminars around the country to familiarise farmers with its products.

Primesoft also offers financial planning systems on the more powerful and expensive Commodore 8096 computer.

Home grown farm software

A New Zealand based and developed system arising from pre-computer farm management practice would be a reasonable description of the Farm Business Management System of a Christchurch firm.

Farm Plan (N.Z.), Ltd, not to be confused with FARMPLAN, has developed its Farm Business Management System over the last three years. The origins are in a manual system used by a farm management consultancy service in North Canterbury.

The software is a versatile range of programs for CP/M computers, with a minimum of 64K and unlike some of its competitors, says the principal, Mr H. A. Lissaman, it has been designed by people fully conversant with New Zealand farming conditions.

The firm's software encopasses the following:

- Sheep farming (stud and grade).
- Beef cattle farming (stud and grade).
- Dairy farming (seasonal supply and town milk).
- Deer farming.
- Horse studs.
- Cash cropping.
- Horticultural units (stand alone or integrated).
- Ancillary, off/on farm, business activities.
- Mixed enterprises including any combination or multiples of agricultural, horticultural or business enterprises. (The FBMS is being used on a farming operation covering three separate units plus the complete entity combined).
- Use as a cash book for non-farming activities, i.e. secretary/treasurer of local or district organisation.

Mr Lissaman was himself a farmer for 13 years, and had been an adviser in farm management at Lincoln College before going into private practice.

He has been closely associated with the development of a farm computing group at Ellesmere, near Christchurch and commends the group approach to all farmers interested in microcomputing.

"It is my belief that if farmers were to join a group they would learn to explore and use more powerful software," he says.

The farm package of Farm Plan (N.Z.) Ltd, is being used by farmers, farm advisers and accountants. The price is $2500, and the package is fully maintained and is expandable.

"Our Farm Business Management System has been designed for small business applications, the definition of small being the number of transactions recorded throughout the year, as distinct from the size of those transactions or the total turnover for the year. Farming falls into this category," Mr Lissaman says, adding that the software can handle large records such as paddock and farm files over a long period.

The system encompasses a farm record book for manual record keeping, preparation of forecast budgets, and production estimates, as well as providing a filing system for computer printouts.

"The advantages of this have been well demonstrated in my consultancy practice, which uses the record book as the focal point of the farm business management operation," says Mr Lissaman.

The computer system provides for planning and the preparation of projected estimates, the input of actual transactions, data, and the compilation of reports. Some programs can be excluded; for example the dairying applications would not be needed by a sheep or beef farmer.

The third part of the package is made up of instruction manuals.

Programs available for the Farm Business Management System are:

- Stock records.
- Dairy farm records
- Paddock records
- Feed budgets
- Non livestock-estimates
- Actual transactions
- Production and valuation reports
- Monthly cash flow reports.
- Animal breeding program
- Analytical program
- Farm Diary and help feature
- Word processing systems
- Data handling programs
- Spread sheet programs
- Summary and future programs

But Mr Lissaman emphasises that farmers can add their individual record requirements. They can use the software to keep records on anything involving quantity, weight, and value over time (days, years, or multiples of these). "They might like to keep rainfall records or records of tree growth," he says. "This is a user friendly package which can meet their requirements."

Mr Lissaman emphasises that the system is integrated: one piece of data can be used to express its effects in a variety of reports.

Cutting costs - one advantage of software package

A potential halving of accountant's fees - that is one of the benefits to farmers claimed for the software package FARMPLAN, which runs on Apple microcomputers.

Initially developed in Britain, FARMPLAN has since been modified for use in a number of countries including Australia and it is that version of the package that is now being offered for sale in New Zealand by Rural Computer Systems of Christchurch.

But one of the partners in Rural Computer Systems, Mr Alistair Burbury, says that the package is very versatile and farmers should have no problems adapting it to New Zealand conditions.

For example the financial management programs, one of eight individually priced program packages that make up the total FARMPLAN offering, can accommodate up to 10,000 codes determined by the user. So the farmer can use standard accounting codes or any number of variations to suit the farm's particular requirements.

The other seven program packages are:

- Ram Management
- Dairy herd management
- Beef cattle management
- Pig breeding management
- Crop management
- Agricalc
- Dataplan

Each of these packages can contain up to three separate programs stored on different disks.

For example, the Financial Management package has three programs. The first is for setting-up the codes for each account (unless a change is needed this only has to be done once), the second is for data entry (entering the transactions from the latest bank statement), and the third is for printing out reports such as gross margins, profit and loss and balance sheet.

Apart from Agricalc and Dataplan, the other packages follow a similar format with programs for set-up, data entry, and reports (although some are combined on the same disk).

It is the Financial Management package that Mr Burbury feels could halve accountant's fees for farmers through its audit trail facility.

The farmer enters details of payments and receipts (account code, item description, amount, and so on) and the package automatically updates the bank balance and at the end of the month a print-out of the different accounts and trial balance can be obtained (a purchase or sales ledger is also included for those who require detailed creditor or debtor control).

The accountant still has to finalise the accounts (taking into account taxation, depreciation, and so on) but that person doesn't have to prepare them to the trial balance stage.

To obtain a comparison of actual figures to budgeted figures the farmer only has to load in the budget disk.

The Ram, Dairy Herd, Beef Cattle, Pig Breeding, and Crop Management packages deal more with physical data and each can produce a number of different reports at "the touch of a button".

For example, the Dairy herd package, once data entry has been completed, automatically prints out two reports. The action reminder list is printed weekly and lists all events such as calvings, first services and so on due to occur in any period while the milk recording summary

Continued on Page 22

Most farmers believe micros will help...

SELWYN ARROW *continues his series on microcomputers and software in New Zealand with a look at how farmers are adapting to microcomputers.*

New Zealand farmers are beginning to realise the benefits that microcomputers can bring as a management aid, just as they are doing in many other areas of business.

Interest is high, but it is difficult to estimate just how many farms have access to a microcomputer. The number would be in the order of 200, and expected to double every one or two years.

Some farmers have formed syndicates to provide joint use of both microcomputer and software. In the northern region a farm secretary regularly calls from farm to farm with a microcomputer to update cash books.

Sharing the resources of a microcomputer certainly cuts down on costs, but suffers from the major disadvantage of lack of immediacy of usage.

Where the microcomputer is most needed, of course, is in the farmer's own office: available at any time it is required. Sooner or later then the farmer will recognise the need and value of buying his own machine for use just when he needs it.

If just 5 per cent of New Zealand farmers and horticulturists bought microcomputers about 2000 people would be looking for a whole range of suitable software.

A recent national postal survey on record-keeping practices and microcomputer interest showed that about 60 per cent of farmers believe micros will be of some use or better, though must have made this judgment without the benefit of "hands-on" experience. Some 50 per cent indicated they are likely to buy one sooner or later.

Obtaining this initial "hands-on" experience to give a better understanding of what a micro can do, before acquiring one is a problem for many.

As in most businesses not many farmers can afford to rush out and buy even a small microcomputer system just to practise on, although a few have done so as a hobby.

These hobbyists, and others who have already taken the plunge businesswise, quite often become the focus of local attention from those who are considering buying a system.

In some areas the local high school or technical institute runs evening classes on using microcomputers. If these machines are compatible with available agricultural software they could become a local area resource, especially as local fund raising is the only way schools can acquire a microcomputer.

It is to be hoped that we may even see "farming uses for microcomputers" included in evening classes in the near future at such schools.

One group of farmers near Christchurch have recently co-operated with a new high school in their district. They raised funds to buy a microcomputer and it is now available for their own use in the evenings.

A common problem facing a farmer who wants to learn more about computing is finding someone who already has a micro, or more difficult still, finding someone who can answer technical queries about the use and programming of the machine they have in mind.

One possibility is through local or national computer clubs and user groups.

By finding and making their questions known to them, it is hoped they should be able to gain access to a pool of such expertise, either in their own locality or via the post or telephone.

Look before you leap

It can not be emphasised too often that people wanting to get into the computing age must consider their particular business requirements, and (of utmost importance) whether suitable software is available to do their job or jobs, before looking at the machine options.

A typical case is where an inquiry is received that goes something like this, "I've just bought a brand X micro - where do I get agricultural software that will work on it?" The cart is well and truly before the horse.

Farmers who own microcomputers generally fit into either of two categories.

The first are personal systems users. They have outlaid about $2000 for a microcomputer, visual display unit, and probably a cassette tape recorder. These machines do not have large memories and are limited in what they can be used for, mainly because of their slow data storage system.

A typical example in this category would be a System 80 used for keeping a diary, records of paddock histories, crop-return projections, feed budgeting, and small herd records.

Such a system would initially be used to replace the existing farm notebook. Then, with the proper software and diligent use, it begins to keep better records, enabling more detailed and sophisticated cross referenced information to be on call when needed.

Do-it-yourselfers

Several farmers have bought their machines for other reasons than just farm management and are happy to experiment and write their own programs or modify existing ones for their own uses. This suits the dedicated do-it-yourselfer, which, of course, is the image projected of the typical New Zealand farmer.

The second category would be classed as small business system users. Costing about $6000 to $10,000, these systems are usually complete with printer and would definitely include one or more disk drives. The most common operating system seems to be CP/M. Such a system would quite often be regarded as a farm management type implement.

Use with such a system is open ended, with plenty of memory on call and a wider range of software becoming available all the time. Some typical uses in the latter category are: keeping detailed stud breeding performance and paddock records; feed allocation; gross margin programs for crops, sheep and cattle; wages; forecast budgeting; and financial programs to help make capital decisions.

The availability of suitable programs for the job required, of course, determines the total usefulness of any system.

As suitable software is not usually included in the purchase price of a micro, to become a going concern, a suitable range of programs can usually cost anything from a half to as much again as the original cost of the machine itself.

Computers can't do everthing

Many farmers have read articles or heard comments that have perhaps been prepared by someone with very rosy tinted spectacles firmly fixed on their nose. While computers can be an invaluable aid to management, they (or rather the software) cannot always do exactly what you want.

Everyone would like to enter the data in exactly the form they have it and obtain an answer or report that exactly matches the nature of the problem being considered. Because, however, it is not possible (at this stage anyway) to prepare standard sets of computer instructions which fit in with everyone's detailed requirements, ingenuity and compromise is sometimes necessary.

Consequently, data may have to be put in a different form and results modified or extrapolated from in some way. In some cases, using a micro may not even be the most appropriate way to tackle the problem at all.

One answer is of course to find someone knowledgeable enough in both programming and the subject in hand to write suitable software from scratch or convert an existing program to obtain the desired effects.

Who better to do that then the farmer himself. So long as he has the time, interest and experience to accomplish it, this has proved to be very feasible in quite a few areas.

Software becoming available

Quite a few programs are available that were written or modified by farmers themselves and a lot of these are now becoming available to other farmers from a variety of sources.

Many programs are available from overseas, especially for those with larger disk-based systems, but there is a lot of work required in conversion to allow for the difference between New Zealand farming conditions and that of Australia, America and Britain.

Typical of such is Farmplan, this runs on the Apple computer and originated in Britain. It was consequently converted for Australian farming and is now modified and available to suit New Zealand conditions.

Of course many of the 'standard' spreadsheet type business programs such as Visicalc, Calcstar, etc, are directly useable for agricultural business.

Cereals to software

One organisation that has been specifically set up to create and supply microcomputer based management aids to the New Zealand farmer is the Kellogg Farm Management Unit at Lincoln College. The initial team of four professionals and several casual staff was started in 1980 with funding from the American W. K. Kellogg Foundation.

The unit has been active in promoting an awareness of the capabilities and pitfalls of these machines at agricultural shows and field days, various national conferences, courses run by technical institutes, and also through its own workshops.

Programs are currently being developed and written at the Kellogg Unit in MBASIC, for operation under CP/M on North Star hardware, as well as for implementation on TRS 80 model 2 and 3 systems. They are also being modified to run on a variety of machines, predominantly those with the CP/M operating system. Some other machines which the unit is working with are Apple, Canon and Sharp.

Most programs require magnetic disk storage. Memory requirements vary but most require a minimum of 25K bytes being available to the application program.

Once tested in the field, the programs will then be available on a cost basis to farmers. They are also available to farming bureaux at double the cost to farmers.

The unit hopes to eventually become self supporting by the time the grant runs out in a few years.

The initial emphasis from the unit is being placed on the use of on-farm microcomputers.

Computer controlled grazing

Computers are invaluable for farmers concerned with long-range planning, according to a Ministry of Agriculture and Fisheries farm adviser, Mr Keith Milligan.

Many farmers are now using some form of controlled grazing system. This involves more than just grazing management, he says; it is a fully integrated farming system.

Many factors are taken into account when using the system: liveweight gains, liveweight targets, pre and post-grazing dry matter levels, and feed requirements must be monitored.

Farmers using controlled grazing will also be aware that stocking rates at particular times of the year can be calculated on the basis of feed in an area, the daily feed requirement, and likely pasture growth rates.

A logical extension of this is to feed budget the whole farm. This mathematical exercise would be daunting for most people, Mr Milligan says, but the possibility of using computers makes feed budgeting more feasible.

"The skills needed for using a feed-budgeting program are no different from those already used by the farmer familiar with a controlled grazing system," he says.

These skills include:

- Visual assessment of pasture dry matter levels before grazing;
- Knowledge of liveweight, and liveweight gain targets at any time of the year.
- Knowledge of expected pasture growth rates. (In Hawke's Bay there are five years' data on 25 sites throughout the region that can be used as a guide.) Environmental conditions should be used to modify average expected pasture growth rates.
- A farm map with reasonably accurate paddock areas, along with the best direction of stock flow.

He says that apart from the farm map and pasture dry matter levels, all the other information required (for example, mating, lambing, weaning or shearing dates, liveweight targets and stock numbers) is readily known.

If not, it can be stored in the computer (for example, pasture growth rates, stock feed requirements and residual dry matter requirements to achieve liveweight gain).

Using this information the computer can quickly tell the farmer whether what he wants to do is possible, Mr Milligan says.

"If a certain plan is not possible the computer can highlight whether area allocation changes could help. The information produced by the program can be quite specific, and tells things such as expected number of days that each class of stock can be left in individual paddocks."

Farmers wishing to find out more about their program should contact their local MAF office.

From page 19

lists the entire herd and shows for each cow the date calved, time in milk, recordings in the previous five weeks, together with percentage change and yield to date.

Other reports on individual cows, groups of cows, or the entire herd can be produced on demand as well as graphs comparing milk production last year with this year month by month.

But all these programs cover historic or current data what about forecasting the future?

This is where Agricalc comes in. Using the capabilities of the popular memory-calculator program, VisiCalc, a number of models or templates have been written for specific farming applications.

These include feed budgets, milk plan, labour requirements, farm budget, and the now almost essential (to obtain a bank overdraft) cash-flow forecast.

The latter allows a complete year's cash-flow forecast to be mapped out and the effect of any number of changes can be seen in seconds without the need for hours of tiresome re-calculation by hand.

Dataplan is a "computer library" program allowing the retrieval and storage of large amounts of information for example livestock pedigrees, paddock records, and so on.

All the FARMPLAN packages are menu driven so the user just chooses from a list of options or answers simple questions with one letter commands (such as 'Y' for yes 'R' for reject and so on). Another useful feature for computer novices is the program keeps the user informed about what it is doing. During blank periods comments such as, "The report is being sent to the printer", will appear on the screen.

The FARMPLAN packages are sophisticated and this is reflected in their price tags and the price of the hardware required to run them.

Prices range from $2250 for the Financial package through $1400 for the specific farm packages such as Dairy Herd, to $250 for Dataplan and the Agricalc templates (VisiCalc costs an extra $500).

The packages run on an Apple 2 plus (64K) microcomputer and altogether, with two disc drives, printer, monitor and so on, the computer hardware needed costs $7800.

A farmer from Waiau, Canterbury, Mr Burbury formed Rural Computer Systems, together with another farmer and an accountant, in August of this year after seeing a need for software to help farmers "bogged down with paperwork."

But Mr Burbury says that if a farmer is not a good record keeper then computers aren't going to help that person much.

The potential cost savings, ease of use, and wealth of information the FARMPLAN packages provide must be weighed up against the cost and how much information the individual farmer requires.

Rural Computer Systems also offer the Sord M23 microcomputer, for which Mr Burbury has written several farming applications in the easy-to-use PIPS language.

The A,B and now C of business software

By KERRY MARSHALL

So far, I have talked about the A and B parts of the ABC of business software.

A - for the Analysis and Investigation of the user requirements;

B - for the Backbone of the system - the programs.

Now we turn our attention to the last part, C - for completing the job.

Many developers of software mistakenly assume that once the programs are written, the job stops. This was an attitude frequently found in large computer installations, where the workload of outstanding jobs resulted in new projects being started, without adequate attention to the finishing touches I am about to outline.

There are three stepping stones to the successful completion of a system.

PRE-IMPLEMENTATION

During the development of the system, there are several things the developer should do to lay a proper foundation for success.

User Backing- This does not meant that the boss rolls his sleeves up and does all the work. Rather, proper allowance is made for the time that implementation will take.

The company may have to employ additional staff for some time, so that employees who are critical to the system, can give reasonable attention to the details, without having to also do their current jobs at the same time. This requires planning and a willingness to recognise the importance of time.

User Involvement - Now that the key people can devote time to the project, involvement should include the areas of design (outputs and inputs, procedures and controls), testing, and preliminary documentation.

It may also be necessary to provide computer familiarisation courses if the computer is a new concept for the organisation.

TESTING

This is crucial - all programs must initially be tested on their own to ensure they function. Then, the

Continued on next page

From previous page

complete set of programs should be tested - by the user.

User testing again requires planning and time. The essential steps are:

- assemble test data that is representative of reality, including known errors;
- manually calculate the control totals and values;
- run tests;
- check that all output and files agree with the previously established controls.

Some points worth watching at this stage are:

- handling error conditions
- large or small values and quantities
- complex calculations
- accumulated data.

DOCUMENTATION

The three types of documentation are:

1. System Notes

- Overview of the System Objectives Scope Constraints
- Flowchart of Programs
- Description of each Program
- File Layouts
- Report Samples/layouts
- Input Formats

This information is, of course, assembled as the system is developed.

2. Program Files

For each program assemble the following information:

- Program Specification
- Flowchart of the Program
- Any queries and their solution
- Latest listing
- Cross-reference list of variables if available
- Test Data showing correct operation.

3. User Documentation

As programs are written and tested, the programmer should prepare draft instructions for using the program. All inputs and error messages must be clearly set out.

Once testing begins, the user and the developer can assemble a complete manual, inserting system information as required.

The involvement of the user is essential for this stage. Most computer people can talk about null-entries, aborts, or control-c, but this needs to be expressed in terms the user can understand.

If user staff have been involved from the start, the manual can be written by them and will be used because they know and understand it.

The May, 1982 edition of Practical Computing has an editorial that I think all software developers should read. It discusses the reaction of a person not used to computers when they are confronted by a keyboard and screen.

The effect of seemingly simple instructions on a screen can be quite devastating to people not familiar with our environment. Make sure you build the confidence of the people through a series of steps that involves them in

- developing the system
- preparing the instructions
- testing the system.

If they are involved in all these stages, they will be more confident and competent - the result will be a useful system.

THE GREAT DAY

At last all is ready and the day for implementation dawns. You have laid the foundations; now to set the system alight.

But wait! There is something we have forgotten.

FILE CREATION

This is the process of setting up the masterfile(s) and can involve a considerable amount of time and effort.

Problems can arise from

- assembling the data (it may not be in a presentable format);
- making sure it is correct to start with (do not convert errors onto the new system if you can avoid it);
- after entering the data double check all the details (do not rely only on totals, check individual records);
- allow for dynamic data; that is data that is output from the final run of the old system and can only be converted at the last minute.

The key to successful conversion is planning - allow time to assemble and check the data, in addition to entering it.

UP, UP, AND AWAY?

Now the system is going, and in the hustle and bustle of settling in it is tempting to take the first opportunity to depart, clutching your final payment. Avoid the temptation to do this. Let the client use the system and then arrange to review the project.

The Post Implementation Review is an opportunity to

- review the project (cost, timetable, problems encountered);
- identify and discuss any continuing problems;
- compare the results with the objectives set back in those early days.

This review process enables you, the developer, to think about mistakes in your estimating and quoting procedures, so you can be more accurate in the future. It also provides the user with a chance to objectively assess the effectiveness of the system.

Using the objectives established at the start prevents unfounded criticism of your new system ("it doesn't prevent stock-outs," says the storeman about a system that records inventory movements).

WHAT NEXT

Now the system is running and you want to go further? I hope you have the basics from these articles and, to enable you to continue, I have compiled a list of books that I have found useful in the past. Text books can help, but you will find courses run at your local Technical Institute or Community College also provide more practical advice.

FOR THE USER:

"The Computer Survival Handbook", by Wooldridge and London.

"Accounting Information Systems", by Hooper and Page.

FOR THE ANALYST

"Basic Training in Systems Analysis", by Daniels and Yeates.

"Systems Analysis", by Andrew Parkin.

"Elements of Systems Analysis", by Gore and Stubbe.

Small business computing

-what to watch for and what to ask for

By JOHN J. VARGO

One of the main difficulties in the selection of a small business computer system is acquiring a system that will meet your needs over a reasonable life span. Another hurdle is uncovering the hidden costs in purchasing, developing and maintaining an in-house computer system (especially your first one). This article will cover these two topics

Last month's article discussed: the advantages and disadvantages of owning your own small business computer, how to tell if your business might be ready for computerization, potential uses of an in-house computer, and how to approach the Feasibility Study.

The Feasibility Study is a preliminary study that determines whether it would be feasible and beneficial to purchase a small business computer. It should also bring together the staff involved in the selection process.

Keeping in mind that the five elements of an information system are people, data, procedures, hardware and software, it is most important to form a team of the people in your organization who will eventually manage and maintain the system.

Part of the Study should include a five-year business plan, since the business will change during those five years and the installed system should be able to change with the business.

This five-year plan should include all the areas that impact the information flow in the business. Sales forecasts, cash flow budgets, purchase plans, production plans for inventory and pro forma financial statements should ideally be included in the plan.

The preparation of this plan serves two very important purposes.

First, it gives a picture of projected changes in the company's information needs, which enables you to design a system that will carry you through the intervening years without becoming obsolete.

Secondly it gets management involved in long range planning that might not otherwise be undertaken. Remember, when you have a manual system you can often afford the mistakes that come from lack of planning. With a computerized system, such mistakes can be very costly.

ESTIMATING THE RIGHT SIZE SYSTEM

The size and configuration of the computer system that you end up choosing may be anything from a single user system for $5000 - $15,000, to a multiuser system in the $15,000 - $100,000 range and even higher. How can you be sure that the system you choose will meet your needs?

With the information that was gathered during the feasibility study and preparation of a five year plan, it is possible to determine the appropriate number and size of the components for your system.

To illustrate the sizing process the following hypothetical information might be gathered during the feasibility study for an inventory control system.

- 4000 items of inventory
- 120 purchase orders each month
- 1800 separate items purchased each month
- 2100 individual sales each month (average of two items per sales invoice)
- 4 employees (two sales clerks, 1 stock clerk, and one shipping/receiving clerk)
- Work load is evenly spread throughout the month
- Business is expanding and all of the above data is expected to grow at the rate of 10% per year.

Based on this information what size of system would be needed?

The hardware components will have to include: a central processing unit (CPU) or network of microcomputers, disk drives for data storage, printers for invoices and reports, and VDUs (visual display units) for data entry and inquiry. First, the disk drives' size can be estimated as follows:

Number of Records x Characters (a character is an individual numeral, alphabetic character, punctuation mark or single space element) in each record +
Number of transactions per month x number of characters per transaction +
Size of Inventory Control Program in characters +
Size of the operating system in characters.

Adding all of these elements together will yield the absolute minimum size for disk drives. So:

Records	4,000 x	120 (assumed 120 characters per record)
Transactions	2,100 x	150 (assumed 75 characters /item, two items / invoice)
Program	35,000	
Operating system	15,000	
	845,000	

So the absolute minimum size is 845K bytes of storage on disk with

Continued on next page

From previous page

795k in data. Add to this a 10% growth rate over five years and you are up to nearly 1.5 megabytes of disk storage for just the inventory control application.

Naturally, you will want to allow additional storage for unexpected growth, changes etc. So two to three Mbytes would probably be the minimum five year configuration for this one application. It will be necessary to do a similar calculation for the other application you are considering, General Ledger, Debtors etc., then you will need to decide if you want all applications on one hard-disk system, or would you prefer using separate floppy disks for each application.

A similar approach should be taken with each of the hardware components as follows:

VDUs - approximate time to handle one transaction times total number of transactions per day divided by total time available in a working day. For example: 10 min./Trans. x 105 trans./day ÷ 480 min./ day. This example, which assumes that it takes 10 minutes to perform a transaction, 105 transactions per day, and 8 hours at 60 minutes per day, yields 2.19 VDUs. So the minimum number of VDUs is three, to which you would want to add one for backup and another for peak periods. Therefore, you would need four or five VDUs and room for expansion if your business is going to grow.

PRINTERs- the number of printers needed will be determined by the maximum output needed at any given time. For example, if all purchase orders were to be placed at the same time each month (unlikely) how much time would it take to print them all?

120 purchase orders x 1600 char./p.o. ÷ 80 CPS = 2400 sec. If we assume 80 characters per line, the average purchase order has 20 lines, and our printer has a print speed of 80 Characters Per Second (CPS), then it will take 2400 seconds or 40 minutes to print all of the purchase orders for a month. One printer will certainly handle that load. Consider also invoices being printed at the sales counter. With potentially four VDUs working at once what would be the maximum wait to print a sales invoice? If the typical invoice has ten lines of 80 char. then an 80CPS printer will take ten seconds to print an invoice. Maximum waiting time would be forty seconds, or say a minute. That does not sound like much time, but try waiting that long with an impatient customer on the other side of the counter - then decide.

CPU- considerations should include RAM size, ROM available, processor speed, word size (8, 16 or 32 bit), expandability and maximum number of VDUs or workstations in the network.

When speaking of the sizing for software the primary concern is sufficient record and field space. Software that can only handle 3500 items of inventory would not be of much use when you have 4000 items currently in stock.

Response time is also of concern since it might not be satisfactory to have your customers waiting five minutes while the computer searched for the item of stock he wished to purchase. The response time is a factor of the software as well as the processor speed, disk access time and other factors, and the best way to test it is to try it on the vendor's equipment.

THE REAL COST OF THE SYSTEM

Having matched your business needs with the type, size, and quantity of system components it is necessary to have a closer look at the cost of the system.

A salesman may quote a specific price as the "cost" of a system. But the full initial investment together with the ongoing expenses will add up to an entirely different figure.

The true, full cost must be known if you are to make an informed decision in your selection of a small business computer. Having discovered the true cost you may even decide that you cannot justify computerisation.

Here is a list of the major elements of cost to be considered:

- Hardware (CPU, disk drives, printers, VDUs if separate from CPU, tape drive or other disk for backup, additional memory etc.)
- Software (Operating system, compilers, programming languages, application packages, custom or modified software)
- Communications (if you will need to communicate with other remote locations, if you will have a network of micro-computers, or for any reason will need to transmit data over the telephone system via modem)
- Hardware Maintenance (either a maintenance contract at a fixed price per year covering all parts and labour, or hourly rate plus parts)
- Software Maintenance (for changes and improvements, as well as fixing bugs that appear)
- Supplies (paper and ribbons for the printer, invoice forms, mailing labels, floppy disks, etc)
- Consulting Fees (if you decide to hire a consultant to help in your evaluation and selection)
- Insurance
- Training and Installation (this may be included in the purchase price of the hardware and software, but if not you will have to consider this additional cost)
- New Personnel (you may have to hire programmers, computer operators or other specially skilled people that you don't currently have)
- Implementation (including file conversion from manual to computer, and documentation of new system).

As you can see, it is not just a matter of buying your computer and throwing the switch. Many of the above costs may not be included in vendor proposals but you should most certainly include them in determining the viability of any given proposal, as well as the feasibility of the computerization project as a whole.

THE RIGHT COMBINATION OF HARDWARE, SOFTWARE, AND SERVICE

When purchasing a computer, especially your first one, restraint is required in not becoming so enthusiastic about the latest technology that you end up with hardware that is unserviceable and software that is either unavailable or unsuitable.

Slightly older technology coupled with proven software, and reliable maintenance may serve your needs better than the latest talking, 64bit microprocessor for which there is no software, and servicing is only available in the U.S. or Japan!

Generally speaking, you will be better off if you can contract for software, hardware and service with the same vendor, rather than multiple vendors. The primary reason for this is that people have a propensity for "passing the buck", and computer people are no different. If you have purchased your new system from one source there is less likelihood of this problem occurring.

Regarding the selection of software. You generally have three choices:

1. Off the shelf prepackaged software
2. Custom software (written especially for you)
3. Modified packaged software.

Off the shelf software will undoubtedly be the cheapest solution in the short run, but it may or may not suit your needs, so be careful; it can become very costly in the long run if you have to extensively modify the software. A major benefit of the pre-packaged variety is that it is often well proven and virtually bug free.

Next month's article will cover preparation of request for proposals, evaluating replies, estimating the financial benefits of computerization and the final selection of a vendor.

The custom software will usually be the most expensive solution to your software needs but it may yield the best result, especially if your needs are very specialised. A major drawback is that the software will inevitably have bugs in it

The custom approach is often worthwhile for a large system but very rarely can it be justified for the smaller systems with which we are concerned. The high cost of custom software may be remedied in the near future because of the appearance of what is sometimes referred to as fourth generation languages. These powerful, user-oriented languages may soon be able to bring custom software within the purchasing power of small system buyers.

The third choice, modified packaged software, will probably fall somewhere between the other two choices in price and in satisfying your specific needs.

Two applications programs for the ZX-81, by R. J. SPARKS, of Wellington. One is suitable for the basic 1K machine; the other requires the 16K RAM module.

A homework checker

"Homework" fits snugly into the 1K ZX-81, and allows a primary school pupil to check his or her homework, where this consists of exercises in basic arithmetic (addition, subtraction, multiplication, or division). It is probably most useful for children at the Standard 2 or 3 level.

A screen message first asks for the problem to be typed in, e.g. 235*47. The input is accepted as a character string, and is evaluated with the VAL function, so multiplication must be represented by an asterisk and division by a slash. After the NEW-LINE key is pressed a message asks for the answer obtained by the pupil. This number is compared with the result of evaluating the expression just entered, and a message indicating its correctness or otherwise is displayed. If the supplied answer is wrong, the pupil is advised to try the problem again.

Note that the right answer is not given by the program. Finally, a message asks if there is another problem to be checked; a YES answer re-runs the program, NO terminates it.

In the case of division exercises, answers are generally required in the form quotient and remainder, i.e. two numbers must be found. To cover this situation lines 52 to 56 check the input string for a slash (/). If one is found, the problem is one of division, and the program branches to a separate answer-checking section, lines 300 to 360, in which two numbers must be supplied, the quotient and remainder. Both must be correct for the answer to be accepted.

A point of notation in the listing is that PRINT strings shown enclosed in square brackets, e.g. line s90 and 110, should be typed in inverse characters.

```
******************
* HOMEWORK (1K) *
******************

 20 PRINT "TYPE THE PROBLEM ";
 30 INPUT P$
 35 PRINT P$
 40 PRINT "NOW YOUR ANSWER"
 52 FOR N=1 TO LEN P$
 54 IF P$(N)="/" THEN GOTO 300
 56 NEXT N
 58 INPUT ANS
 60 LET A=VAL P$
 70 PRINT P$;" = ";ANS
 80 IF ANS=A THEN GOTO 110
 90 PRINT AT 4,12;"[WRONG]";AT 6,4;
    "BETTER TRY THAT ONE AGAIN"
100 GOTO 120
110 PRINT AT 4,12;"[RIGHT]"
120 PRINT
130 PRINT "GOT ANOTHER ONE?"
140 INPUT B$
150 IF B$="NO" THEN GOTO 210
160 IF B$<>"YES" THEN GOTO 190
170 CLS
180 GOTO 20
190 PRINT "ANSWER YES OR NO"
200 GOTO 120
210 STOP
300 INPUT QU
310 INPUT RE
312 PRINT P$;" = ";QU;",";RE
315 LET DE=VAL P$(N+1 TO LEN P$)
320 LET A=VAL P$
330 LET Q=INT A
340 LET R=INT((A-Q)*DE+.5)
350 IF Q=QU AND R=RE THEN GOTO 110
360 GOTO 90
```

Cassette record

A problem with storing ZX81 programs on cassette is that there is no simple way to f nd out what is on a particular cassette unless you keep a record on paper, which of course cannot be found when wanted, and in any case is out of date. A solution is to make the first file on the cassette an index of all the programs currently on the tape, in the order in which they are stored.

INDEX, which requires the 16K RAM, keeps a record of programs on the tape, and can be updated as more programs are added. It can also be used to load a program from the tape, first checking the typed name against the names stored in its program list, so detecting typing errors before the entire cassette has been vainly scanned.

Program names can have up to 25 characters, and are stored in the string array P$. The maximum number of names in the list is set by the variable PMAX when the program is first RUN; thereafter PMAX cannot be changed. A typical value for a C-10 cassette would be about 5.

INDEX is self-starting, and when loaded displays the program list. The word, HOLD, in inverse characters at the bottom of the screen indicates that the computer is waiting for a keyboard character. Pressing any key results in the menu's being displayed, with options to list the program names, add a new name to the list, save the updated index, and load one of the programs in the list. The key-word in each option is displayed in inverse characters (denoted by square brackets in the listing). This word must be typed to select the corresponding item in the menu; e.g. ADD (NEW-LINE) allows a new name to be added to the list.

Note that LIST, SAVE, and LOAD are not the BASIC key-words, but must be typed in full.

It is essential that after new names are added INDEX is saved by means of the SAVE option; the program will remind you to rewind the cassette first. When the LOAD option is selected, you are asked to type the name of the program to be loaded. If this name corresponds to one stored in the index the go-ahead is given to start the cassette running; then type GO to start the loading sequence.

A useful modification for those who have cassette recorders fitted with tape counters would be to store and display the counter reading at the beginning of each program.

```
*******************************************
*           INDEX  (16K)                  *
*******************************************

   5 PRINT "MAXIMUM INDEX SIZE"
   7 INPUT PMAX
  10 DIM P$(PMAX,25)
  15 DIM L(PMAX)
  20 LET P=0
  25 IF P=0 THEN GOTO 80
  26 REM ***********************
  27 REM *LIST THE INDEX       *
  28 REM ***********************
  30 CLS
  40 FOR I=1 TO P
  50 PRINT I;TAB 5;P$(I)
  60 NEXT I
  70 PRINT AT 21,0;"[HOLD]"
  72 IF INKEY$="" THEN GOTO 72
  75 REM ***********************
  77 REM *PRINT THE MENU       *
  79 REM ***********************
  80 CLS
  90 PRINT "OPTIONS:"
 100 PRINT "[LIST] THE INDEX"
 110 PRINT "[ADD] A NEW PROGRAM"
 120 PRINT "[SAVE] THE INDEX"
 130 PRINT "[LOAD] A NAMED PROGRAM"
 135 INPUT O$
 140 IF O$<>"LIST" AND O$<>"ADD" AND O$<>"SAVE"
     AND O$<>"LOAD" THEN GOTO 80
 150 IF O$="LIST" THEN GOTO 25
 160 GOTO 100*(1+(O$="ADD")+2*(O$="SAVE")
     +3*(O$="LOAD"))
 193 REM ***********************
 195 REM *ADD A PROGRAM NAME TO*
 197 REM *THE INDEX            *
 199 REM ***********************
 200 IF P<PMAX THEN GOTO 210
 202 CLS
 204 PRINT "[INDEX FULL]"
 206 GOTO 40
 210 LET P=P+1
 215 PRINT
 220 PRINT "TYPE THE PROGRAM NAME"
 225 INPUT I$
 230 LET L(P)=LEN I$
 235 LET P$(P)=I$
 240 CLS
 250 GOTO 25
 295 REM ***********************
 297 REM * SAVE THE NEW INDEX *
 299 REM ***********************
 300 PRINT "REWIND CASSETTE;TYPE GO TO
          SAVE THE INDEX"
 310 INPUT G$
 320 IF G$<>"GO" THEN GOTO 310
 330 SAVE "INDEX"
 340 GOTO 25
 393 REM ***********************
 395 REM *LOAD A PROGRAM FROM  *
 397 REM *CASSETTE.            *
 399 REM ***********************
 400 PRINT "TYPE THE PROGRAM NAME"
 410 INPUT N$
 420 LET VP=0
 430 FOR J=1 TO P
 435 IF L(J)<>LEN N$ THEN GOTO 490
 440 FOR K=1 TO LEN N$
 450 IF N$(K)<>P$(J,K) THEN GOTO 490
 460 NEXT K
 470 LET VP=1
 480 GOTO 500
 490 NEXT J
 500 IF VP=1 THEN GOTO 530
 505 CLS
 510 PRINT "NAME ";N$;" NOT RECOGNIZED"
 520 GOTO 40
 530 PRINT "START CASSETTE;TYPE GO"
 540 INPUT G$
 550 IF G$="GO" THEN LOAD N$
 560 GOTO 540
9999 REM ***END OF PROGRAM***
```

By-passing the modulator

By E. J. BROWN

For those who have ZX81 microcomputers and wish to by-pass the modulator: first turn your ZX81 upside down and pry out the soft-rubber feet.

Underneath three of these you will find screws holding the case together. Remove these screws and one other and carefully take the case apart.

Take out the screws holding the printed circuit board and in the corner near the plugs and sockets you will find the portion of the board shown in the sketch. With care you can fit a phono or co-ax socket or a 3.5mm socket in the case between the present v.h.f. output and the earphone socket.

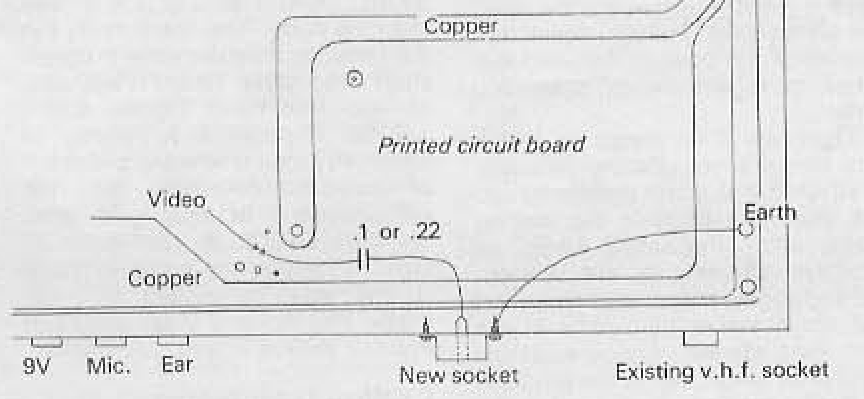
It is best to make a small first and enlarge it with a taper reamer.

The capacitor shown should be about .1.

The television set will have to be adapted but this is very easy and any knowledgeable person will be able to help. This still leaves the TV as a TV.

It is wise to switch to Channel 12 when using the set as a monitor. If the TV is a type without a transformer it may be wise to put a capacitor in the earth wire as well.

ZX81 video modification



How programs can run faster

By GORDON FINDLAY

Many thanks to all the readers who have written or rung me, with ideas or problems. I hope I have been able to help most of you. A frequent problem has been conflict between pieces of software - for example, between a machine-language routine POKEd into memory, and a monitor which is being used to debug the code! Be careful, too, that a lower-case driver occupies space which really is empty!

The most common question I have been asked is, "How can I make my program run faster, but still write it in BASIC", and this month's article takes off from this question. How can programs be made faster?

An obvious answer is to write the program in machine code. But this isn't always feasible. I tend to reserve machine code for places in which it is unavoidable - because BASIC would be in the way, or I need direct access to the hardware of the machine at a fairly fundamental level. This is, of course, the ultimate speed-up in terms of running time, but is certainly much more demanding of the programmer, and much slower to write and debug. As a personal example, I frequently use a memory search routine which I wrote first in BASIC, and then in machine code. It took about an hour to get the BASIC version going, but it was simply too slow. The machine code version took over 40 hours to get running. The conclusion - it wasn't worth it! Each time I use the utility I save at most four minutes by using the code version - which means that the utility will have to be used 600 times to regain the programming time!

There are four places in which time can be saved: the design stage, in which the program is planned and the algorithm chosen; the coding stage, when the actual BASIC or whatever statements are written; the debugging stage, when you find out the mistakes you made in the first two stages; and the actual execution stage, when the program is finally running. It doesn't make any sense to shave milliseconds off the running of a program, if it takes minutes or even hours at the writing or design stage.

There are some times, though, in which it is worth while to speed a program up - a slow game for example, or a very frequently used program which does a lot of work. There are times, though, in which little can be done. Programs which are input/output bound (input and/or output are the speed-determining factors), or spend time waiting for disks or printers, cannot be appreciably quickened by fiddling around with the code.

It is also false economy to write fast, but unintelligible, code which you have to spend hours to correct. With these provisos, here are some techniques.

Choose a better method

No amount of rewriting a bubble sort will make it faster than a heap-sort. Try to find a better way somewhere - in another program probably. A week or so ago I saw a program in which the author had gone to a lot of trouble to calculate an inverse sine with the best possible speed and accuracy. But how can his BASIC compete with the machine code in ROM? To get the inverse sine of X, just take ATN(X/SQR(1-X*X)).

Compile your program

A compiler is a sophisticated program which analyses a program in a high-level language, such as BASIC, and "translates" it into machine code. This takes time, but the resulting machine code program should run much faster. It will also, probably, be much bigger, and it will be impossible to debug in compiled form. There are a number of compilers available for disk systems, and at least one semi-compiler (ACCEL2) will work on tape. However, the compiler route is not very convenient in most cases, and in many cases programs must be altered in order to compile.

Coding techniques

If you have to speed up a BASIC program, FIRST GET IT WORKING! Then, and only then, worry about speed. Carefully examine the program, and see where the time is going. Where does the program spend most of the time? Is there a better way to code the bottleneck-areas? Will the improvement be significant enough to make it worth doing?

The bottlenecks are likely to be in inner-program loops, which are executed very frequently. It may be hard to find the loop - hidden in a mass of GOSUBs, IF-THEN-ELSEs and so on. If your examination finds a bottleneck area it may be worth speeding that area up - but not if the program spends only a small proportion of its time in this area.

Specific suggestions for speeding up inner loops are:

1. Use integer variables wherever possible. Here is an approximate table of times required for arithmetic operations in integer, single-precision, and double-precision arithmetic. All times are in milliseconds:

Operation	Integer	Single precision	Double precision
addition	2.9	3.6	4.4
subtraction	2.9	3.7	4.5
multiplication	3.5	4.2	8.8
division	-	7.6	54.2

(There is no time for integer division because integers are divided by converting to single-precision first.)

2. Use variables, rather than constants. This sounds like heresy to people used to compiled languages, but in an interpreted language, it is much quicker to use a variable than a constant. Compare these two pairs of statements:

```
100 X = 3.14159
110 Y = 3.14159/2

100 X = 3.14159
110 Y = X/2
```

The first pair requires the conversion of 3.14159 to floating-point form twice. This conversion can take up to one-third of a second - not much, unless it's in a loop!

3. Strength reduction. Reduce the strength of operations. Use

Continued on next page

addition, instead of multiplication, and multiplication instead of exponentiation. Never divide by two - multiply by 0.5 instead!

Here is another example of strength reduction: set every second element of an array to zero:

```
10 FOR I = 1 TO 1000
20 x(2*1) = 0
30 NEXT
10 FOR I = -2TO 2000 STEP 2
20 X(I) = 0
30 NEXT
```

The second loop replaces many multiplications with additions - in my timings, this reduced the required time by about 18 per cent.

4. Remove common subexpressions. Examine your code - are you doing the same calculations more than once? Here is an obvious example of a common subexpression:

```
100 FOR I = 1 TO 100
110 A(I) = SQR(X)
120 NEXT
```

Now it is unlikely that anyone needs 100 copies of the square root of X anyway, but why calculate it 100 times? It takes a long time to take a square root. Removing the common sub-expression gives

```
100 RX = SQRT(X):FOR I= 1
TO 100
110 A(I) = RX
120 NEXT
```

The saving in time: More than 80 per cent.

That was a trivial example of a common sub-expression. Here is an example which is less obvious, but even more worthy of attention:

```
10 FOR I = 1 to 100
20 X (I) = 0
30 NEXT I
40 FOR J = 1 TO 100
50 Y(I) = 0
60 NEXT I
```

This certainly clears both arrays to zeros, but wastes time. Here is a better method:

```
10 FOR I = 1 TO 100
20 X(I) = 0
30 Y(I) = 0
40 NEXT
```

The second is faster because it doesn't have to do all the computations involved in the looping twice. Blocking loops together is always worth while. Did you notice that in the second example I deliberately left out the variable in the NEXT statement? Loops run quicker that way.

Take nothing for granted: here is an example of a programmer who couldn't see the wood for the trees. This would be acceptable for a beginner, but in fact was found (admittedly in less obvious form) buried in a piece of software that I paid hard cash for:

```
300 IF A < 0 OR A > 99 THEN
GOTO 1000
310 FOR I = 0 TO 99
320 IF I = A THEN FREQ(I) =
FREQ(I) + 1
330 NEXT I
```

(Statement 1000 was an error routine.)

The program runs considerably faster now that the whole FOR loop has been replaced with:

```
310 FREQ(A) = FREQ(A) + 1
```

Anything you particularly want to know about? Drop me a line care of BITS & BYTES and I'll see what can be done.

HAND-HELD

This beats crossword puzzles

This program for the Casio FX702P with BASIC by C. W. NIGHY, of Auckland, is called 'Commandos'.

T sets the speed and about 5 is good for a start. The enemy, # , build up and feint and finally one at random makes a dart to attack. Your commandoes dash out and leave a remotely detonated bomb, *, on or near the enemy vehicle. If the bomb is successfully attached you have the opportunity of detonating it by remembering its position, 0 to 9. Hits and Misses are counted.

Mr Nighy, who describes himself as a middle-aged businessman/engineer, admits to becoming interested in computers to keep up with child computer users.

After owning a Casio FX601P using calculator language he bought a Casio FX702P

"I find this to be a most interesting and useful machine", he says. "Other than a limited memory capacity which restricts data storage it will cope with everything that interests me and has the advantage of portability. I can take it out on the boat with me instead of a book of crossword puzzles.

"With the cassette interface and the printer it will cope with a limited number of office accounts and the expected memory pack, when available, will extend the use of the Casio greatly."

```
 LIST #0
 3 VAC
 4 INP "T=",T
 5 WAIT 40
 6 PRT "HITS=";E;C
   SR 9;"MISSES=";
   D
 7 IF D+E=10 THEN
   #1
 8 WAIT T
10 A=INT (RAN#*10)
20 B=INT (RAN#*10)
21 PRT "    #"
22 PRT "   # #"
23 PRT "  # # #"
24 PRT " ## ### #"
25 PRT "##########
   "
30 PRT CSR A;"#"
31 PRT CSR B;"*"
32 IF A=B;PRT "DET
   ONATE"
33 IF A≠B THEN 10
70 C$=KEY:IF C$=""
    THEN 70
80 IF C$="0" THEN
   100
81 IF C$="1" THEN
   102
82 IF C$="2" THEN
   104
83 IF C$="3" THEN
   106
84 IF C$="4" THEN
   108
85 IF C$="5" THEN
   110
86 IF C$="6" THEN
   112
87 IF C$="7" THEN
   114
88 IF C$="8" THEN
   116
89 IF C$="9" THEN
   118
100 IF A=0 THEN 200
101 IF A≠0 THEN 210
102 IF A=1 THEN 200
103 IF A≠1 THEN 210
104 IF A=2 THEN 200
105 IF A≠2 THEN 210
106 IF A=3 THEN 200
107 IF A≠3 THEN 210
108 IF A=4 THEN 200
109 IF A≠4 THEN 210
110 IF A=5 THEN 200
111 IF A≠5 THEN 210
112 IF A=6 THEN 200
113 IF A≠6 THEN 210
114 IF A=7 THEN 200
115 IF A≠7 THEN 210
116 IF A=8 THEN 200
117 IF A≠8 THEN 210
118 IF A=9 THEN 200
119 IF A≠9 THEN 210
200 PRT "HIT":E=E+1
    :GOTO 5
210 PRT "MISS":D=D+
    1:GOTO 5

 LIST #1
10 PRT "HITS=";E;C
   SR 9;"MISSES=";
   D
```

Commodore Collections

Welcome to a collection of one-liners, tips and other useful bits of information for the Commodore user. All of these titbits work on my 4032 nine inch machine, but I won't guarantee that they will work on other Commodore machines or even on VICs. You're quite welcome to try them out, though!

Here is a cute one-liner to use as a demo:

```
10 Print "[rvs][spc][off][lft]"mid
$("[up][dwn][lft][rgt]", rnd (1)*4
+1,1"*[lft]";: for i = 1 to 40:
next: go to 10
```

With this program, "[lft]" is the cursor left key, "[off]" is reverse-off, etc., so it will fit on one line, even though it doesn't look like it. To make it go faster or slower, change or omit the delay loop "fori=1to40:next". You may also prefer this version typed in direct mode:

```
f0i=0to1stE0:?"[rvs][spc][lft]"
mI("[lft] [rgt] [up] [dwn]", rN(1)*4
+ 1,1)"*[lft]";:nE
```

This version can be typed in 49 characters using the shorthand method of entering BASIC commands. You may occasionally hit lines of a BASIC program that don't fit in 80 characters, even after abbreviating PRINT to "?" and omitting spaces around the line number. You may enter these lines by typing the first few letters of the keyword in lower case (the number will depend on the command) and then typing the next letter in upper case (or graphics). Example: VERIFY may be entered as 'vE', GOSUB as 'goS', etc.

I have noticed that many BASIC 4.0 owners are using DIRECTORY (or diR) instead of the shorter CATALOG (just two characters cA). For COMMAND-O users, directories may be obtained simply by typing Z (shift-z). Another disk command you may be unaware of is COPY "name*",D1 TO "*", D0. This copies all programs starting with the characters "name" from D1 to D0.

If you have joysticks, the following series of lines will let you detect and take action on them:

```
100 J=PEEK(59471) AND 15:
    GOSUB 900: REM left hand
    joystick
110 J=INT(PEEK(59471)/16):
    GOSUB 900: REM right
    hand joystick
120 more code . . .
890:
900 FIRE=0: X=0: Y=0
910 IF J=0 THEN FIRE=1:
    RETURN:REM fire button
    pressed.
920 IF (J AND 1) =0 THEN
    Y = -1: REM up
930 IF (J AND 2) = 0 THEN X =
-1: REM left
940 IF (J AND 4)=0 THEN Y=1:
    REM down
950 IF (J AND 8)=0 THEN X=1:
    REM right
960 RETURN
```

This code will handle any diagonal motion and set the variables X, Y and FIRE appropriately.

Here's a good little machine-language program that you can use for special effects in zap-pow! type programs. You only need to execute line 10 once.

```
10 for I=826 TO 858:READ
A:POKE I,A: NEXT
20 for I=1 TO 50:SYS
826:NEXT
100 DATA 169, 128, 141, 72,
3, 141, 77, 3, 160, 4, 162, 0,
189, 0, 128, 73, 128, 157, 0,
128,
110 DATA 232, 208, 245, 238,
72, 3, 238, 77, 3 136, 208,
234, 96
```

How many different hex to decimal and vice versa routines have you seen, each claiming to be the shortest? These routines beat all of them!

Hex To Decimal: Call with H$ containing the hex number, exits with decimal in D.

```
10 D=0:IFH$>""THENFORI=
1TOLEN(H$):A=ASC
(MID$(H$,I))-48:D=D*16+A+-
(A>9)*7:NEXT
```

Decimal to Hex: Call with H$="" & D containing decimal number, exits hex in H$.

```
10 IFDTHENA=INT(D/16):H$=
MID$("0123456789ABCDEF",
D+1-A*16,1)+H$:D=A:GOTO-
10
```

Did you know that it is possible to indent BASIC program lines with spaces? Simply follow the line number by any graphic character, some spaces, and another graphics character, and then type the line. If yours is a PET without a repeat key function built in (or if you can disable it), there is a way to tell if a cassette contains any programs. Simply press play, and then hold the '<' key down. If it starts repeating, there is something on the tape.

That's my ranting for this month. I'll see you next issue. Don't hesitate to write to me care of the magazine if you have any problem. Please send me your tips and information on any of the Commodore range and they will be included in this column.

— **Roger Altena**

Making a 'proper' keyboard for your Atari 400.

By. J. W. WHITE

I recently purchased an Atari 400 and quite rapidly got frustrated with its keyboard - especially when it came to longish programs, so I decided to replace it with a proper keyboard.

No problems I thought, I'll just find an Atari repair manual and get the circuit diagram for the 800s keyboard and wire in a new one the same in my 400.

Well, it was impossible to obtain a diagram of the keyboard, so it was a case of dismantle the computer, and work out how the 400s keys were wired. This took about 2½ hours with a multimeter.

Once I had finished working out how the wires (22) connected up - it was a "simple" task of wiring up the new keyboard, and connecting it to the Atari.

It took me about 4 hours and a little over 2 meters of wire - but amazingly enough it worked and I now have a 400 with an 800s keyboard. Here's how.

Items needed:

- 1 x Dick Smith (or similar) keyboard (if D.S. keyboard you'll need 1 extra key switch and key. Atari use 61 keys).
- 1 x 2.5m of wire (single core is best)
- solder
- About ⅓ metre of 22 strand (or there abouts) computer wire
- Around 3 hours free time
- Some experience soldering will help.

The first step is to open up the Atari - this is fairly easy.

1. Remove the ROM Cartridge (and leave door open)
2. Remove 4 screws on bottom of computer
3. Unhitch keyboard

First pull up plastic cover at edges - and locate and unclip pins from keyboard, then gently slide entire plastic cover off and forward. Once keyboard is free, the cover must be lifted up to free cartridge door.

4. Undo screw at rear of keyboard.

This should free keyboard to be flipped over. This should reveal 22 wires connecting the keyboard to the main PC board. Note on the rear of the keyboard where these connect, the top wire should have the No. 1 near it, and the bottom 22 under it. It is important that the new keyboard is connected the same way i.e. 1 to 1, 2 to 2 etc.

From here you can wire up the new keyboard and connect it up.

Now the first eight wires, and the 22 wires all go into the keyboard, and the rest, nine through to 21 all go out. Thus I have drawn this table to show how to connect up the keys.

On one contact follow the rows across the table and on the other follow the columns down the table.

Thus you will find the keys have 2 contacts, on one of them connect the rows, on the other connect the columns, and then connect all in row one to wire one in the old keyboard.

It sounds complicated but is quite simple really.

Notes:

1. Be careful soldering to the Atari keyboard's pins and wires, as the plastic around the wires deforms easily, making it difficult to plug the keyboard back in. It is easier, initially to solder the new keyboard onto the top contacts (between the keyboard and the system reset buttons, and take the wires through the top of the keyboard).

2. Wiring in and attaching a plug may prove best, as if you have to send the machine off for extra memory, the new keyboard won't be damaged, or its wiring.

thus:	9	10	11	12	13	14	15	16	17	18	19	20	21
1	BREAK	' 7		@ 8	(9	) 0	CLEAR <	INSERT >	BACK SPACE				
2		& 6		% 5	$ 4	# 3	" 2	! 1	ESC				
3		U		I	O	P	↑ | – =	↓ | | =	RETURN				
4		Y		T	R	E	W	Q	CLR/SET TAB				
5	CTRL		J	K	L	: ;	← | \ +	→ | ∧ *					
6			H	G	F	D	S	A	CAPS LOWR				
7		N	SPACE BAR	M	[,	] .	? /	/|\					
8	SHIFT (BOTH)			B	V	C	X	Z					
22										SYSTEM RESET	OPTION	SELECT	START

Commodore Collections

Welcome to a collection of one-liners, tips and other useful bits of information for the Commodore user. All of these titbits work on my 4032 nine inch machine, but I won't guarantee that they will work on other Commodore machines or even on VICs. You're quite welcome to try them out, though!

Here is a cute one-liner to use as a demo:

```
10 Print "[rvs][spc][off][lft]"mid
$("[up][dwn][lft] [rgt]", rnd (1)*4
+1,1"*[ lft]";: for i = 1 to 40:
next: go to 10
```

With this program, "[lft]" is the cursor left key, "[off]" is reverse-off, etc., so it will fit on one line, even though it doesn't look like it. To make it go faster or slower, change or omit the delay loop "fori=1to40:next". You may also prefer this version typed in direct mode:

```
f0i=0to1stE0:?"[rvs][spc][lft]"
mI("[lft] [rgt] [up] [dwn]", rN(1)*4
+ 1,1)"*[lft]";:nE
```

This version can be typed in 49 characters using the shorthand method of entering BASIC commands. You may occasionally hit lines of a BASIC program that don't fit in 80 characters, even after abbreviating PRINT to "?" and omitting spaces around the line number. You may enter these lines by typing the first few letters of the keyword in lower case (the number will depend on the command) and then typing the next letter in upper case (or graphics). Example: VERIFY may be entered as 'vE', GOSUB as 'goS', etc.

I have noticed that many BASIC 4.0 owners are using DIRECTORY (or diR) instead of the shorter CATALOG (just two characters cA). For COMMAND-O users, directories may be obtained simply by typing Z (shift-z). Another disk command you may be unaware of is COPY "name*",D1 TO "*", D0. This copies all programs starting with the characters "name" from D1 to D0.

If you have joysticks, the following series of lines will let you detect and take action on them:

```
100 J=PEEK(59471) AND 15:
    GOSUB 900: REM left hand
    joystick
110 J=INT(PEEK(59471)/16):
    GOSUB 900: REM right
    hand joystick
120 more code . . .
890:
900 FIRE=0: X=0: Y=0
910 IF J=0 THEN FIRE=1:
    RETURN:REM fire button
    pressed.
920 IF (J AND 1) =0 THEN
    Y = -1: REM up
930 IF (J AND 2) = 0 THEN X =
-1: REM left
940 IF (J AND 4)=0 THEN Y=1:
    REM down
950 IF (J AND 8)=0 THEN X=1:
    REM right
960 RETURN
```

This code will handle any diagonal motion and set the variables X, Y and FIRE appropriately.

Here's a good little machine-language program that you can use for special effects in zap-pow! type programs. You only need to execute line 10 once.

```
10 for I=826 TO 858:READ
A:POKE I,A: NEXT
20 for I=1 TO 50:SYS
826:NEXT
100 DATA 169, 128, 141, 72,
3, 141, 77, 3, 160, 4, 162, 0,
189, 0, 128, 73, 128, 157, 0,
128,
110 DATA 232 208, 245, 238,
72, 3, 238, 77, 3, 136, 208,
234, 96
```

How many different hex to decimal and vice versa routines have you seen, each claiming to be the shortest? These routines beat all of them!

Hex To Decimal: Call with H$ containing the hex number, exits with decimal in D.

```
10 D=0:IFH$> ""THENFORI=
1TOLEN(H$):A=ASC
(MID$(H$,I))-48:D=D*16+A+-
(A>9)*7:NEXT
```

Decimal to Hex: Call with H$="" & D containing decimal number, exits hex in H$.

```
10 IFDTHENA=INT(D/16):H$=
MID$("0123456789ABCDEF",
D+1-A*16,1)+H$:D=A:GOTO-
10
```

Did you know that it is possible to indent BASIC program lines with spaces? Simply follow the line number by any graphic character, some spaces, and another graphics character, and then type the line. If yours is a PET without a repeat key function built in (or if you can disable it), there is a way to tell if a cassette contains any programs. Simply press play, and then hold the '<' key down. If it starts repeating, there is something on the tape.

That's my ranting for this month. I'll see you next issue. Don't hesitate to write to me care of the magazine if you have any problem. Please send me your tips and information on any of the Commodore range and they will be included in this column.

— **Roger Altena**

Making a 'proper' keyboard for your Atari 400.

By. J. W. WHITE

I recently purchased an Atari 400 and quite rapidly got frustrated with its keyboard - especially when it came to longish programs, so I decided to replace it with a proper keyboard.

No problems I thought, I'll just find an Atari repair manual and get the circuit diagram for the 800s keyboard and wire in a new one the same in my 400.

Well, it was impossible to obtain a diagram of the keyboard, so it was a case of dismantle the computer, and work out how the 400s keys were wired. This took about 2½ hours with a multimeter.

Once I had finished working out how the wires (22) connected up - it was a "simple" task of wiring up the new keyboard, and connecting it to the Atari.

It took me about 4 hours and a little over 2 meters of wire - but amazingly enough it worked and I now have a 400 with an 800s keyboard. Here's how.

Items needed:

- 1 x Dick Smith (or similar) keyboard (if D.S. keyboard you'll need 1 extra key switch and key. Atari use 61 keys).
- 1 x 2.5m of wire (single core is best)
- solder
- About ⅓ metre of 22 strand (or there abouts) computer wire
- Around 3 hours free time
- Some experience soldering will help.

The first step is to open up the Atari - this is fairly easy.

1. Remove the ROM Cartridge (and leave door open)
2. Remove 4 screws on bottom of computer
3. Unhitch keyboard

First pull up plastic cover at edges - and locate and unclip pins from keyboard, then gently slide entire plastic cover off and forward. Once keyboard is free, the cover must be lifted up to free cartridge door.

4. Undo screw at rear of keyboard.

This should free keyboard to be flipped over. This should reveal 22 wires connecting the keyboard to the main PC board. Note on the rear of the keyboard where these connect, the top wire should have the No. 1 near it, and the bottom 22 under it. It is important that the new keyboard is connected the same way i.e. 1 to 1, 2 to 2 etc.

From here you can wire up the new keyboard and connect it up.

Now the first eight wires, and the 22 wires all go into the keyboard, and the rest, nine through to 21 all go out. Thus I have drawn this table to show how to connect up the keys.

On one contact follow the rows across the table and on the other follow the columns down the table.

Thus you will find the keys have 2 contacts, on one of them connect the rows, on the other connect the columns, and then connect all in row one to wire one in the old keyboard.

It sounds complicated but is quite simple really.

Notes:

1. Be careful soldering to the Atari keyboard's pins and wires, as the plastic around the wires deforms easily, making it difficult to plug the keyboard back in. It is easier, initially to solder the new keyboard onto the top contacts (between the keyboard and the system reset buttons, and take the wires through the top of the keyboard).
2. Wiring in and attaching a plug may prove best, as if you have to send the machine off for extra memory, the new keyboard won't be damaged, or its wiring.

<table>
<tr><th>thus:</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th></tr>
<tr><td>1</td><td>BREAK</td><td>' 7</td><td></td><td>@ 8</td><td>(9</td><td>) 0</td><td>CLEAR <</td><td>INSERT ></td><td>BACK SPACE</td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>& 6</td><td></td><td>% 5</td><td>$ 4</td><td># 3</td><td>" 2</td><td>! 1</td><td>ESC</td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>U</td><td></td><td>I</td><td>O</td><td>P</td><td>↑|− =</td><td>↓|| =</td><td>RETURN</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td>Y</td><td></td><td>T</td><td>R</td><td>E</td><td>W</td><td>Q</td><td>CLR/SET TAB</td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>CTRL</td><td></td><td>J</td><td>K</td><td>L</td><td>: ;</td><td>←|\ +</td><td>→|∧ *</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>H</td><td>G</td><td>F</td><td>D</td><td>S</td><td>A</td><td>CAPS LOWR</td><td></td><td></td><td></td><td></td></tr>
<tr><td>7</td><td></td><td>N</td><td>SPACE BAR</td><td>M</td><td>[,</td><td>] .</td><td>? /</td><td>/|\</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>SHIFT (BOTH)</td><td></td><td></td><td>B</td><td>V</td><td>C</td><td>X</td><td>Z</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>22</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>SYSTEM RESET</td><td>OPTION</td><td>SELECT</td><td>START</td></tr>
</table>

Dear Sir,

I very much enjoy your magazine and look forward to future issues.

But I do have some suggestions to make:

1. Could you please do some reviews on home computers as many people reading this magazine will be school children and some may be contemplating buying a computer.
2. A 'Problems' page could be useful for people who have problems that you or your readers could help them solve.

I do however, have a complaint.

I think that the competition time is not long enough for people who do not own their own computer. But congratulations on an otherwise excellent magazine.

Yours sincerely,
J. E. Dobbs.

Thank you for your helpful suggestions. From now on we expect to feature at least one review of a personal computer each month.

Readers' problems are currently being answered by our column correspondents, usually in their columns. But as the number grows we may start a separate problems page.

We agree the time allowed for the first competition was not long enough, and as you will have seen in our November issue, the entry date was extended. We will be running another competition soon and the same mistake won't be repeated.

The Editors.

SPEEDING IT UP

Sirs,

In my program "Games Evening" that you were kind enough to publish in "Bits & Bytes" (No. 1., September) there is a bit of clumsy though workable coding in the heart of a loop.

In attempting to calculate a number MODULO something else, I programmed as follows:

```
230 IF X< GN THEN X = X - GN:
  GOTO 230
```

This actually works, but slowly. In the cold light of print, I realised that the proper way to do this is:

```
230 LET X = X - (INT(X/GN)*GN)
235 IF X = 0 THEN X = 5
```

Line 235 merely reinserts a correction to make the string arrays match up.

One motto of programming is, "Make it work first, then make it elegant." Another motto, however, is, "Speed up the coding in the heart of loops." Line 230 is at the heart of the most time-delaying loop in my program. With 5 games and 20 couples, it originally took 101 seconds to compute the loop between lines 190 and 290 inclusive. The revised lines 230 and 235 gave the same results in just eight seconds!

I wish I could report that the original line 230 was just put in to give your readers a challenge. At least it makes me feel better to make the improvement myself.

Thanks,
Jay D. Mann
Christchurch

CONCERNING THE VIC

Dear Sirs,

Some inaccuracies have crept into some of your articles concerning Commodore's wonder machine - the VIC 20. In your first issue, a hint for the VIC user told how to change to lower case in a program using POKE 36869, 242. This works and works well until 8K or more of extra memory is added. Then the instruction becomes POKE 36869,192. However, a simpler solution, which will work on expanded and unexpanded VIC's alike, and is given in the manual incidentally, is to print CHR$(14) to change to lower case and CHR$(142) to change back.

In the second issue, Pip Forer suggested that Commodore had used misleading advertising and claimed that when Commodore said 16 colours they meant eight border and eight screen colours. Commodore is a reputable company and did not mislead anyone. When it said 16 colours, that is precisely what it meant; eight border and 16 screen colours.

The VIC is often run down by uninformed writers and described as little more than a toy or a video game. After using five computers, I would rate them in this order for ease of use:

1. Commodore VIC 20
2. Commodore CBM 8032
3. Sinclair ZX81
4. OSI Challenger 1P
5. Apple

I rate the Apple as a definite last because of the unnecessary complications involved in editing and programming. As a learning tool I believe the VIC outstrips all others in the field and at a fraction of the price of most of them. Gerrit Bahlman was right: "The VIC shouldn't be scoffed at."

Grant McLean,
Gore.

P.S. I think your magazine is excellent.

Carl McNeil: You left your own address off your letter. We would be interested to see some of your programs.

Basic BASIC

From Page 14

The main operation on strings is "concatenation'. That just means butting the two strings together. BASIC represents concatenation by a plus sign. Try to predict what this program will output:

```
10 X$ = "COMPUTERS"
20 Y$ = "ARE EASY"
30 Z$ = X$ + Y$
40 PRINT Z$
50 END
```

Did you say "COMPUTERS ARE EASY"? Look again - where did the first space come from? The actual output would be "COMPUTERSARE EASY". If you want a blank, put it in: try Z$ = X$ + " " + Y$

Even if a string looks like a number, the plus sign concatenates them, rather than adding them numerically - try it with X$ = "123" and Y$ = "9" - the result is "1239", not "131".

Frequently I find people having problems deciding whether to use a string or numerical variable. The choice is easy:

FUNDAMENTAL RULE: If the variable is ever anything other than a simple number, it must be stored as a string.

There are other operations - for finding the length of a string, and for taking parts of a string - but these must wait for another time.

UTILITIES: Fast routes to exciting effects

Over the last months we have looked at the sort of graphics a typical microcomputer can offer through its built-in performance. In particular last month we considered the sort of capabilities that built-in languages offered. This month and next we consider what you can add on to the basic machine to improve things. This is an important divide, because as soon as we talk about additional facilities we are talking about more money and usually, in the case of alternative languages, less users to share experiences with. There is an interesting phenomenon at work here which accentuates this divide. In general, the hardware capabilities of microcomputers are inversely related to their design age: the older ones are indisputably less powerful if analysed simply in hardware terms. However, the software base, in terms of programs and languages available, is directly related to age. The mature machines are still superior as problem-solving entities because the software-solution tools and extra peripherals are already to hand. We need to get a feel for what this means for graphics.

This month the focus is on graphics utilities. These are graphics aids that do for you some of the fancy things you might like to do yourself but are unable, or unwilling, to. Sometimes they run on their own and sometimes these utilities can be interfaced to be controlled by a BASIC program that the user writes. All in all the utilities give the user a greatly enhanced capability to draw intricate designs on the screen and undertake animation or 3-D perspective drawing. The demands these utilities place on the machine are quite high so the range of machines they can be implemented on is a little more limited than the field we have so far been ranging over. Also each utility plays to the strengths of the particular machines it was designed for. These vary so much it would be pretentious to pretend a knowledge of all utilities. To discuss this software I will deal with the machine I know best, the Apple II. It is fair to say that this machine is the best supported of the current microcomputers in terms of such software but increasingly similar utilities can be obtained for other machines.

The brief review contained here is concerned with three fundamental forms of graphics utilities: screen creators, shape creators and 3-D drawing utilities. Between them these three meet the needs of most users of current microcomputer technology.

Screen creators

These exist to allow you to produce a static disp ay screen. This might be a map display, like the Australian outline two months ago, or a birthday invitation like figure one. The screen may simply be used to convey a message or to act as the

By PIP FORER

background for an animation. The user can create a screen of graphics and then save the results to tape (waiting several hours to do so) or on to disk. He can then recall it at any time for further editing or use.

To produce a screen image usually requires certain component exercises. First, you have simple line drawing. Depending on your machine and utility this may be controlled by a pair of games paddles or by keyboard entry, but it is normally an interactive process involving a flashing cursor on the screen. Drawing a line, for instance, may entail moving the cursor to its desired start of the line, instructing the computer this is the start, moving to the other end of the line and then telling the computer to join the two points. Inevitably this involves also, at some point, being able to tell the computer what colour the line should be. Simple sketches can be made with such instructions.

One step up from this is the process of drawing certain geometric shapes. For instance, many utilities let you draw a square by specifying its top left and bottom right corners or a circle by defining its centre and then any point on its circumference. Most also let you draw the resultant shape as a "frame" or as a fully coloured-in shape.

The next sophistication is to give you better ways to add colour to the screen. This usually means two operations at least: paintbrushing and colour filling. Paintbrushing is essentially choosing a colour and a "brush". The brush can be any set of patterns but an example set of brushes is shown in Figure two. The user can select a brush and colour and, by moving the brush over the screen, colour in large or small areas of the screen with particular patterning effects. Colour filling is a different scheme widely used in Adventure games with graphics illustrations. Basically the colour fill takes any empty shape, such as a hexagon, and will fill it with a specified colour. This can be very useful. Most filling algorithms operate quite quickly, using machine-language routines. This means that a picture outline can be created in black and white and then colour added by filling the various facets of the image with colours. As indicated in my October article by creating colour mosaics of different pixels the apparent colours used can number many more than the standard machine colours available on most microcomputers.

Most screen creation utilities permit the user to place text and labels (usually multi-coloured) on to the screen and then when finished the whole image can be saved.

Continued on next page

From previous page

Forer on utilities

Good utilities offer some extra facilities such as the ability to shrink pictures, move parts of an image around the screen, filter out certain colours and other features. This really begins to offer sophistication in creating backdrops.

The final common feature of such utilities is their ability to recall already defined shapes and draw them on the screen. The men and spaceships in the invitation's frieze, for instance, were placed by this process. The user loads some pre-defined shapes, is allowed to move them around over the screen and then place them at certain points. All this raises the question of how you create the predefined shapes in the first place. This naturally leads on to the next (brief) section.

Shape creators

Shapes can be drawn on microcomputers in several ways as we have already discovered. The most common are character shapes and vector shapes. Two distinct sorts of utilities exist to let you create shapes. Some were originally designed to allow the creation of letters in different font styles. The most well known example of this is the Apple software, Higher Text, which has numerous fonts that allow the computer to write in Olde English, Greek, or even Sanscrit. Now a letter is just a shape and an alphabet of spaceships or map symbols can be created quite easily with such utilities and even used for animation purposes (by writing slightly different shapes in the same screen location at speed). However, since all letters are the same size these utilities have one disadvantage: they don't let you have shapes outside the regular character size. No long sea serpents or tall pine-tree symbols. In some utilities the shapes also have to conform to the limited grid that normal screen letters appear in.

The true shape creators allow far more flexibility (within the hardware system's own limitations) in terms of size and colour. They usually allow the user to create particular shapes with paddles or keystrokes and display them for the user as he creates the shapes. The various shapes can then be stored to tape or disk for recall and later use. Good utilities let you do this operation very quickly and permit editing and correction of previous results. Anyone who has tried to code shapes by hand on any machine will appreciate just how useful these utilities can be.

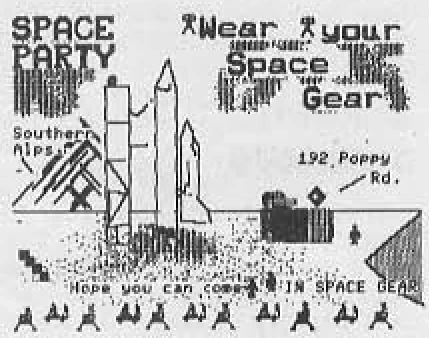


Figure 1: A party invitation created by a paddle-driven screen creation utility, Penguin Software's Complete Graphics System.

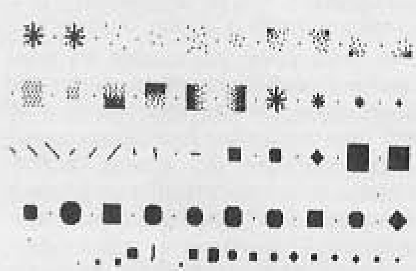


Figure 2: The brush set used for paintbrushing by the Penguin system.

With shapes and backgrounds created you can move quite easily on to animation by using the shapes against the background you have already created. This is quite nice but it is still flatland for most of your creations. Would you perhaps like to add a three dimensional graphics to your capabilities? If so then read on.

3-D Graphics Utilities

This is going to be a very brief review of a fascinating area which stands as a good example as why graphics utilities are valuable. Suppose you have an object that exists in three dimensions and you want to look at it on the computer. How can you do it? Firstly you simplify it if you are working on a microcomputer. You get rid of all the "solid" features and reduce it to a "wire-frame" representation. This can mean ambiguities and loss of colour and shading, but it is necessary short of a few tens of thousands of dollars for a larger machine. Then you can code this frame up into points and lines which have co-ordinates in three dimensions. For one point on a shed outline for instance X and Y might be the horizontal position and Z the height above ground of a particular angle of the shed (say a corner of the door). The outline consists of specifying which points are joined to which others by lines. When you have coded this data for all the lines in the object you have a data base of lines defined by their end points in three dimensions which together comprise the total object. All you need now is to get a program which lets you draw this set of lines onto a 2-dimensional screen and you can view your 3-D object with your microcomputer. With a suitable program you can make it move, spin it around and view it from different perspectives and you can even go inside it and look out.

The barrier standing between most people and the ability to do this is several hundred hours program development and an understanding of 3-dimensional trigonometry. The latter is not as great a barrier as one might first think. The former, however, is both inefficient and beyond most of us. That is where 3-D utilities come in. Some are more flexible than others. Some give true perspective others just scale objects crudely. One I know of lets you create 2-dimensional "panels" and then rotate them and glue them together into 3 dimensional objects. Another is specially designed to let you design and use space games with apparent perspective and up to 32 independent objects shooting around. Many are designed to let you interface your utility (which is in machine code) with an easier-to-program BASIC DRIVING PROGRAM. The utility draws the objects but the driver specifies which ones and where.

The results from these programs are quite encouraging. You cannot get (with any speed) the hidden lines removed (that is the lines you normally wouldn't see because they would be behind a solid surface). However, some nice results can be obtained. An example is the panel of pictures of my house obtained with one utility. I was experimenting with using this utility to give geographers a new way of examining readings from depth soundings. As it happened I experimented on my house as a first trial (it seemed a good object since I had a fair idea of what it looked like). The results in Figure 3 show the data base of lines displayed from various perspectives: a plan from above; coming up the drive and looking out from my sitting-room window. This particular utility allowed me to "snapshot" particular views (which is what I did here), or fly around the property with relatively smooth animation. The total time to get this set up (including experimenting with the new package) was one wet Sunday afternoon.

Of Utilities and Futilities

Re-inventing the wheel is great as a learning experience but doesn't solve problems or achieve goals efficiently. This is the lesson of graphics utilities. They can help you do things with graphics far more efficiently and most are sourced at relatively cheap prices. A good screen of graphics is a complex entity to create and very time consuming to do without aids. These aids exist and multiply severalfold the power at your fingertips for certain applications at minimal cost. However, graphics utilities are unique to each make of machine and some machines are far better supplied than others. In

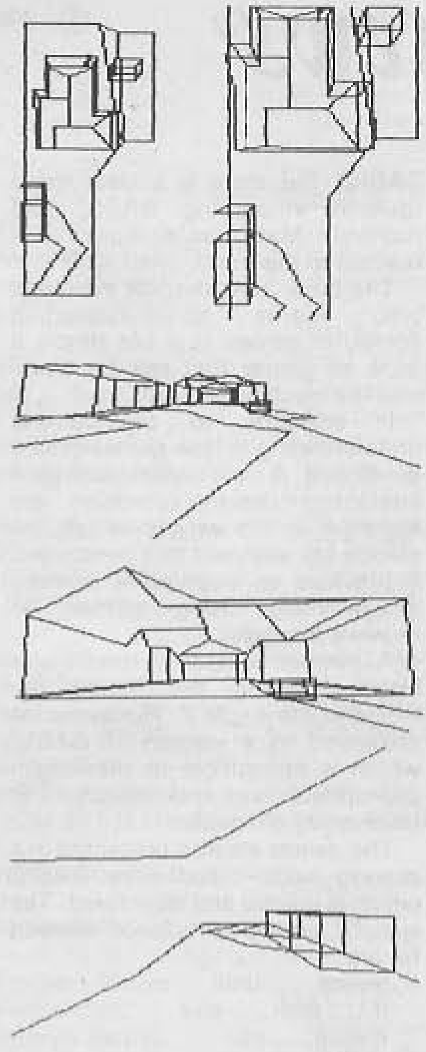


Figure 3: Views of the Forer house. Different perspectives of the same 3-D data base and displayed by the same graphics utility.

microcomputer graphics this software base, which is not an intrinsic part of evaluating the elementary machine, is an important feature of any inter-machine comparison If you want to develop a unique application or draw simple graphs this lesson is a minor concern for you. Otherwise it merits attention.

Utilities as described here are a first step in supercharging your graphics. The next one is a new language or new hardware add-ons. These will be the subject of the next article (February). - **Copyright: P. Forer.**

book reviews

Designing games in BASIC

"Inside Basic Games" by Richard Mateosian. Published by Sybex. 326 pages. $26.50. Reviewed by Gerrit Bahlman.

There has been continued debate about the value of the BASIC computer language. Some maintain that its simple interactive nature makes it an ideal tool for the new computer user. Others argue that it is limiting because it fails to encourage a structured approach to program design. While the debate rages on, it is certainly true that the number of BASIC dialects in existence appears to grow with each newly marketed computer.

Some verions of currently available BASIC strive to address the "not structured" critics by introducing if-then-else-repeat....until, recursive procedure calls and other structured language ploys. Needless to say some argue that the new dialect is no longer BASIC. But there is a clear trend towards structuring BASIC and Richard Mateosian's book has reacted to this trend.

The book is written for everyone who wants to understand computer games. It is not simply a book of games that can be typed into the machine and run.

It attempts to provide an understanding of how games can be developed. A clear understanding of interactive design principles are apparent in the way in which the games are analysed and presented. It provides an avenue for learning programming using games as relevant examples.

Allowance is made or three major microprocessors, the TRS80, the PET and the Apple II. Programs are presented in a version of BASIC which is compatible to them with appropriate flags and subscripts to resolve any difficulties.

The games are a so presented in a pseudo code called Free BASIC which is precise and structured. The specific control structures allowed for are:

repeat ..... until
If..... then..... else
If case..... else
while.

Using this pseudo code it is possible to discover the design fluency of structured languages such as Pascal. One chapter of the book is dedicated to free BASIC and its relationship to the BASIC used on the machine. All the games developed are paraphrased in free BASIC and as such provide excellent examples of the technique.

The games presented vary greatly in difficulty. Early chapters deal with arithmetic games, such as simple addition drills, guessing games such as Hangman, Time games which involve timing loops.

The theory of each style of game is well explained and the final examples of games are challenging. Alien life is a game which is partly graphic and partly a message transmission game.

The range of machinery to which this book is directed necessarily restricts the complexity of machine dependent graphics, but the attempt is made to allow the more common feature, to be employed. The other impressive factor of this book is that each program is given possible extension ideas. The level of difficulty is well graded so that beginners can accept those challenges and provide a personal involvement in writing or extending the games.

The only facet of the book which has failed to impress is the underuse of flow charts or structure diagrams.

MICRO CLUB CONTACTS

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

●●

NZ MICROCOMPUTER CLUB INC., P.O. Box 6210, Auckland. The Monthly Meeting is held the first Wednesday of each month at the VHF Clubrooms, Hazel Ave., Mt Roskill, from 7.30 p.m.

Visitors are also welcome to the Computer workshop in the Clubrooms, 10am – 5pm, on the Saturday following the above meeting.

The following user groups are part of the club. All meetings shown start 7.30pm at the VHF Clubroom.

APPLE USERS' GROUP: Bruce Given, 12 Irirangi Rd., One Tree Hill, Phone 667-720 (h).

ATARI MICROCOMPUTER USERS GROUP: Brian or Dean Yakas, Phone 8363 060 (h). Meetings: 1st Monday.

COMMODORE USERS' GROUP: Doug Miller, 18 Weldene Ave., Glenfield. Phone 444-9617 (h), 497-081 (w). Meetings: Third Wednesday.

CP/M USERS' GROUP: Kerry Koppert, 2/870 Dominion Rd., Balmoral. Phone 69-5355 (h).

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).

SINCLAIR USERS' GROUP: Doug Farmer, Phone 567-598 (h). Meetings: Fourth Wednesday.

SORCERER USERS' GROUP (NZ): Selwyn Arrow, phone 491-012 (h).

1802 USERS' GROUP: Brian Conquer, phone 655-984 (h).

2650 USERS' GROUP: Trevor Sheffield, phone 676-591 (h).

The above contacts can usually be found at NZ Microcomputer Club Meetings, or via P.O. Box 6210, Auckland.

Regular Meetings are:

MICRO CLUB, First Wednesday, plus an all day Computer Workshop the Saturday following, (10am – 5pm), all welcome.

ATARI MICROCOMPUTER USERS' GROUP: First Monday.

COMMODORE USERS' GROUP: Third Wednesday.

SINCLAIR USERS' GROUP: Fourth Wednesday. All meetings start at 7.30 pm at the VHF Clubrooms, at the end of Hazel Ave. (off Dominion Rd), Mt Roskill, Auckland.

Other active User Groups within the Club are:

APPLE, CP/M, DREAM 6800, SMALL BUSINESS, KIM, LNW, SORCERER, 1802 and 2650. They can all be contacted at club meetings or via NZ Microcomputer Club, P.O. Box 6210, Auckland.

Other Auckland-based groups:

ACES (Auckland Computer Education Society): Ray Clarke, 1 Dundas Pl., Henderson, Phone 836-9737 (h).

CMUG (Combined Microcomputer Users' Group): This is an association of Microcomputer Clubs, Groups, etc, formed to co-ordinate activities and to give a combined voice on topics concerning all micro users. Representation from all Clubs and Groups is welcomed to: CMUG C/- PO Box 6210, Auckland.

HP41C USERS' GROUP (Auckland): C/- Calculator Centre, P.O. Box 6044, Auckland: Grant Buchanan, 790-328 (w). Meets third Wednesday, 7pm, at Centre Computers, Great South Rd., Epsom.

NZ TRS-80 MICROCOMPUTER CLUB: Olaf Skarsholt, 203A Godley Rd., Titirangi. 817-8698 (h). Meets first Tuesday, VHF Clubrooms, Hazel Ave., Mt Roskill, Auckland.

OSI USERS' GROUP (Ak): Vince Martin-Smith, 44 Murdoch Rd., Grey Lynn, Auckland. Meets third Tuesday, VHF Clubrooms, Hazel Ave., Mt Roskill.

SYMPOOL (NZ SYM USER GROUP): J. Robertson, P.O. Box 580, Manurewa, Phone 266-2188 (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.

●●

ELECTRIC APPLE USERS' GROUP: Noel Bridgeman, P.O. Box 3105, Fitzroy, New Plymouth, Phone 80-216.

●●

HAWKE'S BAY MICROCOMPUTER USERS' GROUP: Bob Brady, Pirimai Pharmacy, Pirimai Plaza, Napier, Phone 439-016.

●●

TARANAKI MICROCOMPUTER SOCIETY: P.O. Box 7003, Bell Block, New Plymouth: Francis Slater, Phone 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.

●●

UPPER HUTT COMPUTER CLUB: Shane Doyle, 18 Holdworth Avenue, Upper Hutt. Phone 278-545. an all-machine club.

NZ SUPER 80 USERS GROUP: C/- Peanut Computers, 5 Dundee Pl., Chartwell, Wellington 4. Phone 791-172.

●●

Continued over

GLOSSARY

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: 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.

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.

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: 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 and Machine language.

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, or where a change can next be made.

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.

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.

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.

Hardware: The computer itself and peripheral machines for storing, reading in and printing out information.

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".

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

Modem: 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.

Megabyte (or Mb): Represents a million bytes.

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

Menu: 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.

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.

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.

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

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

RAM: 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: Read only memory. Any memory in which information or instructions have been permanently fixed.

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

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

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

From previous page

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 '80 USERS' GROUP: David Smith, P.O. Box 4118, Christchurch, Phone 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 796-100 (w).

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

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

NOTE: If your club or group is not listed, drop a line with the details to: Club Contacts, BITS & BYTES, Box 827, Christchurch.

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 ☐

Macro reading for Micro fans

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

NEW ZEALAND'S
PERSONAL COMPUTER MAGAZINE