VIC COMPUTING

February 1982 Vol. I Issue 3

DIY CHARACTERS

RHINO!

PROGRAMS FROM READERS

BUTTERFIELD ON THE INTERNALS

The VIC centre
It's the add-on to start with for your VIC-20

When it comes to looking at what the incredible VIC-20 has to offer, there's one cost-free add-on it will pay you to consider right from the start. It's called The VIC Centre, and you'll find it at the Adda Home Computers shop in West London. Part of the successful Adda Computers group, The VIC Centre provides the kind of service you'll not find anywhere else in the country.

Our business is dedicated towards providing you with a "one-stop" source for the VIC-20, VIC-20 peripherals, and VIC-20 software. On the basis of a very simple philosophy: providing a friendly, fast and comprehensive service for the world's most user-friendly and helpful personal computer.

With its magnificent sound and colour capabilities, the VIC-20 makes it easy to learn all about computing. And as your knowledge grows, we can supply the software and peripherals which give the VIC-20 system its unrivalled versatility in applications ranging from home budgeting and video games to business records and statistics.

All our customers benefit from our telephone technical advisory service and in-store repair facilities.

So if you want to get to the heart of what the VIC-20 system is all about, go straight to the centre—The VIC Centre.

154 Victoria Road, Acton, London W3 (near to North Acton tube station). Open Saturdays 10.00-17.00, Tuesdays to Fridays 10.00-18.00.

For up-to-date information on the VIC-20 system join our information service. Just complete the details on the FREEPOST coupon below and post it today, or telephone us on 01-992 9904.

- VIC 20 CASSETTE UNIT
- PRINTER
- TELEVISION OR MONITOR
- POWER PACK
- 3K/8K/16K MEMORY EXPANSION
- PLUG- IN PROGRAMS
- MODEM FOR TELEPHONE AND TELECOMPUTING
- SINGLE DISK DRIVE
- RF MODULATOR
- VIDEO CABLE
- JOYSTICK PADDLE LIGHTPEN
- 3K MEMORY EXPANDER
- EXPANSION MOTHER-BOARD
- PLUG- IN PROGRAM

Please put me on your mailing list for up-to-date information on the VIC-20 system.

Name

Address

The VIC Centre
Adda Home Computers Ltd.
FREEPOST
London W3 6BR

Vic COMPUTING

February 1982 **Vol. 1 Issue 3**

CONTENTS

Editor
Dennis Jarrett
Workers
Stacey Tanner, Barry Nicholson, Michael Burlington, Andy Bloom
Subs Manager
Annabel Hunt
Production
Rock Storm
Art and design
Karen the Black Douglas, Paul Hulme
Old master
Julian Allason

Telephone: 01-607 9489 (all departments)

Vic Computing is published six times a year by Paradox Group, 39-41 North Road, London N7.9DP. Annual subscription: £6 per year (UK), £IRL 8.50 (Eire), £9 (Europe), £16 (rest of world). Printing by Thamesmouth Printing Co Ltd, Southend-on-Sea, Essex; Phototypesetting by Carlinpoint Ltd, London WC2

All editorial contributions are welcome; material is accepted on an 'all rights' basis unless otherwise agreed. Enclose SAE if you want the return of original manuscripts and photographs. Please do not send listings: we only accept programs in cassette form.

News

JONATHON REYNOLDS

This is by way of an introduction: starting with this issue Vic Computing has a new editor, a new designer, and a new address — 39-41 North Road, London N7 9DP. We won't be altering the magazine's emphasis on the Vic, of course. But we do intend to enlarge its size, scope, coverage and readership. We've already commissioned over two dozen articles directly on the Vic (memory, the function keys, communications, Vics talking to Pets, reviews, programming tips and tweaks, and lots more). And since we aren't owned by Commodore, that should make for some no-holds-barred analysis.

Over the next year we want Vic Computing to become the prime example of a specialist single-subject magazine and a Bible for Vic users. To do that we're building up a good-looking team of contributors. But we need your input too: write and tell us what you want to read, offer us ideas for articles, and send us your programs. We want to hear from you!

Fortify yourself

You want 40 columns? You got 40 columns. And you can have an extra 32k of memory too, thrown in for free (well, sort of). And you can have it now (well, soon). And it will load and run programs developed for the 40-column Pets, of which there are several hundred ready and willing to go. What's more, the display looks good — for alphanumerics at least.

The 40-column conversion will cost £220 plus VAT. That's more than your Vic to be sure but by no means an outrageous total price for a 40-column 32KB microcomputer.

But there's a catch: you don't get Vic graphics to put in the 40 columns. Instead you can make do with Prestel's less than stunning visuals. Ah well.

The add-on hails from a new company called Beelines, which is located not a million miles from John Blackburn's B&B Computers. Now B&B is doing great business with Prestel adapters for micros, notably a plug-in converter that allows you to use the Pet's keyboard and screen to access Prestel. There's a Vic version of that one coming along. But meanwhile that expertise in displaying viewdata text and symbols has been combined with an eye to the main chance — Vic's disappointing display area, all that 40-column Pet software, the relatively low price of memory chips right now, and so on.

There's nothing wrong with seizing a marketing opportunity, of course. And Beelines plans an interesting approach to flogging its box (which, by the way, will be a flattish panel that sits underneath the Vic): Beelines will sell through Vic dealers in the ordinary way, but it will also operate a mail order department complete with Access and Barclaycard.

Will it fly? Beelines thinks so. "We're gearing up for production in the thousands" said An Anonymous Spokesperson. And what about the viewdata graphics, which certainly aren't as good as Vic's standard set? You can run two screens, said AAS, switching from the 32k-and-40-column mode to the standard 20-columns-and-however-much-RAM-you've-bought.

The thing exists only in prototype form at the moment, but we've been promised one for review ASAP. Meanwhile you can try for full details from Beelines at FREEPOST, Bolton, BL3 6YZ.

The calculating little ...

VisiCalc was a real eye-opener when it appeared. Sure, there had been packages that let you set up 'what-if' situations in the computer and juggle the variables to check out alternative situations; but to use them you had to have access to a behemoth of a computer (an IBehemoth?) and it took about half a day to express the problem in a form the computer found palatable.

Then along came VisiCalc. Suddenly you only needed a personal computer, and it took about an hour to get familiar enough with it to start using the thing to do real 'what-ifs'.

And if you thought VisiCalc was doing a great job in getting lots of facilities on to a very small computer, let's hear it for Mark Turner. He has a son-of-VisiCalc — well, friend of VisiCalc — and it's called SimpliCalc (spot the similarity of names). It runs on the small 8k memory PETs; and his forthcoming version for Vic is a natural development. We hear Commodore itself has just signed up to sell the Vic SimpliCalc, too.

If you're not familiar with this kind of software, it's essentially a modelling package providing you with a big worksheet (bigger than your screen, usually — you can scroll the display over it to get from one end to the other). On this you can set up quantities which can be altered and recalculated on the screen. So you might have an entry for every item of expenditure you make, and a total box at the bottom: clearly the total is calculated by adding up the list, and obviously that's going to be the case whatever quantities are involved. This is a bit simplistic — VisiCalc and its clones are typically used for financial planning in companies, and that can be really complicated — but it gives you the idea.

In SimpliCalc's case you can set up a worksheet with columns up to seven characters wide. Obviously the maximum size of the worksheet is determined by how much memory you've got and how densely populated your columns are; the complexity of the calculations you want is another factor. But Mark Turner reckons you can get something between 250 and 300 elements on an 8k system.

That's enough to set up useful calculations. For instance, in 8k you could have 13 columns, for January to December plus an end-of-year total, and say 16 or 17 rows for separate items of expenditure being incurred during the year. If you have 32k, by the way, you should be able to top 2,500 elements. For each of those row/column elements you can specify what calculated result should be placed there. Either you can add up all or part of the row or the column: or you can perform four-function arithmetic on up to eight other elements anywhere in the worksheet, referencing them by their row and column co-ordinates.

The screen gives you a window on the sheet. For the Pet with its 40-column screen, SimpliCalc shows five columns and 20 rows at a time — you can move this around the worksheet, of course. On the Vic we'd assume Turner will give us three columns.

SimpliCalc for the Vic will require at least an 8k extension. It should be on offer shortly after this mag drops through your letterbox for something under £20, at which price it must be a bargain if it works — we have a review copy booked and we'll let you know. It will be on sale through Mark Turner's company, the Cronite Group; contact 021-773 8281 if you want more info, but calls may well be directed to Commodore if the marketing deal has indeed been signed by them.

How to make money

Commodore's notorious parsimony vis a vis the display size and its reluctance to avoid all these clumsy PEEKs and POKEs in Basic programs on the Vic obviously wasn't due to any shortage of financial resources. The company has just revealed that in the first six months of its current financial year, total sales topped $124 million. That's half as much again by comparison with the halfway point last year.

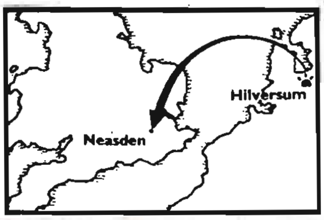
Dulcet tones

That trough around 5.30 on Sunday afternoons is hardly alleviated by David Vine's reports from the loftier regions of Europe about people who voluntarily throw themselves down 1-in-2 hills dressed in spray-on Dulux and wearing two short planks (ever noticed how the men are all "tough" or "professional" while the women are "pleasant" or have "a charming personality"?). Fret no longer: tune to Hilversum. Radio Netherlands has started transmitting 'telesoftware' — programs in the form of two audio tones capable of being taped and loaded into memory. It seems to be working, too: we know people who have successfully received programs in (you're not going to believe this) Neasden.

The bad news is that Vic isn't yet among the microcomputers catered for by the wily Dutch; but we're hoping to change that situation and will report back on progress. If you'd like to put your oar in, write and badger Jonathon Marks of Media Network at PO Box 222, 1200 JG Hilversum, Netherlands.

Meanwhile, how about a UK version? Maybe the more active Vic owners should put up the idea to their local radio stations: after all, a stream of squeaks and blips passing enthusiastically through the ether can't be worse than some of the programs transmitted by Radio Belper...

And would it work for CB?



More from the Commodore:

Hitherto Commodore has generally been prepared to let other people do the running when it comes to Vic add-ons, taking the attitude that while it was attempting to work up a head of steam in the Vic-making factories it might as well concentrate on pushing out Vics rather than diversify too much.

So Commodore makes a lot of the basic stuff, like the computer and the essential hardware, while the independents rush around like crazy selling the less vital elements like programs and other fripperies.

But changes are in the air. We hear that Commodore plans to have at least two significant new lines ready to go this year. For a start, it will be selling TVs with a built-in monitor switch. In other words, you get a real colour TV — but you can switch to A1 display quality for Vic screens.

Commodore hasn't apparently fixed up the manufacturer, but it will certainly be Japanese — and Chateau Commodore in Slough is full of Toshiba monitors.

What about price? Won't the dual-purpose TV/monitor be doubly expensive? The word is that Commodore expects to be able to sell it for about the same price as an ordinary small colour television set in the middling-to-classy price range. That could mean availability to you and me via dealers at £275 or so.

What's more, there's more than a possibility that you'll be able to rent it. For among the current crop of chain stores keen to take the Vic, there's at least two — Currys and Rumbelows — which do rent TVs. Rumbelows is going to sell Vics in something like 600 stores nationwide by the Summer; Currys will probably be testing the water by putting Vics into 20 or so of its shops this year.

Incidentally, there's a long queue for interested multiple retailers for the Vic. Dixons is going into Vics in a big way this Summer (270 shops); the Xerox Stores are already taking them (ten shops); GUS and Grattons are putting Vics into their mail order catalogues; and Harrods, Army & Navy and Debenhams (in Southampton) are trying Vics. What's more, we hear Laskys and Boots are on the point of signing up too — the Boots deal is the one Commodore really wants, of course, but Boots is reportedly quibbling about some of the electrical connections and other external features.

Meanwhile, the other Bright Idea for making Commodore money could also make some for you. Commodore is going into the cassette programs business, with a really competitive standard price — "nearly all" single-cassette progams for unexpanded Vics will be £4.99 as a matter of policy, says Vic marketing supremo John Baxter. He's got the first one ready, a game called Blitz. But he is dead keen to look at anything anyone out there might offer, with a view to repackaging it under Commodore's brand label and selling it on a royalty basis.

Frankly the royalties aren't fantastic: he's talking about five per cent, which on a £4.99 cassette is peanuts. But he argues that some of the programs will achieve sales in the thousands, and five per cent of 10,000 £4.99 programs does sound more interesting.

He'll consider anything, but with the plethora of games hanging around the Vic he would be especially interested in other programs: keywords to excite his interest are 'education', 'home' and 'business' (when applied to programs — other highly exciting keywords will be supplied under plain brown wrapper on request).

Feeling a bit off colour?

Hardware mods of a highly secret but most efficacious nature have been devised to sharpen up the colour quality on Vic displays by Harrow-based dealer, Ira Curtis Coleman. A sin[illegible] and very cheap soldering job will s[illegible] any problems you have with bad registration, colours disappearing on dark backgrounds, and insipid-looking tints — which sounds just what we need, because the office Sony is proving less than wonderful right now.

Negotiations with Commodore over the possibility of its incorporation on new Vics are in progress, and we eagerly await developments. Apparently Commodore hasn't made up its mind yet, and there is certainly much head-scratching going on down there. In the echoing caverns of the Commodore's Slough emporium, baffled technicians walk to and fro mumbling 'how does he do it?'

The good news is that whatever Commodore decides, IRC will be making the mod available to existing users from their own workshops. Currently IRC is sorting out production details. The hardware, itself consists of a smallish circuit made up of s[illegible] discrete components (which IRC hope to make available in the form of one single encapsulated module for easy connection) plus a single component addition to the RF modulator.

Price? It's a little too early to say exactly, but the company hopes to knock it out for under £35.

That's not all — the next IRC project is a pressure pad modification for the printer so that it can handle standard A4 paper. We'll buy it, Mr Coleman.

Action replay

Fed up with slow cassette response times? Can't afford the floppy disk upgrade? You can now opt for a medium price/performance alternative — the digital cassette recorder.

The disk unit will give you a fiver in change from £400. But £195 plus VAT buys you greater reliability and accuracy, and a twentyfold improvement in read/write times over standard cassette machines. The normal type of cassette currently in use transfers data at around 300 bits per second — this one goes at 6,000. It uses the superior hub drive technique rather than capstan and pinch roller; and instead of using audio tones it stores data on to neat dictation-type mini-cassettes by means of phase-encoding (which is basically the same highly superior technique that floppies use).

Depending on the form of sectoring use, mini-cassettes will hold up to 64k bytes per side and they cost about £15 for a box of six. The time taken to spool to a file is typically 20 seconds; the absolute worst case figure (end of end stuff) is 95 seconds. The recorder has its own RAM and doesn't use up any of Vic's.

We'll be reporting on this machine in a future issue. Meanwhile for anyone who can't wait it's available off the shelf from Currah Computor Components of Hartlepool. And yes we know how we spelt 'Computor' — that's how their name goes.

Modem magic

Back in the States, Commodore has announced a $100 modem for Vic. This will allow Vics to send and receive over phone lines. You'll need a bit of special software, but basically it's good news — a modem is a little black box that converts the computer's signals into those used by the telephone system, and vice versa: and $100 is definitely cheap for a modem.

With such a device you can talk to other Vic owners (at the prevailing phone bill rates, naturally). You could also dial up information services, like the kind of 'bulletin boards' and 'electronic mailboxes' that have mushroomed in the States — or like Compuserve and The Source, which are the two commercial databanks mentioned in Commodore's official unveiling of the modem (well, the modem wasn't actually in view: it was just the news that was unveiled. Hmm. In view of the somewhat disappointing history of Commodore's withdrawn Pet modem, maybe a degree of scepticism is in order).

The US announcement also had an aside about a projected VicNet from Commodore as its own version of this kind of service. We'll believe that when we see it.

As for the modem, there's a chance that Europe might see it next year — once the production hiccups are cured and once everyone in America and Japan has bought one. By then the liberalisation of British Telecom could well have meant that several low-cost Vic-compatible modems are already available from independents.

Eye on Commodore

By Dennis Jarrett

Commodore managed to cause a stir or two late in January at the Las Vegas Consumer Electronics Show with three products that we hope to get at during our trip to the Hanover Fair.

They include the long-expected 40-column 16k Super Vic. That'll be on sale in the States this Summer, says Commodore, and the price will give US buyers five cents change from $400. Lightning currency conversion will tell you that this must surely translate into a UK price of between £220 and £270 depending on prevailing exchange rates and the transatlantic mark-up.

On the other hand, Commodore's PR people here put out one of those press releases which quoted the US announcements verbatim and finished off with an "on the other hand ..." tailpiece obviously contributed by Slough-based marketing people.

In this case the afterword piously mentioned "the work required to finalise PAL versions" and the anticipation that "worldwide needs will exceed production caabilities in the short term" (which means the rest of the world will get the Super Vic before we do). For those reasons, intones the press release, the new products won't be seen in Europe until Christmas at the earliest — and probably not until 1983.

That, of course, will give them a chance to sell as many 22-column Vics as they can during 1982.

The two other products are obviously regarded as more important by Commodore's American HQ for Galactic Marketing (K Spencer prop.) since they broaden the company's product line into new areas. One is the Ultimax, a real 'home entertainments' machine that will retail at $149.95. This gives very high-resolution graphics for games, but it also has a clever slip-on second keyboard (they're membrane touch panels as used by Sinclair) that provides for musical keys.

With characteristic modesty and delicacy of touch the press release says the second keyboard "permits the Ultimax to function as a miniaturised musical instrument which duplicates with amazing simplicity the musical notes and sounds of a piano, harpsichord, clarinet or any one of several other musical instruments".

Well, there could well be a market for a really cheap synthesiser — if it plays chords: if it plugs into an audio amplifier: and if it's any good.

The other announcement is the Commodore 64, obviously the long-overdue replacement for the 40-column Pet: it too is a 40-column micro, but you get 64k RAM (clever name, isn't it) and you also have full colour.

As well as that there's music, graphics, joysticks, a serial bus, and an RS232 interface. "The world's lowest-priced microcomputer with 64k" trumpets the US press handout, pointing gleefully at the Atari 800 and the ageing Apple II. All three are full-colour 40-column computers, but the Commodore 64 has more memory — the Apple II gets only 48k into the cabinet, the Atari 800 stops at 16k.

What's more, it will be a lot less expensive too. At $595 the Commodore 64 will be a third cheaper than the Atari and less than half the price of the Apple ... assuming no price wars get started to distort those claims.

That's actually not unlikely. Apple must be interested in moving upmarket with its Apple III, now starting to sell at last, and the forthcoming 16-bit machines: so a bit of stock-clearing on the Apple II isn't out of the question. Atari's machine is not all that youthful, either, and a replacement is rumoured for later this year; you can't believe all you hear, but there might be some discounts going there too.

The trade and business press fell overboard for one of the more throwaway lines in the promotion for the Commodore 64. There was some talk of a CP/M option, a comment which can be taken with a truckload of salt as far as the early production models are concerned: the processor Commodore is using is the 6510, a technology upgrade of the trusty 6502, and for CP/M you really need a Z-80 or 8080 derivative.

Of course you do it by running a kind of slave processor as a peripheral. That's what happens with Apple (the Softcard option is a plug-in PCB containing a Z-80 and running CP/M and with the Pet (Small Systems Engineering give you CP/M by plugging in a box containing it and a processor). In Commodore's case it sounds as though the Commodore 64 will have a CP/M processor in a plug-in cartridge, rather like the Vic's expansion port cartridges.

All that sounds technically quite feasible, and in maketing terms it sounds quite likely too. That way a single machine costing say $1,000 ($595 for the Commodore 64, about $400 for the CP/M addition, say $600 for a disk drive, plus a $400 printer) would be able to run all the software developed for Vics, for 40-column Pets, and for CP/M. Sounds good?

Well, yes and no. So much of the 'standard' CP/M software comes only in specific disk formats. So either Commodore will have to start using someone else's disk formats (not likely): or you'll have to get your CP/M programs by down-line loading from a CP/M machine, by direct cable connection or via the phone links (possible but dificult): or someone will have to start putting CP/M programs on to Commodore-compatible disks

Then there's the next step, the 'all-purpose microcomputer' that so entranced journalists on (among others) The Wall Street Journal, the Guardian, The Financial Times, and Computing. If all the reported interviews with Commodore bigwigs are to be believed, Commodore is on the verge of bringing out dozens of plug-in emulator models — so that the Commodore 64 (and its progeny, of which there are said to be many in embryo form) can run programs not just from the CP/M stable but also from other environments.

In theory this is feasible. You get hold of the processor that IBM is using in the Personal Computer, copy the thing's operating system, and package it as a ROM-plus-microprocessor cartridge which enables your computer to run IBM's programs as well as your own. Or rather, instead of your own: you'll probably have to switch between 'IBM' mode and 'Commodore' operation, such that data files couldn't normally be accessible to both.

Great. Except that it's only easy for CP/M systems. That's because Digital Research, who wrote and own the CP/M operating system, will sell it to anyone with the money to buy. There seems very little chance of Commodore getting at the Apple III operating system or the IBM DOS or even Tandy's TRSDOS without a few patents being infringed.

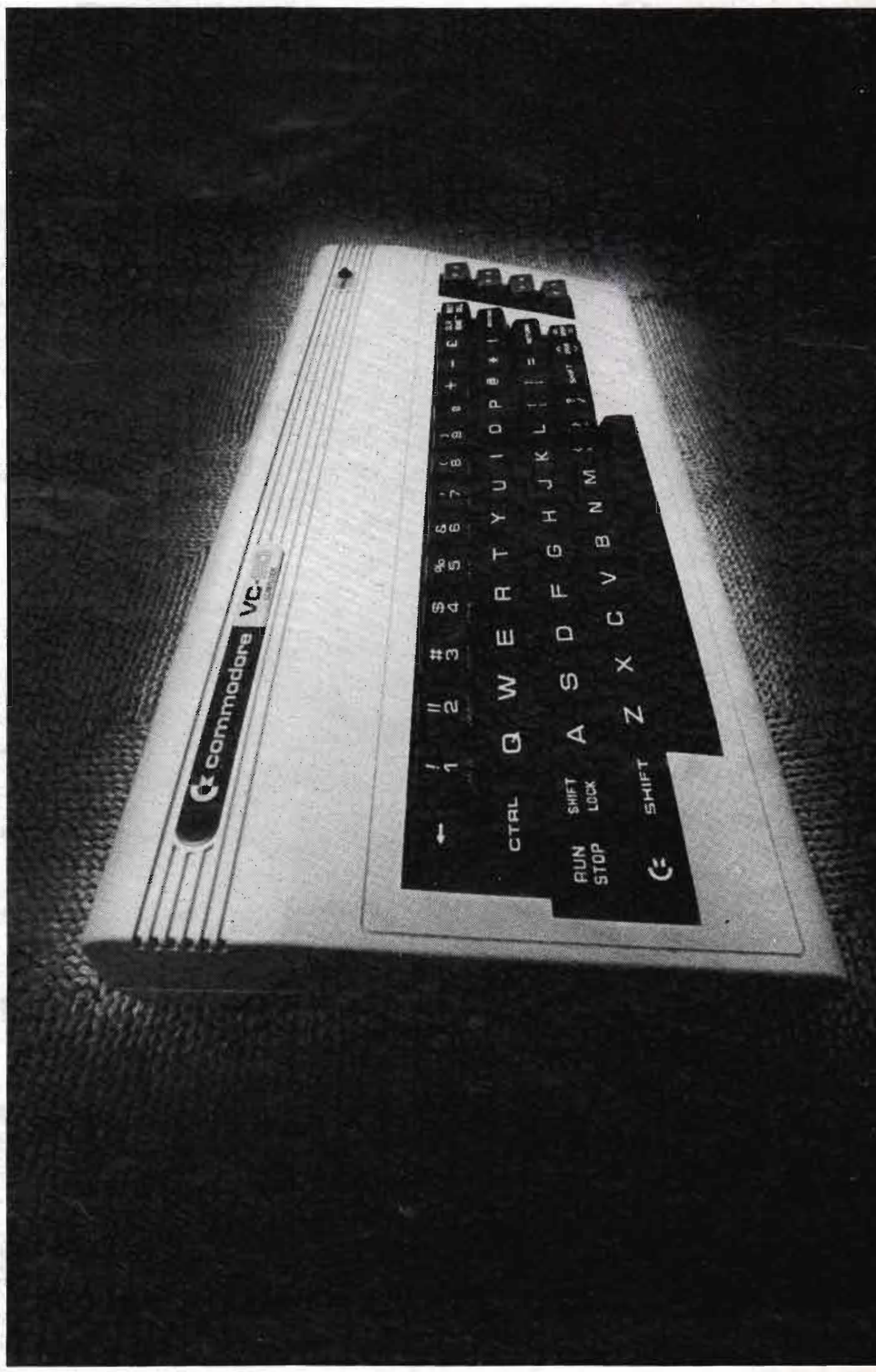
The other problem is the 'so what?' factor. By the time you've added all those trailing leads and extra hardware and other stuff to a Commodore machine, you might have been tempted to buy the neat wholesome 'real' computer from the 'real' vendor — who at least will be keen to supply decent manuals and grade A support. So who's going to try to save a few pounds on that?'

Not me, and no mistake. A cheapish plug-in CP/M cartridge I'll buy, because there is indeed a great deal of excellent software available in package form at quite modest prices for CP/M. But I'm not sure there's anything in the Apple or TRS-80 catalogues that I want so desperately: and as for the IBM Personal Computer, I think I'd rather pay the premium to get the promise of IBM quality.

On the second day

Martin Jacobs

Being an engagingly philanthropic kind of person, Martin Jacobs took it upon himself to provide a basic introduction. Part One appeared in our last issue: this is the second half. These articles are intended to help readers that have no computer background to become familiar (not to say intimate) with their VIC-20s — those you've just bought, or those you're going to buy. And if you want to read on, the standard Vic introductory booklet takes it from here.



We figure you probably don't need any instruction in carrying over the threshold, removing the outer coverings, or making the connections that nature intended. And we trust you spend a rewarding first few hours manipulating the various controls and obsering the reactions induced thereby.

That all happens during the 'Start Right Here' introduction that appeared in the last issue of **Vic Computing.** Those of you who were with us will have tried out all the keys except RUN/STOP, RESTORE, and that block of four ochre-covered keytops on the right.

Those are the function keys (hence the markings f1 to f8) and at this point they're of little use to us novices. They are there so that designers of games and other programs can assign special functions to a single keystroke: so f3 might be the laser weapon trigger in one game and the PRESS TO ADMIT DEFEAT button in another. Or it might be used to convert to a Greek alphabet on demand, or call up a Fourier transform function. Unless you have plugged in a cartridge or loaded a something from cassette tape which calls for one or more of the function keys, you will find that pressing them has no effect.

The experiments in Episode one — moving around the screen, drawing coloured stripes, and so on — were really just doodles. We never told the computer that we were serious: that you do by pressing the RETURN key.

Remember that SHIFT RETURN takes you down a line and back to the left margin, like the carriage return on a typewriter. RETURN without a SHIFT does that too — but it also says to the computer 'Interpret what I have typed as an order'.

Now there are two ways of giving the computer orders. First, you can just type the order on the screen in language which the computer understands and press RETURN. The computer then either does what it was told do do right away (which is why this is called the immediate mode); or alternatiely it goes surly and whiny and responds with some crack like 'SYNTAX ERROR', which means that you are not talking the language its makers taught it.

The other way of giving the computer an order

Continued on page 7

"Give me one good reason why I should choose a VIC 20 home computer."

1. VIC is outstanding value for money. No other colour home computer can give so much for under £200.

2. Total standard memory 25K made up of 20K ROM and 5K RAM.

3. Fully expandable to 32K of user RAM.

4. Microsoft Basic interpreter as standard.

5. Accessible machine language as standard.

6. Connects direct to monitor or standard television.

7. Full size typewriter-style keyboard.

8. Full colour and sound.

9. All colours directly controllable from the keyboard.

10. 62 predefined graphic characters direct from the keyboard.

11. Full set of upper and lower case characters.

12. 512 displayable characters direct from the keyboard.

13. High resolution graphics capability built into the machine.

14. Programmable function keys.

15. Automatic repeat on cursor function keys.

16. User-definable input/output port.

17. Machine bus port for memory expansion and ROM software.

18. Standard interfaces for hardware peripherals.

19. VIC 20 is truly expandable into a highly sophisticated computer system with a comprehensive list of accessories (see panel below).

20. Full range of software for home, education, business and entertainment on disk, cassette and cartridge.

21. Books, manuals and learning aids from Teach Yourself Basic to the VIC programmers' reference guide (a must for advanced programmers).

22. Full support for VIC owners – their own magazine 'VIC Computing' as well as a national network of VIC user groups.

23. National dealer network providing full service and support to VIC owners.

24. Expertise and experience – Commodore are world leaders in microcomputer and silicon chip technology.

25. Commodore is the leading supplier of micro-computers in the UK to business, schools, industry and the home.

26. VIC 20 is the best-selling colour home computer in the UK.

How many reasons was it you wanted?

Accessories include:

- Cassette tape unit.
- Single drive 5¼" floppy disk unit (170 K bytes capacity).
- 80-column dot matrix printer.
- 3K, 8K and 16K RAM expansion cartridges.
- Programming aid packs; machine code monitor cartridge, programmers' aid cartridge, high resolution graphics cartridge.
- Plug-in conversion box for a full 32K, 40-column x 25 lines VIC including Prestel compatability.
- Prestel/Tantel interface package.
- RS 232C communication cartridge.
- Memory expansion board
- IEEE/488 interface cartridge.
- Joysticks, light pens, paddles and motor controllers.

commodore

VIC 20

The best home computer in the world.

FOR MORE INFORMATION ON THE VIC 20, TELEPHONE OR WRITE TO: COMMODORE INFORMATION CENTRE, BAKER STREET, HIGH WYCOMBE, BUCKS, OR TEL: SLOUGH 79292.

is to put it in a program. A program is simply a series of orders; and in VIC's case each order starts with a line number. If you give VIC an order which starts with a number it will not carry it out immediately: instead it will store it internally and wait until you type in R-U-N-and RETURN. Then it will look at all the stored orders, if there are more than one, and it will carry them out in numerical sequence.

Orders can thus be entered into the computer memory by you, in which case you have become a programmer (hooray). Or they can get there by plugging a cartridge in the back of the case, in which case the cartridge actually contains extra memory with the orders already memorized in it. Or they can get there by being played back from a tape cassette or floppy disk into the computer memory.

The fastest to do these methods is the disk, but taking out a cartridge and plugging in another takes only a few seconds. Cassette storage is slow to load into the computer; but it is really money-saving compared with cartridges or floppy disks. Typing in a program by hand is by far the slowest (and the cheapest, unless you charge yourself by the hour).

Let's experiment with this concept of 'immediate' as opposed to 'programmed'. One of the words which VIC recognises is PRINT (which has a handy single-character abbreviation '?'). So PRINT "MABEL" means the same to the computer as the instruction ?"MABEL": you have told it to print the world 'MABEL' on the TV screen, starting when you hit RETURN. In fact it would print anything that you put in the quotes.

Type in ?"MABEL" RETURN. The cursor should go down a line and over to the left: 'MABEL' should appear on the screen: and the cursor should then drop two lines and print 'READY', which means "I, VIC, have done what you ordered (in a twinkling of an eye with one hand tied behind my back) and am ready for another order". The cursor should be blinking on and off below READY.

You have just given an order in the immediate mode. To do something in the programmed mode is almost as easy. I assume that you don't have any programs already in memory: but just to be sure, type N-E-W and RETURN. The word NEW tells VIC to clear out any programs it has stored and it will respond with READY to indicate that that has been done.

Then type in 10?"MABEL" RETURN. This time the computer sees that you have started with a number; it therefore does not carry out your command immediately, but instead stores it. Now you have a program, albeit a trivial one, sitting in VIC's memory.

To set the computer to perform it type R-U-N RETURN, and the results will be just what you expected: VIC prints 'MABEL' again, and 'READY'. The program is still in memory, waiting to be run again if desired.

By now, however, the screen is beginning to be cluttered up with commands and responses, so I shall show you how to instruct the computer to clear the screen before it prints anything. We could clear the screen ourselves, you remember, by pressing SHIFT CLR; but it will be better to have the computer do it automatically each time, as this will erase the command and show only the response.

Let's have a look at our one-line program. It's still there on the screen. Use the CURSOR keys and SHIFT to reposition the cursor so that it sits on the 'M' in 'MABEL'. Then press SHIFT and INST together. This should open up the line and insert a space between the quote mark and the 'M'. The cursor will be blinking in the space, indicating that whatever you type next will appear there.

Now hold down SHIFT and press CLR. Instead of clearing the screen as we might have expected, this will print a reverse heart character in the space.

That is the symbol which VIC recognises as a CLEAR command when it sees it between quotation marks. You can get it by pressing SHIFT and CLR whenever you are within quotes or whenever you have opened up a space by using INST as we just did. The same technique will write other special symbols for HOME and for the cursor movements.

We have modified our program now we must hit RETURN to tell the computer to make a note of it. We don't need to move the cursor to the end of the line — leave it anywhere on the line and VIC will gobble up the whole line. In fact, a program 'line' can be up to 88 characters long, counting the line number, spaces, and all — that's four screen lines.

VIC will throw away the old line that was numbered 10 and replace it in its memory with the new line number 10. The old one is gone forever, so it is best to take a good look at the new line to be sure it's right before you hit RETURN.

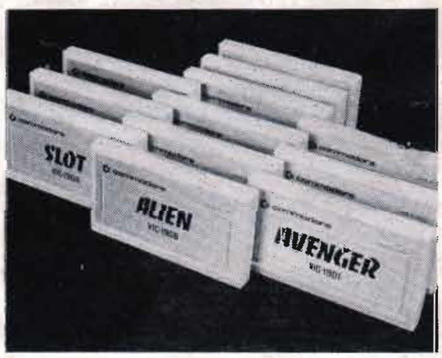
Now, doing all this will leave the cursor blinking on top of the word RUN (which was left over on the screen from typing it before). There is an interesting and useful feature in VIC: if you now press RETURN, VIC will understand that you are telling it to RUN again — because the cursor was somewhere on the line saying RUN, just as before. Be sure, though, if you use this trick, that there isn't anything else on the line that you don't want the computer to react to!

Now press RETURN. The screen will be erased, 'MABEL' will be written starting in the upper left hand corner (because the CLEAR command automatically includes HOME), and the ubiquitous 'READY' and blinking cursor will appear below.

Although everything seems to have vanished, your program is still there in VIC's memory. It can be recalled for your perusal at any time by typing L-I-S-T RETURN.

And if you want to startle a friend who knows even less about computers than you do, prepare a program which tells VIC to clear the screen and then shift the cursor down a couple of lines (putting those commands in quote marks) before printing something like "HI THERE SUSIE OLD FRIEND! GOOD TO SEE YOU!". Test it to make sure it's working properly. Then blank out the screen, type RUN, go and fetch an old friend called Susie, and let her press RETURN.

But what do you do for an encore? Try the next issue if you need some ideas.



Browsing the VIC chip

By Jim Butterfield

The computer is called Vic, for Video Interface Computer . . . but there's a chip inside which is also called VIC, for Video Interface Chip. The chip bears the number 6560 or 6561; it's used to make good things happen on your television screen.

Beginners often don't realize that memory addresses are used for more than memory storage. In the Vic computer, addresses 36864 to 36879 may be PEEKed or POKEd. These locations are not used for memory — they hold controlling information for the VIC chip. We're going to look through those addresses, experimenting as we go. We may learn some new things about our computer.

Location 36864 (Hex 9000)
Values 0 to 127 set the position of the left border on your screen. The usual value is five. Try the following quick "slide change" line:

```
FOR J=5 to 30: POKE 36864, J: NEXT J :FOR
  J=30 to 5 STEP -1:POKE 36864, J:NEXT J
```

If you add 128 to the value in 36864, the screen will go into interlace mode. In most cases, all you'll notice if you poke 36864, 133 is a little dither in the screen detail. However, a few television sets are built in such a way that they won't work unless you set interlace mode with this POKE.

(Editor's note: My Vic comes with 36864 containing value 12 rather than 5. Since Jim lives in Toronto, which is not the kind of place where strange things happen, one can only assume that there's some kind of different settings back at the factory when Vics come to this country.)

Location 36865 (Hex 9001)
Values 0 to 255 set the position of the top border on your screen. The usual value is 25. Try making the screen curtsy with this:

```
FOR J=25 TO 45.POKE 36865,J:NEXT J :FOR
  J=45 to 25 STEP -1:POKE 36865,J:NEXT J
```

Location 36866 (Hex 9002)
Part of this location tells the chip how many columns to put on the screen. This will always be 22. But there's an extra value of 128 may be added to set "alternative screen" mode. Normally, the 128 is added in, and you'll find 150 stored in this location. If you want to go to an alternative screen, remove the 128 element with POKE 36866,22 and the screen will now take its information from a new area. There are quite a few things you need to do if you wish to play with this — see 'Alternate Screens' from a previous issue.

Location 36867 (Hex 9003)
A busy one. In fact, it's always changing. Try typing ?PEEK(36867) several times and you'll see that you get different values — sometimes 46, sometimes 174. Let's ignore that extra 128 for the moment; we'll deal with it again when we describe the following location.

The basic value held in this location, normally 46, is the number of rows on the screen multiplied by two (23 rows, right?). You won't want to change this one.

There's one more thing hidden in this location, and it's an important one. If you add one to the value, the character generator will switch to double-character mode. This means that each character you type will occupy double the usual screen space.

This won't work automatically, however. If we want to draw characters that are twice as big, we must supply the Vic with 'pictures' of the new characters; the old pictures won't do since they are not big enough to fill the new space. So prepare for a little confusion when you try this next experiment. Strange things will happen because we haven't built and connected up new character tables.

Type POKE 36867,47 The screen will go rather strange. Don't worry about it for the moment; just press the screen clear key (shifted, of course). The screen will clear, although the cursor looks rather odd. Not to worry, we'll forge right ahead.

The first character in the Vic's table of characters is the "©" symbol; the next is an "A", then a "B" and so on. Now: type the © key. Instead of getting the first character, we get the first two, one above the other. Try typing the "A" and you'll get B-over-C, the next two characters in the list.

What's happening here? Each character you type is filling double its normal screen area. In doing so, it's grabbing twice as much information from the "character picture" table . . . and since that table hasn't changed, that means two characters. Since the Vic knows (or at least thinks) that the character pictures are twice as big, it reaches further into that table for each character that it needs.

When you decide to use this feature, you'll write your own character picture tables and everything will sort itself out. This feature is likely to be used most for high-resolution graphics. The elements of the character picture table will control individual dots, or pixels, on the screen.

You may bring your VIC back to normal by typing POKE 36867,46 but you'll need to type blind since the screen isn't much help. You might prefer to turn the computer off for a moment; when you turn on again, everything will be back as it was.

Location 36868 (Hex 9004)
This location changes continuously. It's connected with the high-bit (128 value) in the previous location. In principle, it tells you precisely where on the screen the picture is being drawn at this instant. In practice, it's not much use to Basic programmers — by the time you read it, a different part of the screen will be active.

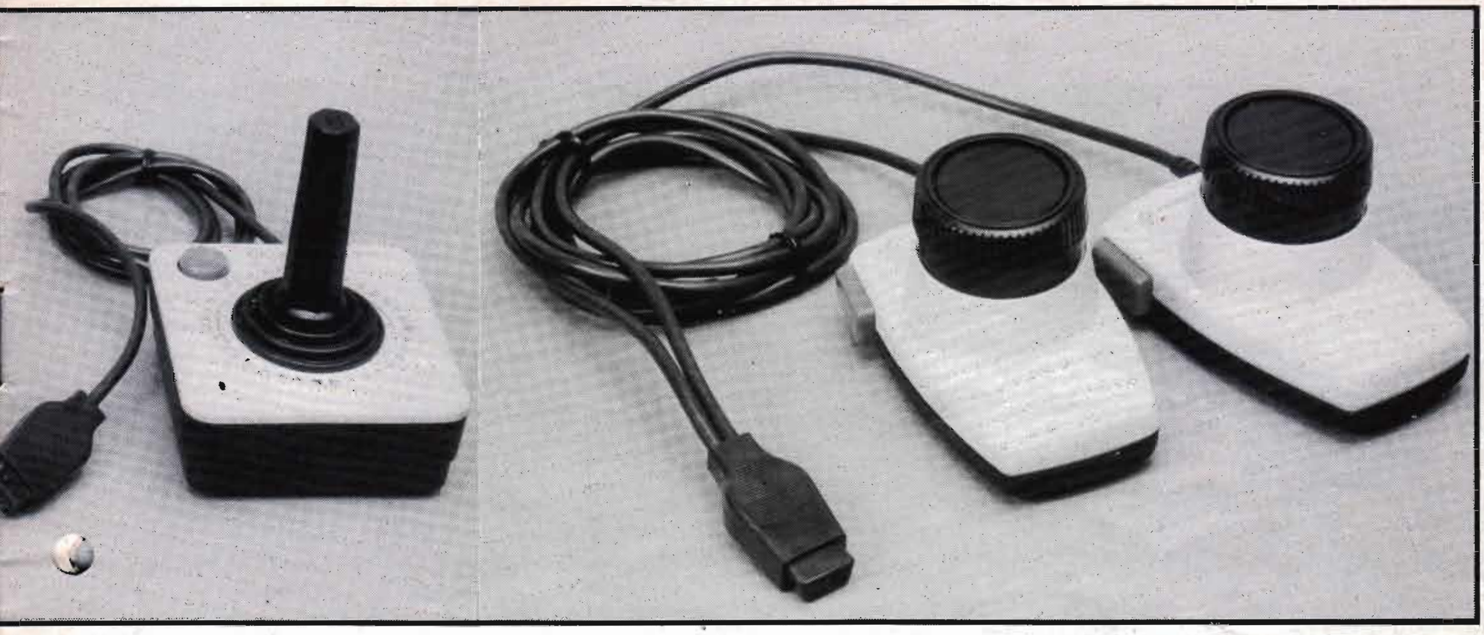
Location 36869 (Hex 9005)
This is a very important address. It controls the location of two tables: the table where screen characters are held, and the table which holds the character pictures. Let's take them one at a time.

The screen table holds the five hundred or so characters that are displayed on the screen. It's quite a job to calculate the screen address: let's take a shot at it.

Take the contents of location 36869, divide by 16, and throw away the remainder. That should give you a number from 8 to 15. Subtract 8 and double it, giving an even number from 0 to 14. Now: if the content of 36866 are 128 or greater, add one to get a value from 0 to 15. Multiply the result by 512. At this point you should have a value from 0 to 7680. That's where your screen table is located; it will normally be at location 7680, but might move if you add extra money.

That's quite a calculation; some of the things it implies deserve a separate article. For the moment, let's observe that the screen table address must always be in the range of 0 to 7680, and must be a multiple of 512. If you wish to set up your own screen table within this range, do the calculation in reverse: divide by 512, subtracting 1 if odd, dividing by two, adding eight and finally multiplying by 16. Whew! We can see that the "alternative screen" bit (128 value) in 36866 is really part of the much larger screen address.

The character picture table address is also defined in this location. We'd need to change



this if we wanted to define our own characters, single or double. Of course, we'd also need to define character pictures for all characters we wished to print.

The computation of the address is complex.

Take the contents of location 36869 and divide by 16. Now take the remainder — not the quotient — and if it's greater than 7, subtract 8. On the other hand, if the remainder is not greater than seven, add 32. By this time, you'll have an adjusted remainder which is either less than seven or between 32 and 39.

Multiply this value by 1024 and you've found your character picture table address. It will be in the range of either zero to 7168 or 32768 (the normal value) to 39936, and will be an exact multiple of 1024

If you wish to set up your own character picture table, you'll usually want it to point to a RAM address in the range of zero to 7168. In such a case, you'd reverse the calculation: take the address, divide by 1024 and add 8 and you're there.

Don't forget that the screen table address and the character picture address are packed together into this location. You'll need to set them both at the same time. By the way, the official name for the two tables are the "Video Matrix" and the "Character Cell Table".

Feel free to play with this location; POKE values as you wish. But unless you plan carefully, all you'll get is a crazy screen

This was a tough one . . . now we can try some easier locations.

Locations 36870 to 36871 (Hex 9006 and 9007)

Here's your input from a light pen. No, a light pen isn't a ballpoint that weighs less than half an ounce — it isn't a pen at all. It's a device that plugs into your Vic that looks a little like a pen. Point it at the screen, and these locations will tell you where you are pointing.

A standard Atari light pen may be used (it's expected that Commodore will manufacture their own light pen soon). Many light pens have either a button or a spring-loaded switch in the tip which signals whenever the light pen operator wants attention. The switch is implemented in the Vic computer, but it is not connected to the VIC chip (you'll find it mixed in with other things in location 37151).

You can read the X and Y positions of the light pen in locations 36870 and 36871 respectively. You won't read row and column values: the numbers will vary between zero and 255, and you'll need to do some calibration for the particular model of light pen that you have fitted.

Watch for 'jitter' on these values. Even though the light pen doesn't move, the readings may jump about a little on successive readings. Depending on what you're doing, you may wish to use an averaging technique to make the readings smoother

Another method is called hysteresis; in simple terms, it means that a value is ignored unless it differs from previous readings by more than a given threshold amount

Locations 36872 to 36873 (Hex 9008 and 9009)

These are paddle input values. Two paddles, similar to Atari paddles, may be connected and their values will be read here. You may not be able to track the full range of rotation of the paddles.

Once again, watch for jitter on the input values here.

To keep the record straight, a joystick can also be connected to the Vic . . . but the position of its buttons are not detected by the VIC chip. They arrive in other locations (37151 and 37152).

Locations 36874 to 36876 (Hex 900A to 900C)

These are Vic's voices. Setting a value of 128 or higher into any of these locations produces sound; the value you POKE produces the pitch. By poking two or three locations, you can produce harmony.

All voices are controlled by the sound level which is set at address 36878; try POKE 36878,15 before you play with the voices so that you'll get good volume. A value of less that 128 in any of thes voice locations makes that voice silent.

It's interesting to note that the voice controlled by 36874 is the softest, and the voice at 36876 is the sharpest. So you'd use 36876 to carry the melody, and the two other voices as the sidemen.

Location 36877 (Hex 900D)

This is similar to the music voices, except that it generates noise. A value of 128 or more produces noise. The higher the value, the higher the pitch of the noise (from growl to hiss). Once again, this is controlled by the sound level of 36878.

Location 36878 (Hex 900E)

If the number in here is less than sixteen, it represents the sound amplitude (see the four previous locations). If it's sixteen or more, an extra factor is at work: multi-colour.

Normally, each character position on the Vic contains only two colours: background and foreground. If we decide to use multi-colour, we can add an extra two colours to each character: the border colour plus one more that we may select. We select this colour in the high part of location 36878. If you divide the contents of this location by sixteen, discarding the remainder, you'll get the designation of the "auxilliary colour".

Interestingly, each character on Vic's screen is

Butterfield

independently selected as two-colour or multi-colour, allowing us to have a mixed screen. This is done in the colour nybble table which sets each character's colour.

Try the following: Clear the screen and type the letter A in the upper left-hand corner. Now go to a new line and type POKE 38400,8. You'll see that the letter A has suddenly turned weird and multi-colour, but the rest of the screen is unchanged. Notice that we did not POKE the VIC chip, but an entirely different memory location that is keyed to the one screen address. To do the job properly, you'll need to define your own character pictures.

Location 36879 (Hex 900F)
Last location in the VIC chip, but a busy one. Let's break it down into its three elements.

Divide the contents of this location by 16, and note the result as "Screen Background Colour". Now take the remainder; if it's 8 or more, subtract 8 and note: Foreground/Background = ON. The remaining value of zero to 7 can be labelled "Frame Colour".

The Frame Colour is a favourite of mine; it's an easy signal to the user of some situation I want to tell him about without affecting the contents of the screen itself.

If there's a danger, an error, or a game explosion, I can flip the border to red with POKE 36879,26 and later restore it with POKE 36879,27.

Another example: Rather than typing a PLEASE WAIT message, I might walk the border through a range of colours so that the user can tell something is happening.

Screen Background colour can be a very nice psychological support. If you set up a system so that accounts receivable can be done on one background colour and accounts payable on a different one, the user can be "keyed" to recognize that he's in the right program. It's a little like decorating each floor of a building in a different colour so that people won't get the wrong one. Try combinations such as POKE 36879,155 as see how you like the effect.

Now for the Foreground/Background business. Normally (F/B = ON) we know that we can type characters of many colours on a single colour background. Sometimes it can be very handy to do the opposite; in other words, we want to type single colour characters on a background whose colour may very from character to character.

Try POKE 36879,19. Now clear the screen and type a few characters. Change colour and type some more. Do you see what's happening? You are changing the colour of the background, not the colour of the characters themselves.

By playing around with these locations, you can discover potential that you never knew existed. Once you know it's there, you can then exploit it for your own special effects.

There's a rich variety of controls and information available in the VIC chip. You may not need to use them all . . . but isn't it fascinating to play around?

VIC 6560 Chip

	$9000	Inter-lace	Left Margin (=5)		36864	36880
	$9001	Top Margin (=25)			36865	36881
	$9002	Scrn Ad bit 9	Columns (=22)		36866	36882
	$9003	bit 0	Rows (=23)	Double Char	36867	36883
	$9004	Input Raster Value: bits 8-1			36868	36884
	$9005	Screen Address bits 3-10	Character Address bits 13-10		36869	36885
	$9006	Light Pen Input		Horizontal	36870	536886
	$9007			Vertical	36871	36887
	$9008	Paddle Inputs		X	36872	36889
	$9009			Y	36873	36890
36891	$900A	ON	Voice 1		36874	36891
36892	$900B	ON	Voice 2		36875	36892
36893	$900C	ON	Voice 3		36876	36893
36894	$900D	ON	Noise		36877	36894
36895	$900E	Multi-Colour Mode (=0)	Sound Amplitude		36878	36895
36896	$900F	Screen Background Colour	Foregnd Backg	Frame Colour	36879	36896

Vic versus Atom

This report from Dr Boris Allan is a brief comparison of two small microcomputers, the Commodore Vic 20 and the Acorn Atom, in terms of their performance on some arithmetic benchmarks that might give an indication of their overall capability.

As both use the MOS Technology 6502 processor it made sense to include in the comparison the two larger microcomputers that are based on the same ship — the Commodore Pet and the Apple II.

I made a conscious decision to concentrate on the relative capabilities of the versions of Basic that were on show. In particular I elected not to consider the graphics side — though I have found the graphics on both the Apple II and the Atom to be highly commendable, and the Vic seems far superior to many machines for colour graphics.

The comparisons are based on my MAGI benchmark tests (more about them in "Practical Computing" for June and November 1981): unlike most benchmarks, these actually do something. The three tests for floating-point calculations are designed to emulate practical situations in numerical work. There is also a test for integer calculations, based on an algorithm for Ackerman's function.

Before discussing the tests, I have to make an important point: the Vic may look like a toy, but it isn't one. It is a true computer, as indeed are all the machines being compared here. To decry the Vic and similar systems because they can be bought as electronic toys, without considering what they can actually do, is narrow minded and short-sighted. Yet you still hear the accusation.

Less than 15 years ago "real" computers included the IBM 7094 and the Elliot 803. On one of the MAGI tests, all four micros tested here turned out to be more accurate than the IBM machine. Another MAGI benchmark, yet to be released, was five times faster on the Vic than on the Elliot 803 — and the Vic version was in Basic, while the 803 ran it as a program written in the Algol language.

Back with the modern micros, one of the most noticeable results was that numerical accuracy of the **floating-point** calculations for the Vic, Apple, and Pet was much the same for all three machines. On average the Atom was slightly more accurate, but only slightly.

I was very surprised by the timings. Although it is as accurate as the Apple and Pet, the Vic is always fastest of the three. The Apple incidentally is always faster than the Pet: the Atom is slower than the Vic or the Apple, and on average its timings are about equivalent to those of the Pet.

The Vic was no more difficult to program for the tests than the Pet (the Basic is more or less the same, of course) or the Apple. On the other hand the Atom Basic does not really seem to be designed for floating-point work: you have to buy an extended system, you cannot use DEFined functions, and arrays can only be one-dimensional with restrictive names. Personally, I find the coding of floating-point work to be over-complex (and tedious) on this machine.

The **integer** benchmark can be run in several modes depending on what the language offers. On the Apple and Atom there could be any integer-only Basic. Or the program could be run with floating-point numbers in a floating-point Basic (all the machines could do that). Or integer numbers could be expressed in a floating-point Basic — all bar the Atom have this facility.

The fastest machine in the integer test turned out to be the Atom in its integer-only mode — it was almost twice as fast as any other. The Apple II running Integer Basic came next, with the Vic in its normal floating-point mode not far behind.

When the Atom was running in floating-point mode it was about 50 per cent slower than the Vic in its floating-point mode. The difference in speed between the Vic and the two larger microcomputers, the Pet and the Apple, became even more marked for programs using the ordinary floating-point functions in Basic. This contrast amounts to more than 10 per cent — quite a large difference, again.

Programming this MAGI test requires that an array be dimensioned as STACK (1000), and this was too large for the little Vic — so the array had to be declared as STACK (500). When all the variables were defined as integer, though, the declaration of STACK%(1000) was accepted (the postfic '%' indicates that this is an integer variable); but then the program ran slightly more slowly.

For the three 'standard' mode machines (all except the Atom) the definition of variables to be integer mean that the programs can be smaller, albeit running more slowly.

Conclusions

The conclusion? For modest-sized programs using floating-point arithmetic, even the unexpanded-memory Vic is the equal of the Pet and the Apple II in terms of accuracy. It has the clear edge in speed over the Pet, and it is marginally faster than the Apple. In the integer test the Vic again did well overall for speed.

Given the excellent graphics and its speed for integer-only work, the Acorn Atom appears tc be a powerful machine for game-playing and discrete simulations, especially given its excellent graphics. It is easy to program in

	Atom	Vic	Pet	Apple
Test 1	110secs	87secs	125secs	92secs
Test 2	7.3secs	5.5secs	7.4secs	5.9secs
Test 3	92.1secs	42.8secs	51.4secs	46.0secs

Note that all numerical accuracies are similar for all machines

Atom		Vic		Pet		Apple		
Int	F	F	F I	F	F I	Int	F	F I
125	333	218	226	252	278	208	257	268

All timings are in seconds. Int = Integer Basic. F = Floating-point Basic, and F I = Floating-point Basic using defined integers

machine code, and it is essentially such an uncomplicated machine that it must be an economical way of learning about the mechanics of computing.

But if you want to start computing with a small machine that enables you to make a simple transition at some time to a larger computer, the Vic 20 is the obvious choice. It will run games and simulations more quickly than the Pet, and probably faster than the Apple (that needs more tests); it will allow the user to perform numerical work in a standard environment speedily and easily

If you are seeking an introduction to computers via a machine that is not intended to reinforce a 'computer knowledge' elite the Vic has a clear claim

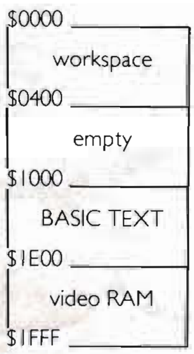
VIC Chips

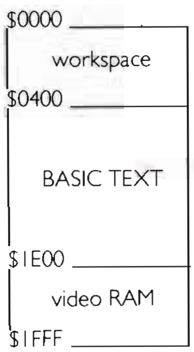
By Mike Todd

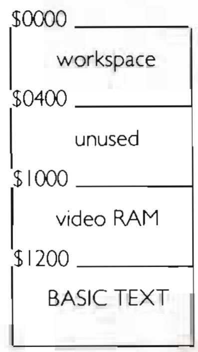
Mike Todd is the resident Vic enthusiast at IPUG, the Independent Pet Users' Group. Since he knows more about the Vic than most, his amblings through Vicland make a good trail for us to follow . . .

Memory Maps

The Vic has a somewhat odd memory map. It comes in three varieties:







These memory maps show the three possible combinations. The first is the bare Vic; you will see that there is a gap from $0400 to $0FFF. In the second, this space has been filled with the simplest memory expansion of 3K. The third shows what happens as soon as RAM is added after $2000 — the video RAM immediately moves to $1000 and BASIC text now starts at $1200.

All this is set up on reset when a RAM check (non-destructive, unlike memory verification in the Pet) identifies the lower and upper limits of RAM and sets the map accordingly. It would appear not to allow for RAM at $0400-$0FFF if RAM is added above $2000. Although this is not accessible to Basic, it could provide a useful home for machine code.

If you are in any doubt as to what RAM is where, the following locations may help:

$0281/2	(641/2)	— address (lo/hi) of lowest RAM location
$0283/4	(643/4)	— address (lo/hi) of highest RAM location
$0288	(648)	— page number (ie high byte) of screen RAM
$2B/C	(43/4)	— address (lo/hi) of start of Basic RAM
$37/8	(55/5)	— adress (lo/hi) of end of Basic RAM

Uncertainty about which set-up a particular Vic has will cause problems when accessing RAM directly — not only in machine code programs but also if programs access the video (or indeed the colour screen RAM, which also moves!). Such programs should determine where RAM starts PEEKing or POKEing. Note that if the video RAM starts at $1000, then the colour RAM starts at $9600; if $1E00, it starts at $9400.

LOADing programs will also cause problems since they may have been SAVEd on a machine with a different configuration. To get round this problem, all normal LOADs from cassette or the serial port (LOADing from RS232 is not allowed) will begin a LOAD at the start of BASIC TEXT. However, this is not much good for machine-code programs or other LOADs which are dependent on memory position.

To allow for this, the format of LOAD is

LOAD"filename",device,mode'

. . . where mode is '0' for the normal relocate LOAD (the default mode) or '1' if you want the program to LOAD at the same address it was SAVEd from.

To make life a little easier, SAVEing to cassette allows for the programmer to specify if a relocated LOAD or an absolute LOAD is required. To do this, the same format is used for SAVE as was used for LOAD, although it only works with cassettes. Mode 0 allows a normal LOAD to relocate the program while mode 1 will force an absolute LOAD.

At the start of every program on cassette is a header with the filename, start and end addresses. There is also a program identifier which is 01 under a normal (mode 0) SAVE. If the Vic finds an 01 identifier then it performs a relocating LOAD. However, a new identifier (03) is used to indicate that an absolute LOAD should be performed, and this is the identifier used in a SAVE with mode 1.

With the Pet always use an 01 identifier, programs developed on the Pet will always relocate unless the absolute LOAD mode is used. Any programs SAVEd on the Vic using a mode 1 SAVE will not LOAD on the Pet since the Pet does not recognise the 03 identifier.

Passing Parameters

The Vic has a very useful facility which allows the X, Y, accumulator and status registers to be passed to and from Basic. This is done by POKEing 780 with the accumulator value, 781 with the X, 782 with the Y and 783 with the status register. As soon as the SYS command is entered, the registers are loaded with these values, the routine executed and then the new values returned to the same locations. Thus it is possible to pass parameters relatively easily to Vic routines without having to write machine code to do it!

Functionality

There is a slight but widespread misunderstanding of the function keys. These are not 'soft' as you may have been led to believe. Instead, they merely generate ASCII characters which have to be detected through the GET command (although machine code software could allow these keys to be actioned directly). The ASCII codes for the function keys are:

f1 = 133, f2 = 137, f3 = 134, f4 = 138, f5 = 135, f6 = 139, f7 = 136, f8 = 140.

RS232 Bug

One serious bug has been reported by Commodore in the RS232 handling software. Releases of the Vic with KERNEL ROM number 07 have this problem: Commodore hopes to release a new ROM when all the other problems are ironed out. The bug occurs during multi-line handshaking (the three-line handshaking is fine); when a specific sequence of signals occurs, the Vic can hand waiting for a handshake signal that will never occur. ●

The Vic chip itself is one of a whole family of video controller chips from Commodore's MOS Technology operation. The 6560 and 6561 are the current Vic chips, the former being NTSC (for USA use) and the other being a PAL contoller (for Europe). There are two other chips in the range, the 6562 and 6563 (again, NTSC and PAL respectively; they provide a screen resolution of 300x200 rather than 190x200.

VIC Rhino

By Tim Duncan

Pet enthusiasts quiver with knowledgeable anticipation when rhinos are mentioned. This manic little game is now on offer from Tim Duncan, self-confessed rhino addict and (since Christmas) proud possessor of a Vic. He's a postgraduate research student in Cardiff.

Someone once told me that more time had been spent in American computer installations on Lunar Lander, than on the whole of the space programme. I can well believe it, computer games are highly addictive.

I first became interested in micros when we had a short course on them in my second year at University. It wasn't long before I discovered that there were a whole range of games programs. One of the most popular at the time was Infoguide's 'Rhino', and I whiled away many a lunch hour on the Pet, dodging the cantankerous creatures. So when I bought my Vic, just before Christmas, I decided to write a version that would fit into the Vic's 3.5K.

How to play VIC Rhino

The scene is a hot African jungle. Hiding in amongst the trees are a number of rhinoceroses (rhinoceri?) — the top of the screen tells you how many. You are the white dot at the foot of the screen, and the aim of the game is to get safely through the jungle to your home ("H" at the top). You move by using the function keys:

Function keys	Function keys with shift
f1 = up	f2 = up/left
f3 = left	f4 = up/right
f5 = right	f6 = down/left
f7 = down	f8 = down/right

The trouble comes when a rhino sees you because it will chase you and stomp on you if it catches you. The rhinos only appear when you come into their direct line of vision — you'll get an "uh-uhh" sound when one is about to appear.

Each rhino can move only one square at a time, and Vic will wait for you to enter your move before the rhinos move again: you will see "YOUR MOVE" at the top of the screen. You have a slight advantage over the rhinos in that you can move diagonally through gaps in the trees while they can't. For example, in Diagram I you could move from square 'a' to square 'b' but the rhino can't.

When you get home safely, Vic will offer you another go ... but this time there will be one more rhino! So how many rhinos can you dodge?

What the program actually does

Now for the benefit of those who haven't zoomed off to type in the listing straight away, I'll explain briefly how the program works. You may be able to pick up a few ideas that you can use in programs you write yourself.

VIC RHINO uses POKE statements to create the jungle and move the players around, so if you're not clear on POKE statements now's the time to look at the handbook. On a 5K Vic the address of the top left square of the screen (which I shall refer to as the Origin) is 7680. All the addresses used in the program are calculated from this address using X, Y coordinates to denote the line and column of the square concerned.

If you have added more than 3K memory expansion to your VIC you will find that the screen addresses are different. Since all the addresses are here derive from the origin, all we have to do is change the value we use for our origin. Line 1001 will do this for you automatically.

The main part of the program runs from line 1010 to 1030, using a number of subroutines to deal with specific tasks.

200 — sets up the screen
5000 — checks your moves
6000 — moves either you or a rhino
7000 — decides when a rhino may appear
7100 — provides sound effects
8000 — decides the direction the rhino should move

The jungle is created by going square by square around the screen, and randomly POKEing reverse squares for the trees. The next two lines ensure that your home and the position you start from have several clear spaces around·

them. Lastly the routine produces random X, Y coordinates for each rhino, and stores them in an array.

Having set up the game, the program returns to 1010 and gets your move. I've noticed people often get carried away at the keys while the computer is busy moving the beasts: to make sure that the program ignores this POKE 198,0 clears the keyboard buffer just before getting the player's move. So keys will only register if pressed after "YOUR MOVE" appears.

An IF statement ensures that only the function keys are accepted, and the program goes to 5000 to check the move. The key is converted to a number in the range 1-8, but just to complicate matters the ASC codes of the keys are not the same as the numbers printed on the keys!

'M' Values	Key	Direction
1	f1	↑
2	f3	↖
3	f5	←
4	f7	↗
5	f2	→
6	f4	↙
7	f6	↓
8	f8	↘

If you are at the top of the screen (ie. X = 1) then the routine will disallow values which would move you upwards. Similarly, if you are at the bottom of the screen (X = 22) then values which move downwards are disallowed. (The use of a coordinate system makes it a simple matter to prevent the player moving off the edges of the screen (Y = 1 or Y = 21). Lastly the routine checks that the square is not already occupied. I have defined two functions FNX and FNY for evaluating vlaues of X and Y for any move.

```
FNX(M)=(M=1 OR M=5 OR M=6)-(M>6 OR M=4)
```

It looks pretty strange doesn't it — never mind, all will become clear shortly.

When Basic comes across a statement such as T=(M=2) it assesses the 'truth' of the expression in brackets — rather like an IF statement. If the expression (M=2) is false, then T=0: if the expression is true (if M does equal 2) then T = −1

Now, if we refer back to the table of M values we find that the three keys which move upwards give M values of 1,5 and 6. So if the player wishes to move upwards, the expression (M=1 OR M=5 OR M=6) will be true but the expression (M> OR M=4) will be false; and therefore FNX will be (−1)−(0) = −1.

Suppose on the other hand, that we wanted to move down, M could not be 4,7 or 8 with the result that FNX would be (0)−(−1) = +1. Thus X+FNX(M) will give us the new X coordinate for any move. FNY works similarly to give −1 for moves left, and +1 for moves right. Defining functions has the advantage that it uses less memory than a series of IF statements.

Having evaluated the move, the program then proceeds to 6000 which actually makes the moves. Again space has been saved by one routine to move both you and the rhinos. The variable CH determines which character is printed. The main part of the program now stores your present position and it is the rhinos' turn.

The first job is to see whether your move has brought you into a rhino's line of vision. For each of the rhinos that are still hidden, the program calculates the number of squares between you and the rhino. Line 7030 determines the direction using the principles we have just described. If (XH <X) is true then the rhino is below you, if (YH<Y) is true the rhino is to your right. In this example D would become −23.

If we added D to the rhino's present address, we would get the address of the square diagonally upward and to the left of the present square. The program checks each square along this line until it reaches the square you are on. If it has not encountered any obstacles, it has you in its sights and may appear. The RN flag is set and the subroutine at 7100 produces the ominous "uh-uhh" noise.

If no new rhinos have appeared the program will move to the routine at 8000 which calculates the best move for each rhino. It does this by considering each of the possible moves, rejecting those which are directly blocked and also rejecting moves along a diagonal, if there is a tree on either side. The routine then calculates how far away you would be if it made that move, selecting the one which brings it closest to its prey.

The algorithm is fairly effective, but now that you know how it works, you can use it to your advantage to trap the rhinos. For example, look at the diagram: the rhino won't be able to circle round the trees, so you can amble on home while the nasty little thing fumes with frustration!

VIC RHINO will fit into the standard 5K Vic, but will also run on expanded Vic's with no alterations necessary.

Rhino Listing

```
1000 NR=3:D=7680:CO=30720:S1=36874:S2=S1+1:POKES1+4,10:POKE36879,10:DIMR(20,2),V
(20)
1001 IFPEEK(4096)<>0THEND=4096:CO=37892
1002 DEFFNA(X)=D+22*X:DEFFNX(M)=(M=1ORM=5ORM=6)-(M>6ORM=4)
1003 DEFFNY(M)=(M=2ORM=5ORM=7)-(M=6ORM=3ORM=8):PRINT"   RHINO  ":GOSUB7300
1005 GOSUB2000
1010 NF=0:X=XH:Y=YH:CH=46:CL=1:POKE198,0:PRINT"   YOUR MOVE  "
1015 GETG$:IFG$<" "ORG$>" "THEN1015
1018 PRINT"                 ":GOSUB5000:GOSUB6000:RN=0:XH=X:YH=Y:CH=94:CL=2:IFW>0THEN
1990
1020 IFNF<NRTHENGOSUB7000
1021 IFRN=0THENGOSUB8000:IFW>0THEN1970
1024 IFRN=0THEN1010
1025 FORI=1TO10:IFV(I)=0THEN1010
1030 AD=D+22*R(I,1)+R(I,2):POKEAD,94:POKEAD+CO,2:NF=NF+1:NEXTI:GOTO1010
1970 X=XH:AD=FNA(X)+YH-1:POKEAD,26:POKEAD+1,1:POKEAD+2,16:T=50:C=0
1972 FORP=0TO2:POKEAD+CO+P,C:NEXTP:C=2*(C=2):POKES1,129+3*C:T=T-1:IFT>0THEN1972
1974 POKES1,0:PRINT"   CRUNCHED";:GOTO1996
1990 PRINT"   SAFE HOME";:GOSUB7300:IFNR<20THENNR=NR+1
1996 PRINT":  ANOTHER GO? ":POKE198,0:W=0
1997 GETG$:IFG$<>"Y"ANDG$<>"N"THEN1997
1998 IFG$="Y"THEN1005
1999 POKE36879,27:PRINT" ":END
2000 PRINT"   THERE ARE"NR"RHINOS"
2010 FORI=D+22TOD+483:IFRND(1)>.75THENPOKEI+CO,5:POKEI,160
2012 NEXTI
2014 FORI=D+30TOD+118STEP22:FORJ=1TO5:POKEI+J,32:NEXTJ,I
2015 FORI=D+426TOD+470STEP22:FORJ=1TO5:POKEI+J,32:NEXTJ,I
2018 POKED+77,8:POKED+77+CO,1:POKED+473,46:POKED+473+CO,1
2060 FORC=1TONR
2070 Y=INT(RND(1)*16+3):X=INT(RND(1)*8+8):R(C,1)=X:R(C,2)=Y
2075 IFPEEK(FNA(X)+Y)<>32THEN2070
2080 NEXTC:FORC=1TO10:V(C)=0:NEXT:XH=21:YH=11:RETURN
5000 A=AD:M=ASC(G$)-132
5100 IF(X=1AND(M=1ORM=5ORM=6))OR(X=22AND(M=4ORM>6))THEN5110
5102 IF(Y=0AND(M=2ORM=5ORM=7))OR(Y=21AND(M=6ORM=3ORM=8))THEN5110
5104 AD=FNA(X+FNX(M))+Y+FNY(M)
5106 T=PEEK(AD):IFT<>32THENW=2:IFT<>8THEN5110
5108 RETURN
5110 AD=A:M=0:W=0:RETURN
6000 AD=FNA(X)+Y:POKEAD,32
6010 X=X+FNX(M):Y=Y+FNY(M)
6020 AD=FNA(X)+Y:POKEAD,CH:POKEAD+CO,CL:RETURN
7000 FORI=NF+1TONR:X=R(I,1):Y=R(I,2):IFV(I)=1THEN7060
7010 N=ABS(X-XH):IFN=0THENN=ABS(Y-YH)
7030 D=22*(XH<X)-22*(X<XH)+(YH<Y)-(Y<YH)
7040 FORJ=1TON:AD=FNA(X)+Y+J*D:IFPEEK(AD)=160THEN7060
```

```
7050 NEXTJ:IFPEEK(AD)=46THENRN=1:V(I)=1:GOSUB7100
7060 NEXTI:I=1:IFRN=0THENRETURN
7070 IFI=NR+1THENRETURN
7080 IFV(I)=0ORI=1ORV(I-1)=1THENI=I+1:GOTO7070
7090 V(I-1)=1:V(I)=0:SX=R(I-1,1):R(I-1,1)=R(I,1):R(I,1)=SX
7095 SY=R(I-1,2):R(I-1,2)=R(I,2):R(I,2)=SY:I=I-1:GOTO7080
7100 S=150:D=8:GOSUB7200:GOSUB7200:S=130:D=16:GOTO7200
7200 POKES2,S:POKE162,0:WAIT162,D:POKES2,0:S=0:RETURN
7300 D=3:S=231:GOSUB7200:GOSUB7200:Q=Q+1:IFQ=3THEN7310
7305 S=229:GOSUB7200:GOSUB7200:GOTO7300
7310 Q=0:S=223:GOSUB7200:GOSUB7200:S=228:GOSUB7200:GOSUB7200:S=225:GOSUB7200:GOS
UB7200
7320 S=219:D=32:GOTO7200
8000 FORI=1TONR:X=R(I,1):Y=R(I,2):IFV(I)=0THENRETURN
8020 FORM=0TO8:RM(M)=PEEK(FNA(X+FNX(M))+Y+FNY(M)):NEXTM
8030 A=3:B=1:C=6:GOSUB8800:B=4:C=8:GOSUB8800:A=2:C=7:GOSUB8800:B=1:C=5:GOSUB8800
8040 FORM=0TO8:IFRM(M)=46THENW=1:GOTO6000
8041 IFRM(M)<>32THEN8044
8042 RM(M)=ABS(X-XH+FNX(M))+ABS(Y-YH+FNY(M))
8044 NEXTM:MV=110:M=0
8046 FORJ=0TO8:IFRM(J)<=MVTHENM=J:MV=RM(J)
8048 NEXTJ:GOSUB6000:IFW>0THENRETURN
8050 R(I,1)=X:R(I,2)=Y:NEXTI:RETURN
8800 IFRM(A)=160ANDRM(B)=160THENRM(C)=160
8810 RETURN
```

VICALC Assessed

By David Pocock

A couple of days ago a cassette labelled ViCalc landed on my desk and a disembodied voice from somwhere in the distanced asked 'Could you do a review of that for **Vic Computing**?' Having nothing better to do knowing that this would at last mean an opportunity to use a Vic, and eager to see my name in print I jumped at the chance.

The ViCalc cassette comes well packed with a cartoon picture on the front that could have been a picture of my desk at ten past nine on a Monday morning (a mess). A nice feature for those who remember the original Pet cassette decks, although not required anymore with the vastly more reliable cassette decks now being sold for the Vic, was a helpful notice to the effect that the ViCalc programme was recorded twice on each side of the cassette. As it happens, the copy I had loaded first time without the slightest hint of a problem.

The inlay card for the cassette opens out into a five-page instruction booklet. At this point came my first (and in fact, only) major problem with the package: The Fine Print. Trying to run the program while sitting very close to a bright and large (14in) television set, plus my normally poor eyesight, meant that the instructions were slightly difficult to read. Reading them later without the Vic to distract me proved a lot easier.

The instructions themselves laid out well, and they are simple to learn. Moral: always read the instructions (although I admit that I rarely do and not often that I have the instructions).

What happens in ViCalc

VicCalc has four data registers (A to D) and ten memory registers (0 to 9) which are constantly displayed on the screen. The program is written in Basic but this did not prove to make it all that slow — I could not out-type it and I don't expect that many Vic userswill manage to do so.

Data entry is extremely simple: just type in a number followed by RETURN and it is entered into the lowest data register (A). Any numbers in the registers are moved up to make room. There are five types of operation:

Binary Operations These all leave their result in data register A and clear the contents of register B by moving the other registers down. The operations available are

B+A
B−A
B*A
B/A
A^B
A+(B/100)
−A

> ViCalc is a cassette that turns the Vic into a programmable calculator. It gives you ten memories displayed on the screen along with four working registers, as well as the usual four-function arithmetic you get single-key commands for compound interest calculation, percentages, and several scientific functions. It works on an unexpanded Vic — all you need is the computer, a TV, and a cassette deck. ViCalc costs £8.99 from dealers or direct from Audiogenic (0734 595269).

Functions There are eight functions defined. All leave the result in data register A

A^e
ln A
LOG (base) A base is stored in memory register 9
Square Root A
Random (0) gives a value n where 0 n 1 the data register are moved up
SIN (A) angles are
COS (A) measured
TAN (A) in radians

Memory Operations These operations work on on (or all) of the ten memory registers:

Clear all ten memory registers
Clear one register
Transfer A to register
Transfer register to A
Swap A with register

The following all leave the result in the original memory register which you specify.

REGISTER + A
REGISTER − A
REGISTER * A
REGISTER / A
REGISTER

Fnancial Operations The two financial functions provided are Compound Interest Earnings and Amortization. Data for these is entered into data registers A and B (and C as well for Amortiztion calculation). The computations are performed for ten periods; these are displayed in the ten memory registers. It is only a matter of pressing a single key to calculate the next ten periods.

Stack Operations These move the contents around or clear the registers:

Roll registers up
Roll registers down
Swap A and B
Clear A
Clear all

Setting Precision You may set the number of decimal places to be displayed. All calculations are performed to the maximum precision (nine places, which seem familiar). You may however wish to display only one part of this data — to two decimal places, for instance. All numbers displayed are corrected for that precision, so 9.987 displayed to two decimal places will appear as 9.99

What can you do with ViCalc?

Ideas on a postcard please. No, seriously, apart from calculating that the payment for this article invested at 10 per cent per annum for ten years works out at five million (does anyone know where to store that many peanuts?) it's definitely the case that ViCalc can perform like a real value-for-money calculator.

As an example of how to use ViCalc, I'll go through the steps to perform a compound interest calculation. The aim is to calculate the final value of £1000 invested for 10 years at a yearly interest rate of 11.5 per cent compound annually

Enter 5000 (RETURN) —the initial investment
Enter 11.5 (RETURN) —the interest rate
Enter I —to calculate the interest for 10 periods

The ten memory registers hold the initial value (0) and the compound values for nine periods (1-9). Data Register C holds the number of periods calcualted (10), B holds the next principal balance (the figure you want) and A holds the interest rate (11.5).

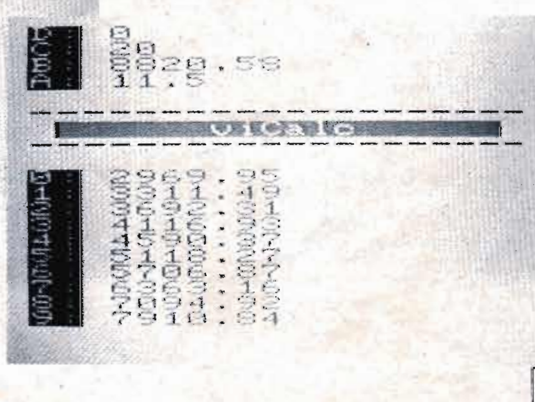
If you press 'I' again the next ten periods are calculated taking the value in B as the initial balance. So C will now hold 20.

Further periods can be calculated by pressing 'I' until the desired number of periods is shown in data register C.

If you wish to work out the value of your investment after a period which is not a multiple of ten, the value can be found by looking at the appropriate memory register. Press 'I' until C is greater than the desired number of periods, take 10 off the value in C and subtract that value from the number of periods required. Then look in the memory register corresponding to the remainder.

ViCalc is a useful package for anyone who is just starting with Vic and who wants to be able to do some calculations quickly and conveniently — particularly if they do not have the time or patience to learn.

Most of the functions are good though I am still puzzled about RANDOM and why it was included. Apart from that, ViCalc is a well-thought-out program — easy to use and well-documented (provided you own a magnifying glass).



This is your part of the magazine: Victuals consists of programs from readers. We love getting them, so send in what you regard as your best work — provided the programs come on cassette, not as handwritten or typed listings (transcription errors are the bane of all computer magazines) and not as output from the Vic printer (we wouldn't have the time to key in and check out all the programs we get).

Bouncy

Julian Rosen of Southsea sent us a fiendishly difficult game called Bouncy which he's modified from an original Pet program (you see, it can be done). It sounds simple enough: a ball is bouncing around a square and you have to direct it at numbered targets in sequence, aiming to hit as many targets as possible before your time limit of one minute expires. But just you try it . . .

```
5 REM*BOUNCY*
6 REM*FOR VIC 20*
7 REM*BY J.ROSEN*
10 GOSUB600:PRINT"■"
20 POKE36879,45
30 A=7680:T=1:P=22:Y=100:Z=101
40 FORI=1TO20:A=A+T:POKEA,Y:NEXT
50 A=A+T
60 FORI=1TO21:A=A+P:POKEA,Z:NEXT
70 A=A+P:IFT<0THEN90
80 T=-1:P=-22:Y=99:Z=103:GOTO40
90 POKE36878,15:POKE36876,235:POKE36876,238:POKE36876,0
100 L=7792:C=0:B=1:P=0:T=0:Y=176:X=TI
110 Z=INT(470*RND(1))+A+22
120 IF(Z=L-B)OR(PEEK(Z)<>32)THEN110
130 Y=Y+1:POKEZ,Y
140 REM*GET KEY TO PRINT PADDLE*
150 GETA$:IFA$="Z"ORA$="M"THEN300
170 K=PEEK(L+B):IFK>32THEN200
180 POKEL,32
190 L=L+B:POKEL,81:GOTO150
200 IFTI-X>3600THEN840
205 IFK>78THEN250
210 C=SGN(B)*(23-ABS(B))
220 Z=87:IFK=78THENC=-C:Z=43
230 GOSUB830:IFPEEK(L+B+C)<>32THEN180
240 POKEL,32:L=L+B:B=C:GOTO190
250 IFK=YTHEN270
260 B=-B:Z=61:GOSUB830:GOTO180
270 Z=225:FORJ=1TO9:Z=390-Z:GOSUB830:NEXT
280 T=T+1:GOTO110
300 C=SGN(B)*(23-ABS(B))
305 REM*DEFINE PADDLE*
310 Z=78:IFA$="Z"THENC=-C:Z=38
320 IFPEEK(L+C)<>32THEN180
325 REM*PRINT PADDLE*
330 POKEL,78+(A$="M"):GOSUB830
340 P=P+1:B=C:GOTO190
590 REM*INSTRUCTIONS*
600 PRINT"■■■■■■***■BOUNCY■***"
610 GOSUB970
615 PRINT"■■■■▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
620 PRINT"■■■■■THE ULTIMATE"
625 PRINT"■■■■■GAME OF SKILL"
626 PRINT"■■■■▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
627 PRINT"■■■IIIII■"SPC(13)"■":PRINT"■■■■"SPC(13)"■":PRINT"■■■■"SPC(13)"■"
628 PRINT"■■■■"SPC(13)"■"
629 PRINTSPC(9)"■■■■■■■\ /":PRINTSPC(10)"■■":PRINTSPC(9)"■/ \":PRINTSPC(10)"■■1■
"
630 GOSUB970
640 PRINT"■■BOUNCY GIVES YOU":PRINT"■A TIME OF ONE MINUTE"
645 PRINT"■TO HIT AS MANY":
650 PRINT"■TARGETS AS POSSIBLE.":PRINT"■YOU MUST HIT THEM "
655 PRINT"IN NUMERICAL ORDER,":PRINT"■■■■STARTING WITH"
660 PRINT"■■■■'■1■' BY PLACING":PRINT"PADDLES IN THE PATH"
670 PRINT"■OF THE BALL.■"
680 PRINT" 'Z' GIVES YOU A /"
690 PRINT"■ 'M' GIVES YOU A \"
700 PRINT"■■■PRESS SPACE TO PLAY■"
```

```
710 GOSUB1000:RETURN
820 REM*BALL NOISE*
830 POKE36878,15:POKE36876,240:FORI=1TO11:NEXTI:POKE36876,0:RETURN
840 FORZ=250TO4STEP-3:GOSUB830:NEXT
845 REM*SCREEN COLOUR*
850 POKE36879,45:PRINT"ЖЖ"       "
860 PRINT"ЖЖЖЖЖЖЖЖGAME OVER"
870 PRINT"ЖЖЖЖЖЖЖЖ            ":GOSUB970
880 PRINT"ЖЖЖЖ YOU HIT";T;"TARGETS"
890 PRINT"Ж YOU USED";P;"PADDLES"
900 PRINT"ЖBOUNCY GIVES YOU A":PRINT"SCORE OF";INT(8000*T/(P+70)/10);"%"
905 FORI=1TO9:GETA$:NEXT
910 PRINT"ЖЖЖ ANOTHER GAME (Y/N)";:INPUTA$
920 IFLEFT$(A$,1)<>"Y"THENPOKE36878,0:POKE36879,27:/"Ж":END
930 PRINT"Ж":POKE36879,45:GOTO30
970 FORI=1TO3700:NEXT:RETURN
980 GETA$:IFA$<>""THEN980
990 RETURN
1000 IFPEEK(197)=32THENRETURN
1010 GOTO1000
```

Raspberry

Well, this one isn't going to win any prizes. But as author John Wilson of Edinburgh puts it, "the program is written for the very young computer user — specifically my two-year-old daughter". And 'the instructions are verbal — 'press the space bar' — so as not to spoil the display".

That's just about all the information you need. It's very short; so try it on your own juvenile acquaintances.

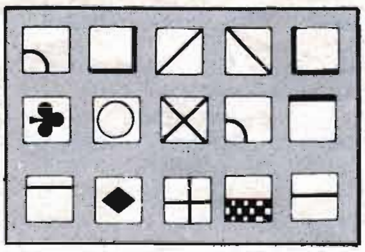
```
1 REM  RASPBERRY
2 REM
3 REM  J.N.WILSON
5 PRINT"Ж"
10 PRINT"        ┌┴┴┴┴┴┴┴┐
20 PRINT"        |       |
30 PRINT"        |—    — |
40 PRINT"       [ O    O  ]
50 PRINT"       [ O    O  ]
60 PRINT"        |   !!   |
70 PRINT"        |   v    |
80 PRINT"        |        |
90 PRINT"        |  \_/   |
95 IFX/2<>INT(X/2)GOTO200
110 PRINT"       \        /
120 PRINT"        \      /
130 PRINT"         \____/
140 PRINT"
141 R0=36874:R3=36877:R4=36878:POKER4,8
143 IFX/2<>INT(X/2)GOTO300
150 GETA$:IFA$=" "THENX=X+1:PRINT"Ж":GOTO10
155 GOTO150
200 PRINT"       \   v   /":GOTO120
300 FORI=1TO20
310 POKER0,135:FORJ=1TO10:NEXT:POKER0,0
315 POKER3,180:FORJ=1TO10:NEXT:POKER3,0
320 NEXTI
330 PRINT"Ж":GETB$:IFB$<>" "GOTO330
340 X=X+1:GOTO10
```

Repeating patterns

These small but hypnotically effective programs comes from S Pidd in York. They are not taxing to key in: nor are they particularly taxing to play. But they produce some really excellent combinations of shape and colour: we tried it on the people who design **Vic Computing,** and they were open-mouthed in admiration (though truth to tell that may have been the drink).

```
1 PRINT"Ж":PRINT"TO STOP, USE RUN/STOP
2 PRINT"TO PAUSE, TYPE @
3 PRINT"TO START OR RESTART,
4 PRINT"TYPE @
5 A=RND(-TI)
6 GETQ$:IFQ$=""THEN6
10 PRINT"Ж":P=160
12 REM      GETX$:IFX$="@"THEN240
13 POKE36879,INT(RND(1)*256)
15 FORI=1TO20
20 X=INT(RND(1)*5+1)
30 Y=INT(RND(1)*5+1)
40 P=INT(RND(1)*63+64)
50 CL=INT(RND(1)*8)
100 REM
200 FORK=X-1TO21STEP5
210 FORJ=(Y-1)*22TO484STEP110
215 POKE38400+K+J,CL
220 POKE7680+J+K,P
230 NEXTJ,K,I:FORK=1TO9999:NEXT:REM GOTO10
250 GETZ$:IFZ$<>"@"THEN10
260 GETGT$:IFGT$=""THEN260
270 GOTO10
299 REM**************
300 REM RANDOM REPEATING PATTERN
305 REM**************
310 REM S PIDD
320 REM 8 GALEGARTH
330 REM ALNE
340 REM YORK
350 REM YO6 2LQ
```

```
1 PRINT"ƆTYPE @ TO START
2 PRINT"EACH PATTERN.
3 GOTO190
10 X=RND(-TI)
11 PRINT"Ɔ":X%=RND(1)*256
15 POKE36879,X%
16 X=11:Y=11
20 KK=RND(1)*20+15
30 DX=INT(RND(1)*11)
40 DY=INT(RND(1)*11)
50 N=INT(RND(1)*63+64)
51 IFN=81ORN=87ORN=90ORN=91ORN=96ORN=102ORN=209ORN=215ORN=219ORN=218THEN60
52 IFN=32ORN=160ORN=214ORN=240THEN60
53 IFN=35ORN=43ORN=42ORN=48ORN=15ORN=143ORN=163ORN=170ORN=171ORN=176THEN160
54 IFN=86ORN=224ORN=230THEN60
55 GOTO50
60 C=INT(RND(1)*8)
61 GOSUB70
62 Z=DX:DX=DY:DY=Z:GOSUB70
65 GOTO180
70 Q%=X+DX+22*(Y+DY)
80 POKE7680+Q%,N:POKE38400+Q%,C
90 Q%=X-DX+22*(Y-DY)
100 POKE7680+Q%,N:POKE38400+Q%,C
105 Q%=X+DX+22*(Y-DY)
110 POKE7680+Q%,N:POKE38400+Q%,C
120 Q%=X-DX+22*(Y+DY)
130 POKE7680+Q%,N:POKE38400+Q%,C
150 FORK=1TORND(1)*1000+500
160 REMGETA$:IFA$=""THEN160
170 RETURN
180 W=W+1:IFW=<KKTHEN30
190 GETA$:IFA$=""THEN190
200 CLR: GOTO10
299 REM
300 REM***************
310 REM RADIAL RANDOM
320 REM PATTERN
330 REM***************
340 REM
350 REM S PIDD
360 REM 8 GALEGARTH
370 REM ALNE
380 REM YORK
390 REM YO6 2LQ
```



```
1 PRINT"ƆTO START NEW PATTERN, TYPE SPACE
2 INPUT"■DATA IN YET (Y/N)";A$
3 IFA$="Y"THEN10
4 PRINT"Ɔ":FORK=1TO5
5 PRINT200+K;" DATA";CHR$(34);"      ";CHR$(34)
6 NEXTK
7 PRINT
8 GOSUB 400        :END
10 REM
20 FORK=1TO5
30 READA$(K):NEXT
40 PRINT"Ɔ"
50 FORL=1TO4
60 FORJ=1TO5
70 FORK=1TO4
80 PRINTA$(J);:NEXT:PRINT
90 NEXT:NEXT
100 GETZ$:IFZ$=""THEN100
110 GOTO1
201 DATA"▔▔▔▔▔"
202 DATA" * * "
203 DATA"  *  "
204 DATA"*   *"
205 DATA" * * "
400 PRINT"USING CURSORS,TYPE
401 PRINT"YOUR PATTERN IN THE
402 PRINT"BLOCK ENCLOSED BY
403 PRINT"QUOTE MARKS, TYPE
404 PRINT"HOME AND ENOUGH
405 PRINT"RETURNS TO ENTER THE
406 PRINT"LINES. THEN RUN10
407 RETURN
500 REM
510 REM***************
520 REM REPEATING PATTERN
530 REM***************
540 REM
550 REM S PIDD
560 REM 8 GALEGARTH
570 REM ALNE
580 REM YORK
590 REM YO6 2LQ
```

How to create your own graphics

Design and build your very own Vic characters: this month Julian Ravest tells you with a canter through the principals of character formation

The Vic, like Commodore's other computer, the Pet, sports a wide range of graphic characters: and you can access them from the keyboard, using them in game displays, graphs and the like. But they are chunky, suffering from the limited number of character positions on the screen (23 lines of 22 positions per line).

On the other hand, user-defined characters can greatly enhance a screen display. Consider the appeal of shooting down alien bodies as oposed to firing at mere filled-in circles. With the Vic it is possible to define shapes of aliens — or Greek letters, Egyptian hieroglyphs or whatever you need — and to use those shapes as easily as any other letter (or any other Commodore graphic shape).

The trick lies in the manner in which the Vic uses address 36869.

Normal Operation

Address 36869 is the fifth of 15 registers which the Vic uses to control its displays and sounds. This register, R5, does two things. It determines the area of memory that it will be using as the screen map: more significant for us, it also defines where the computer is to look in its memory for the character table. For example, in the 3.5K Vic you'll find that R5 normally contains the number 240.

With bit 7 of register 2, this tells the computer that the screen map is to start at address 7680 while the characters are located in ROM from address 32768. The instruction POKE 7680,0 will result in the first character (which happens to be '@') being displayed in first position on the screen, the top left corner.

Of course, the @ will not be visible unless the corresponding position in the screen colour map (38400) contains a colour code to distinguish the shape from the background. The command POKE 38400,6 will suffice for this.

Similarly POKE 7680,1 will cause an 'A' (which is the next character) to be displayed.

Each of these characters is defined by sets of eight numbers to be found in the character table — the diagram shows how it works for these two characters. The numbers are stored in their binary form; and the resulting pattern of ones and zeros is interpreted by the computer as the shape to be displayed, a zero being shown on the screen as a picture element (or 'pixel') in the background colour map while a one appears as a pixel in the colour specified in the screen colour map.

In this way the numbers 28, 34, 74, 86, 76, 32, 30, 0 (the first eight numbers of the standard character table) define the shape.

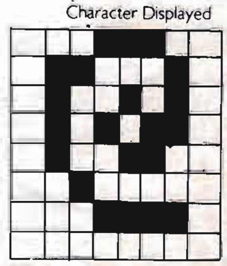
A number found POKEd to the screen map indicates which group of eight numbers in the character table (ie which character) is to be printed on the screen.

FIG 1 **STANDARD CHARACTER SET**

Starting location 32768. The first two characters and their method of representation are shown.

Location	Number Stored	Binary Form
32768	28	0 0 0 1 1 1 0 0
32769	34	0 1 1 0 0 0 1 0
32770	74	0 1 0 0 1 0 1 0
32771	85	0 1 0 1 0 1 1 0
32772	76	0 1 0 0 1 1 0 0
32773	32	0 0 1 0 0 0 0 0
32774	30	0 0 0 1 1 1 1 0
32775	0	0 0 0 0 0 0 0 0
32776	24	0 0 0 1 1 0 0 0
32777	36	0 0 1 0 0 1 0 0
32778	66	0 1 0 0 0 0 1 0
32779	126	0 1 1 1 1 1 1 0
32780	66	0 1 0 0 0 0 1 0
32781	66	0 1 0 0 0 0 1 0
32782	66	0 1 0 0 0 0 1 0
32733	0	0 0 0 0 0 0 0 0

Character Displayed



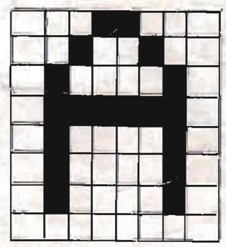
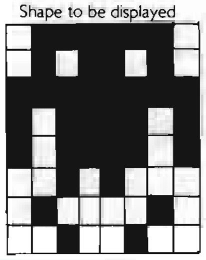


FIG 2 **STAGES IN CODING AN ALIEN SHAPE**

Shape to be displayed



Binary form of shape	Decimal Number
0 1 1 1 1 1 1 0	126
0 1 0 1 1 0 1 0	90
1 1 1 1 1 1 1 1	255
1 0 1 1 1 1 0 1	189
1 0 1 1 1 1 0 1	189
0 0 1 0 1 0 0 0	36
0 1 0 0 0 0 1 0	66
0 0 1 0 0 1 0 0	36

FIG 3 LOCATIONS OF START OF CHARACTER TABLE FOR SOME USEFUL REGISTER/VALUES

Value in Register 5	Start of Character Table	
	Decimal	Page No.
225	7168	28
254	6144	24
253	5120	20
252	4096	16

The use of the 255 character table is limited to 64 shapes since the screen map starts at 7680.

Changing the table location

The area of memory to which the computer will look for the table may be altered by POKEing R5. For example, POKE 36869,255 will change the character table start to 7168, an area in RAM. The screen map is unaltered.

We can now build up our own shapes in successive groups of eight numbers in the new table area. The first step in the construction of an 'alien' form is to draw the desired shape on an 8x8 grid. This is changed into its binary equivalent, with a '1' for each filled square and a '0' for each blank.

This binary number is then converted to the corresponding decimal number, and the resulting sequence can be entered into the new shape table by means of a simple program like this:

```
10 DATA 126,90,255,189,189,36,66,36
20 FOR A = 7168 TO 7175
30 READ S
40 POKE A,S
50 NEXT
60 POKE 36869,255
```

When this program is RUN all the letters on the screen will change to arbitrary patterns of dots. But pressing @ on the keyboard displays a somewhat bandy-legged alien, the only shape we have so far defined.

POKE 38400,6 : POKE 7680,0 puts the alien in the top left position of the screen.

You can restore the normal character set by using POKE 36869,240 or more simply by pressing the RUN/STOP and RESTORE keys at the same time.

Animation

A simple way to produce animation is to alternate two or more shapes in the same position. Try this:

```
10 DATA 126,90,255,189,36,66,36
20 DATA 126,90,255,189,189,36,66,129
30 FOR A = 7168 TO 7183 : READ S : POKE A,S : NEXT
40 FOR A = 7168+32*8 TO 7168+32*8+7 : POKE A,S : NEXT
50 POKE 36869,255
60 PRINT "clr"
70 POKE 7680, 0
80 FOR T=1 TO 100 : NEXT
90 POKE 7680, 1
100 FOR T=1 TO 100 : NEXT
110 GOTO 70
```

Line 60 places the value of '32' into each screen position to print a blank; line 40 ensures that '32' specifies a blank. Lines 80 and 100 are simple delay loops.

An alternative way to produce animation is to alter the character table. This can be very effective when the shape to be altered is present in many screen locations. As an extreme example, consider the case of our alien occupying every screen location:

```
10 DATA 126,90,255,189,36,66,36
20 FOR A = 7168 TO 7175 : READ S : POKE A,S : NEXT
30 POKE 36869,255
40 FOR A = 7680 TO 7680+505
50 POKE A, 0
60 POKE A + 30720, 6
70 NEXT
80 POKE 7175, 129
90 FOR T = 1 TO 100 : NEXT
100 POKE 7175, 36
110 FOR T = 1 TO 100: NEXT
120 GOTO 90
```

Here line 60 fills the screen colour map with blue (7680 + 3027 = 38400). Lines 80 and 100 change the character being displayed at every screen location — causing 506 aliens to waggle their legs simultaneously.

Letter Plus New Shapes

In some circumstances it may be desirable to have both the normal letters and numbers on the screen at the same time as your own user-defined shapes. An easy way to do this is to copy the required part of the standard set to the position of the new table and to replace certain characters or add new ones.

Some POKEs for R5 that give usefully placed character tables are shown below.

In cases where a clash over memory use may occur it is advisable to protect the table by POKEing the top of memory allowed into locations 51 and 52. For example, to protect a table starting at 7168 (i.e. page 28) you will find that POKE 52,28 is sufficient.

Beelines give your VIC a 40 column display with 32K of extra RAM as well!

12 months parts and labour guarantee.

PRICE:

Nett	VAT	TOTAL
£220.00	£33.00	£253.00

The Colour Writer VIC Expansion gives you the most cost effective way of expanding your VIC to 35K AND we give you the added bonus of a 40 Column display free of charge. Now you can have the luxury of a 32K Pet AND Colour to enhance your programs. The Colour screen consists of 960 characters in a 40 column by 24 row layout (to Viewdata standard) with the 25th line for status information. The programmable features include 7 foreground and background colours, flashing, double height characters, block graphics and any combination you care to name; and not content with a 35K VIC, we give you an extra motherboard slot for further expansion.

BEELINES (Bolton) LTD

FREEPOST (No stamp required)
Bolton BL3 6YZ
Telephone (0204) 382741 & 384599

24 hour answer phone: 0204 385299
Reg. office: 124 Newport St., Bolton BL3 6AB.
Dealer enquiries welcome.

Please send me:

Qty.	Item	Price
	Beelines VIC 35K + 40 Column Expansion Unit @ £220 + £33 VAT	

I enclose cheque/P.O. ☐
OR Please debit my:

Access **5224**
Barclaycard **4929**

Expiry Date ____
Name ____
Address ____
____ Code ____
Tel. (day) ____

Official orders welcome –
we will ship to approved account holders on receipt of firm order.
Delivery free within mainland UK.
Access and Barclaycard welcome.
Telephone answering machine for 24hr/7 day credit card orders **0204 385299**
Freepost: **Beelines, FREEPOST, Bolton BL3 6YZ**

VC3-82

CRA — Computer Retailers Association

Dear Vic

Will the Seikosha GP80 Printer work plugged straight into the Vic or does it need an interface?
L Higham, Peterlee, Co Durham

Well, of course all printers need an interface of some kind — an interface is basically the boundary between two parts of any system, and in computer terms it is usually a plug-and-socket connection that allows one thing to be used with another. Both have to agree on their electric and physical properties, so in effect the connectors — the plug and socket — have to match each other.

The Vic comes with one type of interface socket as standard: it's the socket at the back. What Mr Higham wants to know is whether the Seikosha printer (one of the cheapest matrix printers you can find at about £195) will run straight from it, or whether an interface adapter will be needed. This intermediary device would convert one type of interface into another, and vice versa.

The Vic connects a printer via the serial port, and the standard GP80 printer doesn't have this kind of serial interface. Instead it comes with another type, a parallel interface. That means it will either need some kind of interface converter; or you will need some extra software to drive it from the other Vic socket, the user port.

So has anyone out there done either of these to drive a Seikosha?

My Vic was built in Japan — and not Germany, as indicated in Vic Computing! I suspect that Commodore may have converted some models to meet the initial demand, but the fact is that the current models appear to be all of Japanese origin.

I had reason to write to David Briggs, Technical Support Manager for Commodore. He informs me that a 40-column upgrade is being developed for the Vic; it is not, however, a replacement ROM, but a plug-in board (with additional RAM probably). It will probably be available in "the very near future" according to Commodore — there has been a lot of rumour and comment in the recent microcomputer press as to the possible upgrade of the Vic-20.

Briggs also mentioned that it will be quite some time before the Vic-40 is officially announced and that it is going to be a fundamentally different machine to the '20. I suspect much of the confusion has arisen over the fact that the Vic-20 will not be upgraded to give all the facilities of the Vic-40, but only to give a 40-column screen format.

By the way, I have a copy of Tim Hartnell's Getting Acquainted with your Vic-20. While it is a useful book, one should be wary of the large number of printing errors in the programs which will cause repeated crashes unless amended. I feel that at that price a little more care could have been taken in the proof-reading stage, especially since it is aimed at the complete beginner.

The Vic-20 is an excellent machine and I think the idea of a magazine catering exclusively for it is marvellous.
C J Durham, Blandford Forum, Dorset

All compliments, comments and criticism (in descending order of welcomeness) are gratefully received. We tend to share that view of Tim Hartnell's book, a fully review of which is scheduled. But give the work its due — there is no alternative to it (untill we produce our Best of Vic computing, of course...)

We also have some sympathy, albeit no excuses, on the problems of reproducing program listings — it's not easy!

The Vics currently being shipped now are, indeed, being made in Germany. Our spies at Commodore say that to cater for initial demand, though, several batches of the Japanese models were marketed.

As for Mr Durham's comments about the Vic-20 and Vic-40 models, he's quite right. By the by, a number of 40-collumn upgrades are being developed by independent suppliers — but these will not be in the form of a ROM chip to put inside th Vic. Instead they'll be plug-in boards for the slot at the back.

May I say how much I enjoyed reading Vic Computing, and I look forward to the future issues. I have recently received my Vic and I must say that it was well worth the wait. In addition the 3K RAM provided by Commodore in recompense for the various delays was very well received.

As my Vic now has 6.5K of available memory, could you confirm that high-resolution graphics are now possible on my machine with enough memory left for a games program or whatever?

And is it possible to program the hi-resolution graphics directly from the keyboard? If so, can you help me do it?

One other point bothers me: the function (or lack of it) of the special function keys to the right of the keyboard. How do you address these keys into a program? Could each key be used as a one-key Basic instruction for programming, with f1 programmed to be 'PRINT', f2 'INPUT', and so on?
J W Hough, Crosby, Liverpool

Yes, with expansion memory you shouldn't have any problem running high-resolution graphics and another program simultaneously on your Vic: it depends on how long your programs are, of course. As for programming hi-res directly from the keyboard, see elsewhere in this issue of **Vic Computing**: we'll be publishing a type-in-and-go editor program in the next **Vic Computing**, too.

The matter of the function keys is one on which both the magazine and Commodore have had numerous enquiries. It is possible to address these keys into a program, of course, and in theory each of them could indeed be used as a one-key Basic instruction for programming — though that's actually harder. Having said that, I'm afraid you'll just have to wait for the articles we have commissioned.

In the first issue you mentioned the user port and a few of the ways in which it could be used. If I wanted to connect the user port to eight separate motors, what complete electronics would I need to place between the motors and user port? I would be grateful if you could send me a circuit diagram of the necessary electronics bewteen the port and eight motors.
S Delucia, London NW5

The user port on the Vic computer could indeed be used to switch on or off up to eight small electric motors. Sadly, though, it simply isn't possible to lay out a standard diagram of the electronics necessary between the user port and the motors: this will depend on the power and size of the motors involved. We would however expect that a number of independent suppliers will very shortly be selling low-cost interfaces for the Vic — interfaces which can be directly linked to light bulbs, electric motors, or whatever else you fancy.

If you do wish to interface a device yourself right now, we suggest you get in contact with someone who knows something about electronics. The output from the user port is 5 volts, but cannot deliver any significant current: you will therefore need a buffer, which most simply is an ordinary transistor between each line on the port and each motor.

It sounds like an interesting application, one that could well merit a write-up in Vic Computing.

I've had the Vic-20 for a week now and in that short time I have come to the conclusion that it's a winner: so here are a few random thoughts. If they seem obvious, or even untested, please bear in mind that I didn't have a Vic manual at the time!

It is possible to down-load programs to tape from a Pet. But keep in mind that the start of Basic is different from one machine to the other — you will need to save from the monitor. Someone gave me a disk full of

Send us comments, queries and complaints: we'll answer everything we can and print anything that isn't boring or illegal. Address yourself to The Editor, Vic Computing, 39-41 North Road, London N7 9DP

Vic programs for testing, and these were dumped to tape from the Pet. You'll be pleased to hear that the current cassette deck works OK with Vic.

Those of you who went to the Pet Show last summer will have seen the impressive band of sticky-fingered youths who were prodding at Vic games all day. That particular demo was run from ROM packs plugged into the expansion slot. My mole inside Commodore tells me they are using 2x4K chips and 8K chips as was first thought.

I've had the lid off the Vic: it's interesting to note that my version contained static RAM, and not a lot more besides a huge heatsink which gets boiling hot.

You'll be thrilled to hear that it's easy to crash the machine. My first program was full of odds and ends at the end, after loading from tape. Trying to remove offending lines produced a low-pitched tone and the character set displayed all over the TV, and the system locked out. Only solution: power off!

On the subject of TVs; I tried the Vic with three colour models and they were all superb — there was no instability or running of colours, although some sets work better with the AFC switch in the off position. There was a distinct lack of video rattle on the sound channel — full marks there.

It's possible to scroll the centre frame of the video by POKEing 36864 — that location scrolls left/right, 36865 scrolls up/down:

```
10 FORD = 1 TO 255
20 POKE 36864,D
30 FOR I = 1 TO 200: NEXTI:
REM SLOW IT DOWN
40 NEXTD
```

Somehow I managed to get the useable video box reduced to a small square in the top left-hand corner of the display, but I can't manage to repeat the effort again. Anyway, for what it's worth, it was location 36868.

Cursor flashing rate and hence repeat key speed can be altered by POKEing location 36967: the higher the number POKEd, the slower it becomes.

John Nuttall
University of Notre Dame, Indiana

As a novice to computing and the proud owner of a Vic I happily paid for a subscription to your magazine, the Christmas issue. I hope the standard of printing of the listings is not the shape of things to come. Was I the only reader who could not get the listings on 'alternate screens' to work?

It must be difficult to write for novices and experts in the same magazine. But as it is part of the sales pitch for Vic, why did Jim Butterfield not tell us how to program the eight function keys?

It would also be useful if not essential for you to print a complete list of the characters obtainable from the Vic and how they are obtained. This is not readily accesible in the manual and they often seem to look rather different on the screen from their printed representation.

Dr Dan Tunstall Pedoe, London E9

Taking those points in order: we are very sorry about the errors and now that the mag is past the infant stage things should vastly improve.

Articles on the function keys are scheduled — and by the way there are actually 12 functions available, not just eight: you can use the Commodore key as an extra shift. More about this anon.

I have written a couple of programs occupying approximately 3K each. They are simple games and incorporate an option for 'another go' — ie the program returns to start without being re-RUN. My problem is that after perhaps 20 trips through the program the computer gives the error message OUT OF MEMORY and always quotes the same line number.

John Brown, Harrow Weald, Middlesex

Tricky one, this, since it could be any number of problems. Perhaps the most likely answer is that you have an unclosed FOR-NEXT loop with a conditional GOTO jumping you out of it before it can complete. But there are many possibilities apart from that

Have you any information about machine-code proramming for RTTY reception in different speeds and codes (Baudot and ASCII)? One problem I have now is that I am not aware of the adresses of the ports used for the input of the RTTY signals coming from the demudulator.

Bo Carnerius, Sandviken, Sweden

Gulp — this one flummoxed us all in the office. Any contributions?

While awaiting the second issue of Vic Computing (and my Vic-20!) with itching fingers, I have been playing with my new Teletext TV set. It may interest my putative Vic comrade that there appears to be an experimental computer programming service on BBC's Ceefax and ITV's Oracle. Unfortunately they seem hard to 'page'.

The BBC1 programs are on pages 700 to 703. If you cannot get at pages 700 and beyond directly, try going via page 131. Don't ask me why, my set seems OK in other respects.

Similarly ITV Oracle's service is on page 175 and then 180 to 188. If you don't page successfully at first, keep on paging to see if you can break into these pages which do not appear on the standard Teletext index. Be warned though — it seems that if your TV set is not tuned correctly, the program symbols may be seriously corrupted. I'm not sure what value this information will be, I'm unable to evaluate the program myself because of lack of expertise. Any comments anyone?

I notice that advertisers of computers and computer-related products are becoming more and more outrageous in their claims; my latest Access catalogue informs me that Vic-20 "speaks English" so I "don't have to learn a special language"! I can't wait!

I really welcome your magazine: one devoted to the Vic seems especially welcome to a beginner who may be confused by too many cross-references to other machines, dialects of Basic, etc.

J D Collins, South Woodford, London E18

Mr Collin seems to be having a lot of fun exploring the pages of CEEFAX and ORACLE. You may be interested to learn that B&B Computers of Bolton has a 40-column add-on board for the Vic which copes with standard teletext graphics; we've asked for one to review, but they're on the market at around £100 right now.

We intend to carry an article on telesoftware in the near future, as it seems to be a rapidly developing area "Practical Computing's" project looks particularly interesting.

It is possible to access what appears to be the high resolution graphics on the basic Vic by using POKE 38400, 10R8 and similar combinations. Here's a program to illustrate this:

```
1 REM COPYRIGHT ROGER
COCKFIELD 1982
10 PRINT "clr"
20 FOR J=8 TO 12: PRINT J;
30 FOR K=8 TO 12: PRINT K;
40 FOR X=1 TO 255
50 POKE 7680+220=X,X: POKE
38400+220+X,J OR K
60 NEXT X
70 NEXT K
80 NEXT J
100 END
```

Note where the cursor finishes at the end of this program.

Roger Cockfield, Kettering, Northants