DERRICK

ORIC USER

ISSUE NUMBER 1

INSIDE:

All you need to make the most of your Oric

Programming for the beginner

News and reviews

Oric v. the rest

Six pages of software

NEW!

Contents

Editorial

ORIC USER

THE TEAM

Editorial:
Fin Fahey
Ian Williams
Frank Brecht

Software:
Tony Sellen
Tarquin Binary

Advertising:
Maggie MacDonald
01-833 1299

AT THE LAUNCH of the Oric it was announced that to break even Oric Products International would have to sell 250,000 micros. That is a very tall order. Sales on such a scale would put Oric Products up there in the big league with the Sinclairs, Apples and Commodores of the world. It would make the Oric a first division micro.

The Oric user would benefit greatly from such a large user base. It would mean that the big microcomputer software houses such as Bug-Byte, Quicksilver, and Imagine will be turning out exciting games programs to delight us. It means the book publishers will be printing vast libraries of books to enlighten us. It means that there will be hardware add-ons, business programs, modems, prestel pages, electronic mail and everything else that companies will only provide for large user bases.

The power of the consumer can be overwhelming. We can and we must use this power to ensure that we get the service we want from the forthcoming Oric Industry.

In the past the microcomputer industry has had a shocking reputation. Manufacturers have failed dismally on delivery, and software producers have often failed to deliver a good quality product.

We the users are the suppliers' bread and butter. To survive they must give us the service and the products we want. If you buy a book and it is inadequate, send it back to the publisher and demand your money back. If you buy an exciting adventure game that isn't exciting, complain to the advertising standards people. It is rumoured that almost half the complaints dealt with by the advertising standards people are from the microcomputer industry.

A quarter of a million machines must be sold this year. Oric Products say that there are already orders for 350,000. So there is a future for our micro. The big software houses are already advertising plans for Oric software, and the smaller ones are requesting would-be authors to submit programs for publishing.

Inevitably, after using other people's software for a while, you will find yourself thinking "I could do that". And, unlike poor Yosser Hughes, the chances are that you will be able to do it as well as the professionals. When that moment comes, and it might be nearer than you think, you will have to make a decision about how you are going to go about it. Whether you do it by yourself, or as part of a team, or even by working on a royalty basis for someone else, you will have a great advantage over the bigger competition. Because you will know better than they do just what Oric users want and expect from the product they are being sold. Good luck.

Typesetting and printing by
Calverts Press,
55 Mount Pleasant,
London WC1
01-278 7177

Published by:
Co-operative Computers &
Communications Ltd.
20 Wynford House,
Wynford Road,
London N1

Price war looms

THE ORIC PRICE structure is becoming increasingly complex. The 16K version recently had its mail order price hiked up to £129 — as advertised in issue one of Oric User, the official organ of Tangerine, the Machine's designers. As yet this 30% increase has not been reported anywhere else.

One week after this information became public W. H. Smiths cut the price of the Sinclair Spectrum micro to £130 for the 48K version and £99 for the 16K model. This preceded the official Sinclair price cut. About this time the future of the Oric, and our future as Oric users, looked bleak. It looked as though we could be left in the lurch as users of an interesting but forgotten machine.

We await with interest an official response from Oric Products, but in the meantime there are some signs of what is to come. Some Oric dealers are already discounting the 48K machine — the only version yet in existence. For example Twillstar Computers of Southall in Middlesex, telephone 01-574 5271, are now selling the machine for £155. Lower still is the £149.95 price offered by Galaset Ltd of Littlehampton, telephone 01-549 8229.

WP now on Oric

A WORD PROCESSOR for the Oric is on its way. The program, commissioned by Oric, will be available from the company shortly. It was written by Dr. John Dawson, who is best known to microcomputer users for his contributions to both Your Computer and Practical Computing magazines. As yet we don't know the price, but we do know that it will be, at least for the present, cassette-based. Dr. Dawson hopes the package will be purchased by home users, students and small businesses.

Hi-di-hi-Oric

THE INTERNATIONAL Computer Camp is both a holiday and an informal forum where people from all over the world interested in microcomputers can meet and discuss their hobby with one another. The camp is for everybody over the age of nine — old age pensioners qualify for a price reduction.

The campers will be living at the University of Southampton, each person occupying their own room. Food can be purchased nearby or cooked in the University kitchens. Special weeks are being arranged for users with more specialised microcomputer interests such as; Artists (who will be looking at computer graphics), teachers, motor traders, doctors, engineers, shopkeepers and a number of other groups.

If you would like more details about the computer camps contact Dr. Lionel Wardle, Computer Holidays, 37 University Road, Southampton. Telephone 0703 558621.

Backlog ends

ORIC PRODUCTS International has stopped accepting requests for Orics sold by mail order. The company is now selling the micros through a nationwide network of retail outlets. Among the big high street names stocking the Oric are W.H. Smiths, Dixons, Greens, Laskeys, Currys Micro C, and a number of more specialist shops such as the Spectrum retail chain of independent dealers.

There have been a number of complaints from the independents that they were not getting the machines they had ordered from Oric. This they felt was because Oric were giving priority to orders from the larger dealers.

The move to go over to retail-only sales of the 48K machine comes now that Oric have cleared the mail order backlog. Commenting on this, Barry Muncaster said "It is a great relief to all of us to be able to say that we currently hold no order that has been with us longer than 28 days".

New Oric boss

BARRY MUNCASTER — previously of Tangerine Computers, has been appointed managing director of Oric Products International, where he will be responsible for the day to day running of the company. The outgoing managing director, John Tullis, becomes executive chairman with responsibility for financial matters.

Oric User: Your Magazine

Wanted: your programs, letters, hints and tips

ORIC USER is your magazine. It doesn't matter who you are, or where you come from, all our readers have one thing in common, a thirst for more knowledge about the Oric in particular and about computing in general. You could be a beginner, or an expert, if you have something to say we want to hear.

The main sections of the magazine are outlined below, together with details of how to submit material for them. If you have an idea that doesn't fit one of our existing categories, fear not, we would still like to see it, who knows, it could be the start of something big.

In addition to wanting your submissions we want your questions — if it's to do with computing or the Oric we will try and answer. If the question is how to fend off Orcs in an adventure game (or even getting started) we will print the letter and await replies from fellow adventurers. Above all else remember: whatever it is, don't keep it to yourself, share it with us all.

Letters.

If you have something to say, we want to hear from you. You might want to moan about something, or praise something else — providing it isn't libelous we will try and print your grievance. If enough people moan together, they can often get things done.

Criticism, of software, products, even of Oric User is welcome. By listening to your ideas, manufacturers, publishers and even ourselves can provide you with a better service. You may want to express an opinion, about the square root of minus one, or the colour of your Oric, or answer someone else's letter. You can even use our letters page to get in touch with other Oric users in your area.

Ideally letters should be typed — but as long as your handwriting is clear we may still be able to use it. It is important to leave double spaces between the lines, and to only write on one side of uniform-sized sheets of paper. Don't forget to include your name and address.

Questions

These can be on any relevant matter and we will answer as many as possible. Please only send one question on each sheet of paper and put your name and address on each sheet.

Feature Programs and Software Bin

These are the two main programming sections of the magazine and they really are up to you. We are interested to see any software you have. No matter how simple, it will be of interest to some of our readers. Remember that it is an offence to copy programs out of other magazines and you may be prosecuted for doing this.

Until there is an Oric printer, we will accept handwritten listings, but would prefer software on tape. Remember to tell us what speed you used to record the program. Handwritten programs are more liable to error so please check these carefully before sending them.

The most important part of any software submission is the documentation. This can make or break an article. It would be nice if it could be sent typewritten, but again handwriting will do, on one side of the sheet only please, and with a lot of space on the page for our team to go to work on.

The Oric AND ITS RIVALS

The Oric 1 — Just in case you forgot what it looks like.

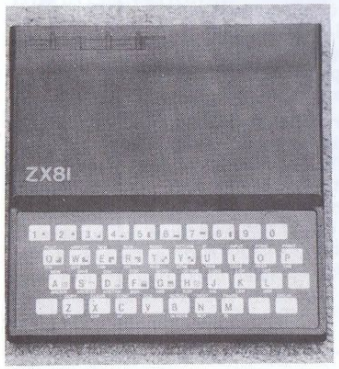
AS HOME computers go, the Oric is probably the best value machine on the market. How does it measure up to the competition?

To answer that question we must first decide what the competition is. Any home computer that costs less than £200 is a potential rival, and that leaves a lot of scope.

At the bottom of the range is the good old ZX81. This is the staunch old workhorse of home computers, and it has a lot of mileage left in it yet. I expect a lot of Oric owners will remember it affectionately as their first micro.

Today the ZX81 can be bought for less than £50 new, and secondhand machines are cheaper still. By the time the 16K RAM pack is added, the cost is nearer the £80 mark.

At £90, the Jupiter Ace can also be considered as a rival to the 16K Oric. Although it only has a basic memory size of 3K, its inherent speed — due to its native Forth interpreter in place of the standard Basic — will make it a possible upgrade micro for someone junking their ZX81. Of course Forth is available as a standard second language on the 48K Oric, but a tape based Forth is rarely as good as the Rom-based Forth in the Ace.



The ZX-81 — First of the cheap micros.

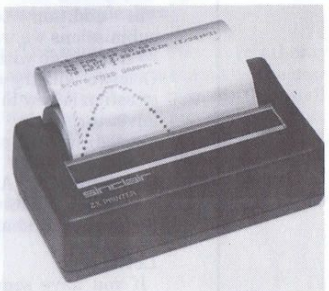
Acorn has gone on to greater things since the early days of the Atom. But the Atom is still selling in reasonable numbers, numbers that are high enough to make it a force to reckon with. The going rate for a 12K Atom without colour is around £120.

Atari is one of the biggest manufacturers of home computers, but has never really been much of a success in Britain. Now the price of the Atari 400 has dropped from its previously expensive level to a sensible £160 it might be worth examining. There is a wealth of games software for the machine as well as a huge variety of applications packages.

There is one major drawback with the Atari 400 and that is that it does not come with Basic as standard. Similar considerations apply to the Texas Instruments TI99/4, which doesn't have the same amount of software as the Atari.

Sinclair's ZX Spectrum is the obvious rival to the Oric. In fact the Oric was clearly conceived as a rival to the Spectrum. Both micros come in 16K and 48K versions and they are similarly priced. They are both designed for the same market and they will both sell thousands.

Commodore is another manufacturer that has been around a long time. Although the Vic-20 — which means something rude in German — only has 3K of memory, it has enjoyed a great deal of success in the past. Thanks to the worldwide success of the machine there is an excellent range of software from independent suppliers that makes the Vic at its current price of around £130 another competitor to be taken seriously.



Sinclair's Printer — we want one too!

From the Far East comes the Texet TX8000, a colour computer for around £167. It is actually being sold in a number of different versions, at least one of which will appear in this country. Although it has nothing like the performance or specification of the Oric, it has such a low price that it cannot be ignored. Parents with a tight budget may find that a Texet will comfortably fulfill a child's wish to own a home computer.

The Spectrum — beaten by the Oric?

The Colour Genie — distributed in this country by Lowe Electronics — is not to be considered as a serious contender to the Oric. Which brings me to the three machines that are probably the Oric's major rivals in the home computer stakes. Don't forget, Oric say that they must sell 250,000 micros this year.

The Dragon 32 is again a strong contender but at £200 the Oric has it beaten on price and in a number of other areas. However the Dragon does have a reasonable keyboard and some promising expansion possibilities.

A number of new micros about to appear, like the Acorn Electron, might beat the Oric. It doesn't matter though because the Oric is here and now: a microcomputer in the house is worth two on a twenty-eight day delivery promise.

Keyboards

There are three different types of keyboard on home computers in this sub-£200 range; tactile or touch sensitive, moving contact and typewriter-like — also known as a real keyboard. The Oric keyboard is of the second type, but it is an exceptionally high class member of this group, in fact it is the best moving contact keyboard.

Oric's keyboard scores points on every count. Its movement is more positive and firmer than that of the Spectrum or Texet, which feel like the clammy hands of a dead man. This is thanks to the extra layer of plastic that has been added to the keytops. More points are scored because the Oric makes a clear bleep noise whenever it is pressed. Better still this bleep can be turned off should you get fed up with it.

My favourite feature of the Oric's keyboard is none of the above, it is the clear uncluttered layout that I like. There is no playing around shifting here and there to access keywords. And you don't have to squint your eyes to read the multi-coloured legends written on the keys.

Of course those micros with 'proper' keyboards — the Vic, the Genie, the Atom and the Dragon — are easier to touch type on, but the firmness of the Oric's keyboard plus the bigger sized space-bar put the Oric almost in that class. Another feature that helps with typing is the way that the keyboard slopes at an appropriate angle.

Sound

The Oric is justifiably famous for its sound creation. Of all the microcomputers in the under £200 range, the Oric is undoubtedly King, at least in this field. It is the first microcomputer to have a decent-sized speaker, together with some amplification included as standard. The sound commands are very sophisticated for a machine of this price, in fact they are almost as good as those of the BBC micro, which costs £300 for the cheapest model.

Texet Tx8000 — will it ever appear?

The Oric and its Rivals

More important than the flexibility of the sound-handling software is the sheer volume of the Oric. It is loud enough to wake the zombies of death. No other computer, at the time of going to press, has anything approaching this feature.

The ZX81 is silent, to get it to make any noise you need to buy an add-on. Good as these add-on noise units might be, they do add an extra amount to the cost of a system.

It is also possible to get sound out of a ZX81 through the cassette port, but it isn't very good. Another option is to place a transistor radio near the computer. This picks up a signal generated by the electrical currents swinging about inside the micro. To program the ZX81 to make a sound by these means is not easy, in fact it will almost certainly require the programmer to delve into machine-code programming.

Machines with a slightly better way of creating sound include the Jupiter Ace, the ZX Spectrum, the Acorn Atom and the Texet. These machines all have internal speakers to generate sound and noises, but rather than having proper loudspeakers they use piezo crystals, which vibrate when an electric current is applied. They never produce a very loud noise, in fact if you live in a noisy environment you might not be able to hear much.

The BBC Micro — much more expensive.

Piezo crystal speakers can produce variable frequencies by having different electrical signals applied to them, so they are capable of playing music. Just how this is done depends to a large part on the specific software the machine in question operates.

The Spectrum has a very useful BEEP command to control its speaker. It can be used to specify the note played and its length, it doesn't allow volume control but it does have the advantage of letting you play notes between those of the musical scale. Being able to use those extra notes is an advantage — you can even create Arab or Indian-sounding music.

Playing music on the Jupiter Ace is as easy as the Spectrum. In fact, the Forth language is eminently suitable for musical programming. I look forward to having Forth running on my Oric — the musical possibilities are endless.

One reason the Ace is good is that it is fast, but it is more thanks to the way that simple commands — called words in Forth — can be combined to make more complex ones. All you have to do is associate passages of music with words and you can quickly compose an entire symphony.

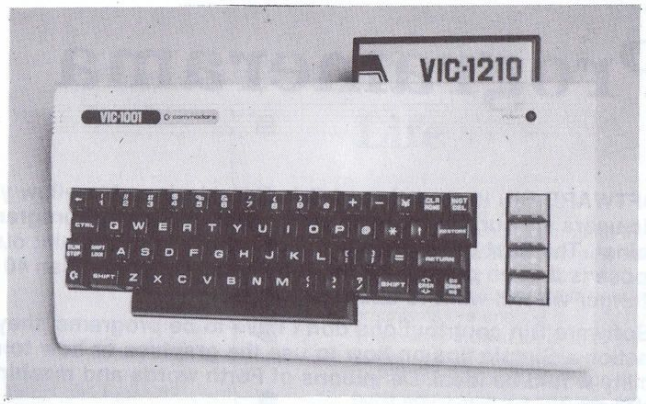
The Lynx — proper keyboard, good graphics, but slow.

Forth was used to write the excellent sound commands on the Oric. They were then compiled into machine code and included in the Oric's Basic ROM. Those of you who have received your versions of the cassette-based Forth should also have a music program written in Forth which plays a tune with three different voices. Playing around with Forth on the Oric demonstrates the speed and the other advantages of the language when it comes to musical programming. It is of course equally useful for simple sound synthesis too.

The Texet has rather poor control of its music feature, and as far as Basic is concerned so does the Acorn Atom. However, the Atom has the advantage that the machine has a built in Assembler. The manual is also very good at explaining how to create music using that Assembler. It is all done by PEEKing to a certain address, which is in fact an output port. This port, like any other memory address, consists of eight bits, one of which is connected to a small speaker.

If a one bit is sent to the speaker not much happens, if a one bit followed by a zero bit followed by another one bit and so on is sent to the speaker then it will vibrate, and consequently generate sound. This can be done from Basic but only notes of a certain frequency range can be achieved by this method due to the slow speed of any Basic interpreter. Other, higher frequencies can be generated on the Atom using machine-code. It is even possible to generate white noise by sending random numbers to the speaker.

The Dragon 32 does not have an integral speaker, instead it outputs sound through the television speaker. This technique has both advantages and disadvantages over the integral speaker approach to sound generation. Advantages include the ability to control the level of the sound directly by the television volume control and the fact that the user's perception of the computer is that its soul is somehow on the screen, so it seems only logical that its voice should come from next to the screen rather than from the lifeless main unit.



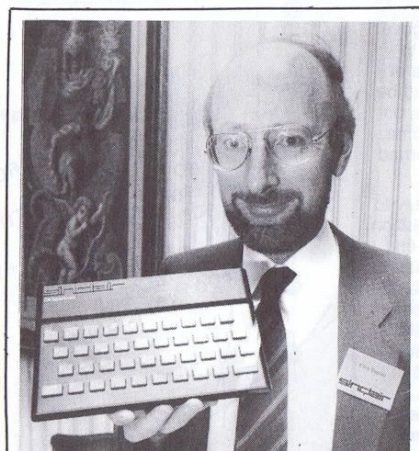
The Commodore Vic20 — can now be bought for under £100.

Another good feature of the Dragon's sound capability is the way that the signal from the cassette player can be switched to the television sound output. This can also be done on the Atari 400. It is a feature that is remarkably useful for teaching software, especially the teaching of foreign languages. But for all the flexibility of such a system there is one very major flaw.

A television sound signal has to be tuned in the same way as a video signal — it simply sits in a nearby frequency range. Normally this is fine, but we all know how difficult it can be to tune in a picture, let alone the sound as well. My experience of the Dragon is that you can have good sound or a good picture, not both. This kind of problem can never happen with the Oric.

The sound commands available from Dragon Basic are very good, especially the PLAY command. This allows you to specify music in a string form. There is only a single sound channel available on the Dragon, while the Colour Genie has three and the Commodore Vic-20 and the Atari 400 have four. The controlling software varies, but never does it get to be as good as that on the Oric.

Nowhere else in the entire microcomputer market is there a micro with a ZAP, SHOOT or EXPLODE, let alone a PING.



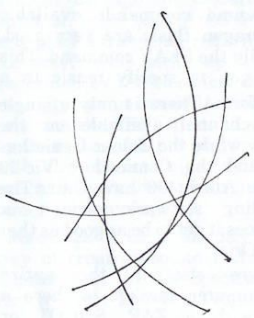
Why is this man smiling? If you think you know why send us a caption for this silly picture. We'll print the funniest.

Programerama

SOFTWARE BIN is a section of Oric User designed to allow you to swap ideas with other Oric users via our pages. Each month we will file your programs in the Oric User office filing cabinet. The best ones though will get chucked in the bin; our Software Bin. The limit we impose is that no program in the Bin should be longer than 40 program lines. If your program is longer we will want to use it elsewhere.

Software Bin contributions don't have to be programs, they can be routines or even functions. Simple tips on how to use the graphics or how to make the most of the sound facility would be ideal. Definitions of Forth words and machine code routines are just as welcome.

Send your contributions to Tarquin Binary, Software Bin Editor, Oric User



KEVIN SMITH of Hackney writes that the character set of the Oric contains certain surprises. Most of these characters can be examined by running a program that prints all the characters one at a time onto the screen. A typical program to do this looks like this;

```
100  FOR I=1 TO 128
200  PRINT I, CHR$ (I)
300  WAIT 200
400  NEXT I
```

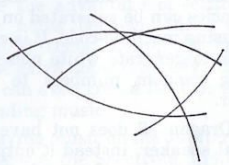
Running this program makes it obvious that some of the characters don't actually print anything on the screen. Some of these can be very useful. The control characters start with **CHR$(1)** which is **CTRL-A** and must go through to **CHR$(26– CTRL-Z.** These can be used to read the key-board or to cause the relevant control code to operate.

Of these control characters the most useful to know when writing software, especially games software are:

← (back cursor)	8
→ (right cursor)	9
↓ (down cursor)	10
↑ (up cursor)	11
RETURN	13

WHEN I WAS young, teachers gave school children Spirograph sets to play with. They consisted of sets of wheels with teeth on, which could be used to draw interesting patterns on paper. We didn't know it at the time, but these patterns were all governed by simple rules of mathematics.

Spirograph-like patterns are easy to draw using the Oric in high-resolution mode. Line 200 is the business-end of this program which can be used to draw a number of delightful patterns. The ratio 2/3 can be altered to other simple ratios to get different patterns. Various other parameters — that is numbers that are fed into functions — can be changed to get other effects, and the sines and cosines can be changed to achieve other effects.



```
100 REM SPIROGRAPHIC PATTERNS
200 HIRES
300 FOR I=0 TO 6*PI STEP PI/10
0
400 CURSET 120+100*COS(I*2/3)+
COS(I),100+100*COS(I*2/3)*SIN(
I),1
500 NEXT I
600 WAIT 5000
```


```
10 LORES 1
20 DIM A(17,15)
30 DIM B(17,15)
40 MUSIC 1,4,5,0
50 FOR I = 2 TO 16
60 FOR J= 2 TO 14
70 IF RND(1) <.4 THEN A(I,J)=1
:PLAY 1,0,1,15:PLOT I,J,"#"
80 B(I,J)=A(I,J)
90 NEXT J,I
100 FOR I=2 TO 16
110 FOR J=2 TO 14
120 Z=0
130 Z=Z+A(I-1,J-1)+A(I-1,J)+A(
I-1,J+1)+A(I,J-1)+A(I,J+1)
140 Z=Z+A(I+1,J-1)+A(I+1,J)+A(
I+1,J+1)
145 IF Z=3 THEN B(I,J)=1:GOTO
160
150 IF Z=2 AND A(I,J)=1 THEN B
(I,J)=1 : GOTO 160
155 B(I,J)=0
160 NEXT J,I
170 FOR I=2 TO 16
180 FOR J=2 TO 14
190 A(I,J)=B(I,J)
200 IF A(I,J)=1 THEN PLOT I,J,
"#":PLAY 1,0,1,15 ELSE PLOT I,
J," "
210 NEXT J,I
220 GOTO 100
230 REM
240 REM     # replaces pound s
ymbol
250 REM
```

Life

LIFE IS ONE of the oldest microcomputer games around. It is more of a simulation than a game really, because it uses the micro and its graphics to demonstrate how living organisms live and die in a colony. The colony lives in an area of the screen, 14 by 12 character squares big. This could easily be made bigger, but it would slow the program down too much.

Initially the cells are plotted to the screen at random. This is done in the lines 50 to 90. Line 70 sets the probability of a cell existing in any one square at 0.4. This can be changed to any other number.

Lines 100 to 160 calculate what the next generation of cells will look like by the following rules: Any cell that is surrounded by 2 or 3 other cells will stay alive, any empty location that is surrounded by 3 cells will grow a new cell, any other cells die.

The last loop, lines 170 to 210 plot out the current state of the colony. Eventually certain patterns will emerge, or the colony might simply die away.

Remember to replace the hash symbol in the program listing with the £ sign.

This version of life is very simple, if you think you can do better, either by speeding the program up, or making it look better, then we welcome your programs. How about writing a colour version, or even better life in three dimensions?

Nightmare Train!

THERE IS A MARX brothers film set in the Old West in which Harpo, Chico, Groucho et al are on a train being chased by people on another train. They can't stop to pick up fuel so they start chopping up the train.

This game is the exact opposite. You are in control of a train which keeps getting longer and longer. It is nuclear powered and generates waste. Every so often a bit of toxic waste drops off.

You can steer the train with the I,J,K and M keys. You must avoid hitting any bits of waste, crashing into the sides of the screen or turning back onto yourself. When the train gets too long waste-laden wagons drop off and the engine races ahead on its own, so you have to avoid them too.

Your objective is to survive as long as possible.

The problem with this kind of game is to make it fast enough to be exciting. The solution is to avoid plotting to the screen too often, which takes a relatively long time in Basic. So here we do not plot the whole train each time it moves. We plot the head, and keep a record of each place we have plotted. Later on we can come along and unplot the tail, resetting it to the background colour so the train appears to move.

Important variables

W is a two dimensional array which is used to keep the record of every point plotted on the screen to make up the train. A two dimensional array is just a table kept in computer memory, the equivalent of

x1 x2 x3 x4 x5 ... xn
y1 y2 y3 y4 y5 ... yn

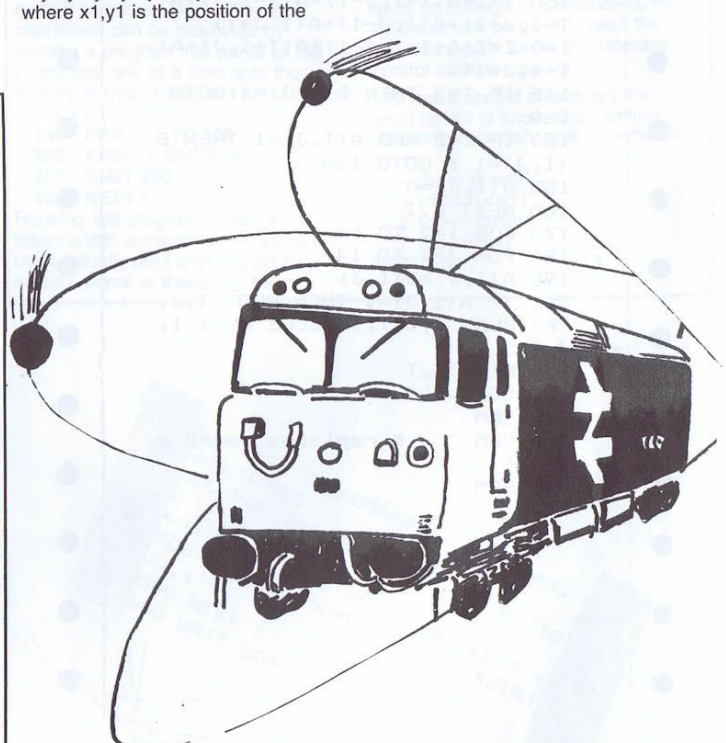
where x1,y1 is the position of the first point plotted and so on.

HP and TP are the head and tail pointers respectively. The head pointer points to the position in the table where you are going to store the x and y co-ordinates of the point you are about to plot. The tail pointer points to the place in the table where you will find the x and y co-ordinates of the point you are about to unplot, the tail of the train on the screen in other words. Unplotting resets the tail to the background colour.

All that is necessary to make the train lengthen or drop bits off is to make the pointers move around at different rates. Don't worry if you don't understand this bit of the program — it is quite intricate and it is never as easy to understand other people's programs as ones you have written yourself.

Program lines

10	Remarks, set up constants used throughout program, dimension array
100	Set up new game — clear W array
110	set initial values for one game
130	clear screen, draw borders
160	make start noise
200	main loop — check for crash
300	plot head of train
320	make chuff noise
400	save head position in array
500	unplot tail of train
600	advance array pointers so train will move
700	read keyboard to get changes in direction
800	work out new head position for train
820	add to score, display score
830	end of main loop — do it again
900	crash routine — make noise and display message
950	go and set up new game



Nuclear race to doom...


```
10 REM NUCLEAR WASTE
20 REM DOOM TRAIN
30 REM BY IAN WILLIAMS
40 REM COPR. CCC LTD.
50 REM 1983.
60 S=100'MAX LENGTH OF T
RAIN
70 PRINT CHR$(6)+CHR$(17
)'TOGGLE OFF KEY CLICK A
ND CURSOR
80 DIM W(S,2)'ARRAY FOR
SCREEN POSITIONS
90 REM DISPLAY INSTRUCTI
ONS
92 CLS
94 PLOT 10,8,"NUCLEAR DO
OM TRAIN"
96 PLOT 10,16,"USE I,J,K
,M KEYS"
98 WAIT 300
100 REM ---SET UP
102 REM ---NEW GAME
110 FOR I=1 TO 2:FOR J=0
 TO S:W(J,I)=0:NEXT J,I:
REM CLEAR ARRAY
120 X=19:Y=2:C=3'TRAIN P
OSITION
122 DX=0:DY=1'TRAINS DIR
ECTION
124 T=0'GAME SCORE
126 TO=0:HP=0'ARRAY POIN
TERS
130 REM SET UP SCREEN
132 LORES 0
140 FOR I=0 TO 38:PLOT I
,1,"+":PLOT I,26,"+":NEX
T
150 FOR I=1 TO 26:PLOT 0
,I,"+":PLOT 38,I,"+":NEX
T
160 SHOOT
170 PLOT 0,0,CHR$(8)+"HI
GH SCORE="+STR$(OT)
180 PLOT 24,0,"SCORE="
190 REM
200 REM -----MAIN
202 REM -----LOOP
210 REM DOES ONE TRAIN M
OVEMENT
220 IF SCRN(X,Y)>32 THEN
 900'CHECK FOR CRASH
300 PLOT X,Y,"O"'PLOT HE
AD
320 PLAY 0,1,1,101-(HP-T
P)'MAKE CHUFF
400 REM SAVE HEAD POSITI
ON IN W ARRAY
410 W(HP,1)=X:W(HP,2)=Y
500 I=W(TP,1):J=W(TP,2)'
UNPLOT TAIL
520 PLOT I,J,CHR$(16)
600 REM ADVANCE ARRAY PO
INTERS
610 HP=HP+2:IF HP>S THEN
 HP=0
620 TP=TP+1.8:IF TP>S TH
EN TP=0
700 REM GET KEY
710 C$=KEY$
720 IF C$="J" THEN DX=-1
:DY=0
730 IF C$="K" THEN DX=+1
:DY=0
740 IF C$="I" THEN DY=-1
:DX=0
750 IF C$="M" THEN DY=+1
:DX=0
800 REM FIND NEW HEAD PO
SITION
810 X=X+DX:Y=Y+DY
820 T=T+1'UPDATE SCORE A
ND
826 PLOT 30,0,STR$(T)'DI
SPLAY IT
830 GOTO 220'GO DO NEXT
TRAIN MOVEMENT
840 REM -----END OF
842 REM -----MAIN LOOP
850 REM
900 REM CRASH ROUTINE
910 PLOT 8,12,"CATASTROP
HE..."
920 EXPLODE:WAIT 200
930 IF T<OT THEN GOTO 95
0 ELSE OT=T
940 FOR K=1 TO 8
942 PLOT 0,0,CHR$(K)+"HI
GH SCORE="+STR$(OT)
944 PING:WAIT 35:NEXT K
950 GOTO 100'GO DO NEW G
AME
980 REM ---END OF
990 REM ---GAME LOOP
```

Stopwatch

HAVE YOU SEEN those quiz shows on television where the audience bite their nails as the seconds tick away on a huge clock? This program can be used to the same effect. When run the program starts by asking you how long? This can be any number — a single unit is about 2½ seconds. The clock then shows with the seconds ticking away. A ping tells you that the time is up, and this audio signal is confirmed by the screen returning to text mode and a message being printed out.

The real part of the program starts at line 1500, where the centre of the clock is defined. A circle is then drawn to define the edge of the clock face. Line 2000 is the start of the central timing loop, which counts up to pi times H, the time unit, in steps of pi divided by 10. Lines 2100 and 2200 calculate the points to plot the clock hand to. This is done in line 2300, lines 2700 and 2800 provide the tick sound and 3000 ends the loop.

```
100 CLS
200 INPUT "HOW LONG";H
300 HIRES
400 CURSET 120,100,0
500 CIRCLE 75,1
600 FOR I=1 TO PI * H STEP PI/10
700 CURSET 120,100,0
800 X=75*COS(I/2*PI)
900 Y=75*SIN(I/2*PI)
1000 DRAW X,Y,1
1100 CURSET 120,100,0
1200 DRAW X,Y,0
1300 MUSIC 1,5,1,0
1400 PLAY 1,0,1,10
1500 NEXT I
1600 TEXT
1700 PRINT "TIMES UP !!!"
1800 PING
```

Rootfinder

ALTHOUGH THE ORIC has a square-root function — SQR(x) — of its own, this program is of interest. It shows how roots to equations can be found by using a process called iteration. This works by starting off with an initial guess at the value and then making an improved guess and then another and another until the two last guesses are the same, which means the root has been found.

In the case of a square-root there is always a root of a positive number somewhere between zero and the number itself. The formula can be written:

$$c = SQR(n)$$

or more simply:

$$c^2 = n$$

which can be rearranged to give

$$c^2 - n = 0$$

This equation could be almost any equation, and in fact this method will find roots of any equation provided that there is at least one root between the first two guesses. In the case of the square-root the initial guesses are given in line 400 as zero and the number itself. The method finds the average of the two numbers, that is the midpoint between them. It then looks to see if the root lies between this midpoint and the first of the two previous guesses. If it does then this midpoint becomes the new lower guess, otherwise it becomes the new higher guess.

To find the root of any other equation lines 700, 800 and 900 need changing.

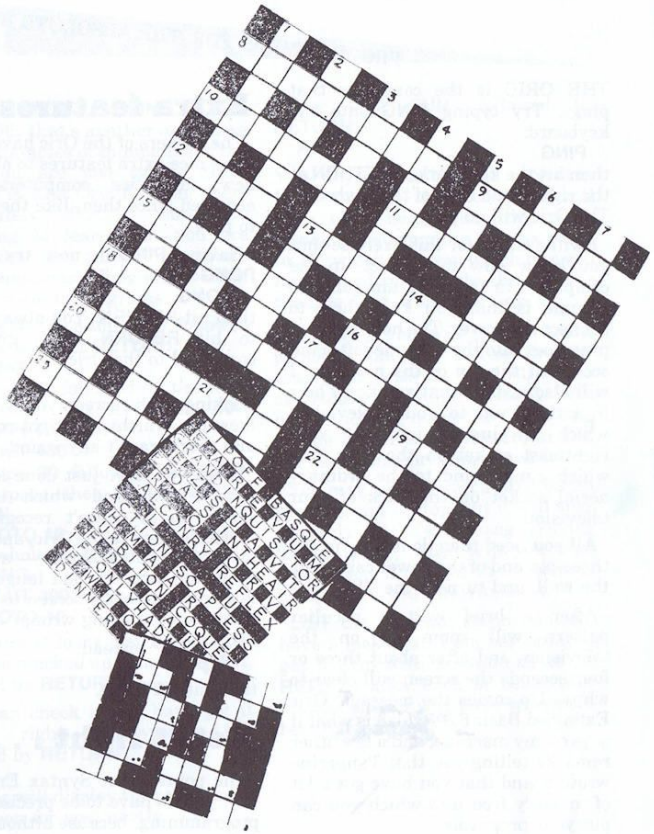
```
100 REM SQUARE ROOTS BY  CHOPPING IN
 HALF
200 INPUT "NUMBER ";N
210  INPUT "NO OF ITERATIONS";I
220 A=0:B=N
230 FOR J=1 TO I
240 C=(A+B)/2
250 AA=A*A-N
260 BB=B*B-N
270 CC=C*C-N
280 IF SGN(AA)=SGN(CC) THEN A=C : GO
TO 300
290 B=C
300 PRINT C
310 IF CC=0 THEN STOP
320 NEXT J
```


Wordfinder

THIS PROGRAM helps you to find a word when you know some of the letters, but not all of them. Simply type in the letters that you know, and leave blanks for the letters that you don't. The computer will then reply with a list of possibilities.

It is particularly useful when you are doing a crossword. Line 40 breaks up the word into single characters. If they are known then line 50 will send the character to line 80 where it is printed. If the character was a space, then in line 60 it is changed for a character chosen at random from a list. The list in this program contains all the alphabet once, together with some more of the commonly used letters — you may be able to improve on my list.

Short words are printed very quickly, so it might be a good idea to include a WAIT statement somewhere between lines 100 and 110. Another idea is to make the WAIT time dependent on the length of the word in question.



```
100 CLS
200 INPUT WORD$
300 FOR C=1 TO LEN(WORD$)
400 A$=MID$(WORD$,C,1)
450 IF B$ <> " " THEN GOTO 600
500 IF B$=MID$("ABCDEFGHIJKLMN
OPQURSTVWXYZEEAAIOUTHSS",RND(1
)*38+1,1)
600 PRINT B$;
700 NEXT C
800 PRINT
900 GOTO 300
```

PROGRAMMING FOR THE ABSOLUTE BEGINNER

THE ORIC is the computer that pings. Try typing **PING** onto the keyboard:

PING

then hit the key marked **RETURN** at the right-hand edge of the keyboard. The Oric will go ping.

I will assume for this exercise that you know how to connect up the computer. In fact all I am going to assume is that you know how to connect the power. The hollow round plug goes into the extreme left-hand socket at the top of the machine. I will also assume that you know how to connect up to your Television, which is plugged into the very rightmost socket on the Oric, and which runs round to the ordinary aerial socket on the back of your television.

All you need then do is plug in the three-pin end of the power cable into the wall, and turn on the TV.

After a brief wait a peculiar pattern will come up on the television, and after about three or four seconds the screen will clear to white. Up comes the message 'Oric Extended Basic 6.0', which is what it says on my machine, and a few other remarks telling you that Tangerine wrote it and that you have got a lot of memory free into which you can put your programs.

The final line of this display is 'Ready', and that tells you the system is ready for you to start entering Basic commands. That is what you did when you entered **PING**, for **PING**, followed of course by **RETURN**, is an Oric Basic command.

The Oric is an ideal machine to learn programming on. It is very easy and straightforward, and yet really very powerful. Programming is a very open-ended activity. There is no end to the challenges which you can find with your Oric.

Basic itself is used on the largest professional minicomputers from Digital Equipment, Wang, and Hewlett-Packard, to name but a few, but it was designed to be easy to learn, at Dartmouth College in America in the 1960s.

Extra features

The makers of the Oric have added some nice extra features to allow for new capacities computers have acquired since then, like the ability to ping.

Having **PING**ed, now try typing **PONG:**

PONG

then hit **RETURN.** You always have to hit **RETURN** when giving a command to the Oric.

Looking at the screen you will see a message, which says 'Syntax Error', and there wasn't any sound.

What you have just done is given Oric a command which isn't in Basic, so it doesn't recognise it. **PONG** is news to the Oric, and so the message 'Syntax Error' — a rather academic term for bad language — is displayed on the screen to let you know you are going wrong.

Try a **ZAP** instead:

ZAP

then hit **RETURN.**

Get it right . . .

The point about Syntax Errors is that you do have to be precise when programming, because although the Oric represents the state of the art in computers in the affordable price range, computers really are still not that bright. So they do take the instructions they are given very literally, and if they can't understand them they throw them out.

So any examples that you try and type in will only work if you type them correctly.

But the most important thing is that there is absolutely no way you can hurt your Oric by hitting the keys. So, why not just start hitting the keys to prove it. Just hit all the keys.

The keys are like the keys of a typewriter in the way they are laid out, and they even make a clicking noise. There are some special keys, for instance at the right-hand end the **RETURN** key, and four arrow keys either side of the space bar. These are there to let you move the black flashing dot around the screen — that dot incidentally is called the Cursor. An Escape key labelled **ESC** and a Control key labelled **CTRL** are at the left-hand edge of the keyboard; we will be using one of these quite soon.

So if you keep thumping madly away at the keyboard for a bit, and really do thump it, press absolutely every key in all possible combinations, probably by now something absolutely disastrous has happened. The machine will have made a few screaming noises and the screen gone blank or flashing or something. But you have done your worst, and it is worth knowing that whatever you have done now you can recover from. You can always get back to that 'Ready' prompt we saw at the beginning.

One way of doing it is to simply pull out the power plug, at the left-hand side of the keyboard at the top, and then push it back in again. This isn't at all dangerous, because the Oric runs off a mere six volts, which the power supply unit plugged into the wall generates. So even if you tear the casing off and start licking the machine there really is little you can do to hurt yourself.

So you can't hurt the machine, and the machine can't hurt you.

A slightly more classy way of getting the 'Ready' sign back in the case of any disasters you might run into in the future is the Reset button, located underneath the machine. If you turn it over you will see, to the right of the large round loudspeaker opening, a square hole just underneath one of the rubber feet of the Oric. If you poke a pencil into that hole, up against the lever in there, the screen will clear of the garbage you have put up on it and display the Ready sign.

This is actually the best way of getting back to something sensible you can understand when learning to program, because you don't lose any actual program you might have stored in memory.

Repeat, repeat, repeat

Another thing you might have noticed whilst playing around with the keyboard is that if you hold any Oric key down for very long it starts automatically repeating. This is true of all the keys, including the arrow 'cursor control' keys. So you can zoom the black flashing cursor dot around the screen and start writing anywhere on the screen. It doesn't really make much difference to the Oric. So zoom around a bit, and type **SHOOT**, yet another Oric noise.

ZAP, SHOOT, PING, and **EXPLODE,** that's another one, these are all examples of commands to the Oric. You are just typing them straight in and Oric is doing them immediately. So it won't be surprising to learn that you are using what are called 'immediate mode commands'. They also work if you use them in programs.

So type **10,** the number **10,** followed by **PING,** then hit **RETURN:**

10 PING

Then type **20,** followed by the word **WAIT,** then a space, then **400,** followed by **RETURN:**

20 WAIT 400

Then type **30** followed by **GOTO,** then a space, then **10,** followed by **RETURN**

30 GOTO 10

So the whole lot looks like this

10 PING
20 WAIT 400
30 GOTO 10

Make sure it looks just like that. If you have mucked up a line, retype it, followed by **RETURN**

You can check if you have got it exactly right by typing **LIST** followed by **RETURN**

LIST

and Oric will display the program on the screen for you, neatly.

Now type **RUN**, followed by **RETURN:**

RUN

You might possibly get a Syntax Error message, in which case it will be a more helpful one this time, saying something like 'Syntax Error in 30', which tells you which line you have typed in wrongly. Just retype the line as it is written here.

Otherwise you will get a gentle, persistent pinging. You set the program going with the word **RUN**. You can stop it by typing **CTRL C**. This means holding down the **CTRL** key, on the left of the keyboard, and at the same time pressing down **C**.

You may have to wait a little bit for the program to stop, but it will stop, probably with a message saying 'Break in 20', which tells you what instruction it was in the middle of doing when you interrupted it.

LIST the program again and let's have a look at it. I am beginning to get fed up with mentioning to hit **RETURN** after each of these commands, so you will have to remember that yourself.

Simplicity and power

The way the Oric understands a program is to look for commands that it recognises. It does this starting with the command with the lowest number, in this case **10 PING**. After finding **PING** it makes the ping noise. The command with the next highest line number is **20**, which is **WAIT 400**.

WAIT is another useful Oric Basic command, which tells the Oric to wait. If you tell it to **WAIT 100** it will wait a second, if you tell it to **WAIT 1** it will wait one hundredth of a second, so here it is obviously waiting four seconds. During the wait it does nothing except keep track of the time. It then goes on and looks for the next instruction. By the way, instruction is just another word for command. And it finds line **30**, which is the Basic instruction **GOTO** followed by **10**, and that just tells it to go and look for **10**. So it does. It finds line **10** and pings. Then it goes and looks for the next highest number, line **20**, waits 4 seconds, finds line **30**, and goes off and pings again.

It's in a loop. It would have gone forever, if you hadn't pressed **CTRL C** to stop it. Loops are the most common thing you will do in programming and instantly make clear the power of the computer: however simple the task, the ability to do it endlessly.

ZAP, SHOOT, PING and so on, as well as **WAIT**, are all special Oric Basic commands. Let's try a Basic command you will also find on lots of other machines.

The first one given in the not-very-good Oric programming manual is **PRINT "HELLO"**. That is in Chapter Three.

PRINT "HELLO"

It must have the quotes around the **HELLO**, and it must be the double quote rather than the single quote mark. That's obtained by holding down the **SHIFT** key, like on a typewriter. Both quotes are next to the **RETURN** key on the right of the keyboard. And of course hit **RETURN** after typing **PRINT "HELLO"**. The machine will say hello, on the screen.

Understanding loops

The first program given in the average dull programming manual does something like print hello a thousand times. If you crave recognition from your machine I expect you can figure out how to do it using the **PRINT** command and the **GOTO** command and Basic line numbers. If you want to get rid of the previous pinging program type in **NEW**, followed by **RETURN**. **NEW** clears out old programs.

We are going to be slightly more adventurous. After typing **NEW** type:

10 PRINT "STOP MAKING THAT DREADFUL NOISE"

followed by **RETURN**.

You will notice here that the program line went beyond the width of the screen, which does not matter. It was just continued on the next line. There is a limit to how long program lines can be, which you will sometimes come up against when printing long messages, and that is 78 characters long. Oric will warn you automatically by making a ping all by itself if you go beyond this number.

Next type:

20 ZAP

RETURN, then:

30 PRINT "SHAN'T"

RETURN

Note how the **SHAN'T** is enclosed in double quotes, got by using the shift key, and the single quote is in the normal place in the word. If you put single quotes around a message you want to print you will get a Syntax Error. Oric doesn't recognise it as a quote sign, only as an apostrophe, in other words. Computers can be pernickety.

Type:

40 GOTO 20

This time we are not going right back to the beginning of our program so the loop isn't going to include all the instructions of the program.

Type **LIST** to check you have got it right, and just retype, for the time being, any mistaken lines.

Now type **RUN**.

Write your own games

A recent survey revealed that 43% of the homes with microcomputers in them were using them to play arcade-style games. So, as there are not that many games around for the Oric yet, you might as well start picking up the programming knowledge to write your own.

If you have had enough, stop it with **CTRL C**. It should obey you. Otherwise there is always the Reset switch and the power plug.

LIST it again and have a look at it.

10 PRINT "STOP MAKING THAT DREADFUL NOISE"
20 ZAP
30 PRINT "SHAN'T"
40 GOTO 20

Line 10 is carried out once by the Oric. Lines 20 and 30 are in the loop caused by line 40, so **ZAP** and the printing of **SHAN'T** are carried out endlessly.

This is quite a good program, as it illustrates two important things about programming computers. It is really about controlled repetition a lot of the time, and it really has nothing to do with maths, which some people still seem to think lies at the heart of programming. Of all the computers in existence, most of them are probably home computers playing games. And as you move into business, there the machines tend to be handling words, like we are doing here. Maths comes a very poor third in terms of practical day-to-day use of computers, even in business.

Summing it up

If you are no good at maths you will be pleased to know that the Oric can be used for doing simple calculations in a pretty straightforward way. You don't even have to write a program for doing it. If you just type:

PRINT 4+8+7

then hit **RETURN** it will give you the answer. The plus key is at the right-hand edge of the keyboard, at the very top, and is a shifted equals sign.

And you can type:

PRINT 8−9

then **RETURN**, and it will tell you it is **−1**. The minus sign is next to the plus sign.

Or you can type:

PRINT 55 / 7

The '/' sign, pronounced 'over', is located next to the right hand shift key. Hitting **RETURN** brings up the answer, which in this case is a decimal fraction. The Oric is very accurate. It gives you nine digits precision.

There is no multiply sign on an ordinary typewriter style keyboard like the Oric has, so the ★, asterisk, symbol, which is over the eight — a shifted eight — is used instead. So

PRINT 7★9

RETURN gives the correct answer.

If you make a minor mistake hitting the keys you can use the delete key, **DEL**, just above the **RETURN** key, to backtrack over the incorrect letter, and just retype. Oric is definitely superior to a typewriter when it comes to making corrections.

PRINT 6−7★8

gives the right answer following the normal rules of mathematics about which operation you do first. Putting brackets round numbers works in the normal way if you want to change mathematical precedence, which if you do suggests you know what it means.

Let's get back to sound with a new program. Type **NEW**. What we really need now is a way of getting information from the keyboard while the program is running, so that you can tell the Oric what you want it to do without having to stop and type in immediate mode commands.

This can be done with the **KEY$** command. Type:

10 K$ =KEY$
20 IF K$="S" THEN SHOOT
90 GOTO 10

The dollar sign is over the four key, a shifted four, and of course everything has to be typed precisely. Let's run it first.

RUN

Nothing happens until you hit a few keys. Run your fingers around the keyboard depressing keys and you will get the usual click until you hit 'S', when it will make a shoot noise. If you hold the S down it of course autorepeats, so you get lots of shoot noises.

If anything really weird happens you have probably held down the **CTRL** key and pressed some other key. Some of these **CTRL** key combinations are used for things like clearing the screen and turning the blipping noise on and off.

You can stop the program with **CTRL C**. If, by hitting a weird key combination you have caused something to happen you don't want, turn the machine over and poke a pencil through to hit the Reset button. you will come back up with a clear screen and 'Ready' message. If you type **LIST** you will see your program still there; that is why Reset is better than pulling the power plug out. RAM, remember, looses what is in it when you turn off the power. This is why the Oric has a cassette deck connector, for storing programs in, but we won't be dealing with that now.

So let's look at the program. Line 10 is unfamiliar looking. Looking at the end part first, **KEY$** is an example of what is called a Basic function. When Oric carries out the instruction at line 10 the **KEY$** part makes it go and see if the keyboard has been pressed since it last looked at the keyboard. If it has it will return that single key to the program currently running. A function returns something, a value, to a Basic program.

Pigeonhole storage

The first part of the line 10 has got two new things in it. **K$** is where you are going to put this value. **K$** is a place in the Oric's 48K of memory specially set aside by you, by the mere fact of giving it the name **K$** in a program. It is a special pigeon hole for you to store things in. You can create a large number of these pigeon holes, officially known as variables because you can keep changing what they hold, but we only need the one for this program. We have given it the name **K$**, but other names like **L$**, **M$**, or even **SID$** would work equally well. Any name you use must have a dollar on the end of it — more about that next month.

The second new thing is the equals sign. This is a special use of equals, which is one of the few slightly confusing things about Basic. What it really does is makes something equal. So in this case, **K$**, your special location, is made equal to the letter sent back by the **KEY$** function from the keyboard. That sounds a bit complicated, but it is straightforward when you think of what we are trying to do, which is: get the key pressed on the keyboard and remember it somewhere in memory.

The next line the computer is going to carry out is line 20, and it has another interesting new instruction in it. It's an **IF . . . THEN** statement. This is another of the Basic statements which are common to all versions of Basic not just the Oric, so if you come across another machine you will find it there too. Here it means if what you have put in your **K$** memory location is in fact an S, then do something, in this case **SHOOT**.

The **IF** something something **THEN** something statement is officially called a conditional statement, because what happens is conditional upon something else that you check, in this case what is in **K$**.

Line 90 is the good old loop statement, the **GOTO** statement. The only thing different here is that the Basic statement line number is 90. There is no need to go up in steps of 10. All that the Oric Basic interpreter is concerned about is whether it can sort the line numbers into order so it can know which of your instructions to carry out first. As long as no two statements have the same line number, in which case you just wipe out the first one when you type in the second, and as long as no line number is greater than 63999, in which case you get a Syntax Error, you are all right.

Let's not New this program, in other words wipe it — let's add to it. Type:

30 IF K$="Z" THEN ZAP

RETURN. Now **LIST** that, and notice that we can add a line anywhere and Oric will sort the lines into the right order, so that when we **RUN** it it will carry out the instructions in the right order.

Now **RUN** it. Nothing much happens — it is just the same as before — until you hit the keys. If you hit the Z now it makes a zap noise; if you hit the S it makes a shoot noise. We can **CTRL C** it to stop, and add more lines in a similar way, to do any of the things that we have learnt, print, shoot, explode, whatever.

To make this process more simple — **LIST** the thing again — the Oric has a helpful little feature, which is the **CTRL A** key. That is if you hold down the **CTRL** key and press **A** it will copy things for you. This is used with the arrow keys, the cursor control keys, to make writing long programs a good deal easier, both to copy and to correct mistakes.

Copying made easy

So using the arrow keys move up the left-hand side of the screen until you get to line 30. That is place the flashing cursor to the left of the '3' of 30, then hit the **CTRL A** key — holding down the **CTRL** key hit **A** and when it goes over the 3, take your finger off the **CTRL** key, and hit the 4 Key. Now go back to the **CTRL** Key, hold it down, hit **A**, and hit **A** again, still holding down the **CTRL** key, several more times, until the flashing cursor is over the 'Z'.

Now take your finger off the **CTRL** key and hit P, then go back to the **CTRL** key and hit **A** again. Continue hitting **A**, with the **CTRL** key held down, all the way along the line until the cursor is over the Z of **ZAP**. At that point take your finger off the **CTRL** key and just type in **PING**. Then hit **RETURN** and **LIST**.

There should, in addition to line 30, be a new line 40, which reads:

40 IF K$="P" THEN PING

If instead you have got something else you can do the whole thing again, this time to correct line 40. That is move up with the arrow keys then use the **CTRL A** key to copy in any characters appearing on the screen which are to your liking. If they are not to your liking and you want to change them just take your fingers off the **CTRL A** key and type in whatever you want the new thing to be. And that includes things like the space bar.

The delete key, on the right-hand side of the keyboard, also works. When you have made your modification, list your program again. Continue doing that till you get it right. It soon becomes natural. Then **RUN** it.

It will come as no surprise what it is going to do, when you hit P. **CTRL C** it to stop.

Save it

There is one further refinement we might want to add. At line 50 type:

50 PRINT K$

K$ remember, is the special location in the Oric's memory where the value of the last key pressed on the Oric keyboard is saved. So if we print it we should get the letter you have just pressed appearing on the screen. So with the new line 50 in, run the program.

What happens is that what was on the screen rolls off the top. This is because each time you go round your loop if you haven't hit a key when you print **K$** it prints nothing, but then moves the cursor down to the beginning of the next line. When the cursor reaches the bottom of the screen it stays there, but moves the rest of the screen up. This is called scrolling, and is done deliberately by the print statement to create space for itself on the screen in case it has any actual message to print. If you now start hitting the keys you will notice the letters appearing on the screen. And if you hit one of the letters you have set up to make a noise, it will both make a noise and appear on the screen, because **K$** has now got something in it.

CTRL C it to stop. There is something else that can be done to line 50. Type **LIST**. You can in fact list just one particular line. So having typed **LIST**, **RETURN**, now type:

LIST 50

and that will display just line 50, which is quite convenient if you want to change it, which we do.

Move, with the up-arrow key, up to the beginning of line 50 then hit **CTRL A**, all the way along the line until we get to the dollar sign. Then hit **CTRL A** once more, so that the cursor moves just beyond the dollar sign. If you now type perhaps a not very obvious thing, the semi-colon, which is to the right of the **L** key and is typed unshifted, and then hit **RETURN**, you have actually changed what the print statement will do. It should look like this when you list it

50 PRINT K$;

If you print something without the semi-colon it puts it out on the screen on a line of its own, as we have seen. If you put that semi-colon there it stops that happening, so that the next time you print something out it will go right next to the last thing you printed. You can see this if we just **RUN** the program, and start typing.

What we now have is program which allows you to write on the screen any message you like. The Oric is, I suppose, now slightly booby-trapped, as every so often, just when you have forgotten, it issues a strange explosion-type noise.

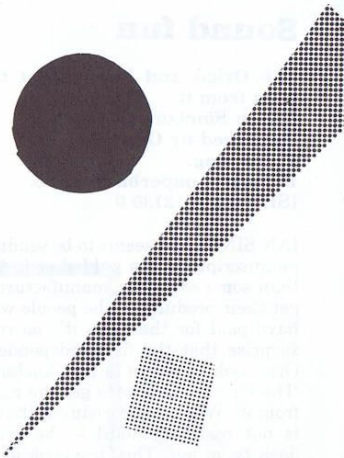
The cursor keys — the up, down, right, left arrow keys — still work whilst in this loop under the control of your program. So you can write messages absolutely anywhere. And most of the special **CTRL** keys work, in particular **CTRL L**, which is got by holding down **CTRL** and pressing **L**. **CTRL L** clears the screen allowing you to start afresh with a new message. So you now have a slightly booby-trapped message board.

What you have learned

1. Layout of the keyboard
2. How to give immediate commands to Oric
3. Getting out of trouble with Reset
4. Making Oric obey programs — **RUN**, **LIST**, and **CTRL C** to stop a program
5. Editing programs with **CTRL A** and the cursor control keys
6. Storing values in a variable.

The Basic commands used

1. **ZAP**, **PING**, **SHOOT** and **EXPLODE**. The predefined sound commands
2. **GOTO**, the loop control command
3. **PRINT** with quotes. The statement for typing messages on the screen
4. **PRINT** with numbers. Oric as a calculator
5. The **KEY$** function. It gives your program the value of the last key pressed
6. **IF . . .THEN**. For making things happen when conditions are right
7. **WAIT**. Gets the Oric to do nothing till the amount of time you specify has passed.



NEXT MONTH

Next month: Graphics, and another look at sound.

We will begin to delve further into Oric's very fine sound abilities, and will start drawing with the machine.

If you are using a colour television you might like to get into the feel of things by trying out the INK and PAPER commands. For instance, try typing **PAPER 2**. These commands work on black and white sets with slightly different results. You can use numbers between 0 and 7 with both commands, otherwise you get an error message you won't have seen before.

Don't forget, if you get into any trouble, Reset underneath the machine will get you back to that familiar and reassuring 'Ready' message — and without spoiling your program.

Oric Colour Numbers

0 Black
1 Red
2 Green
3 Yellow
4 Blue
5 Magenta
6 Cyan
7 White

Sound fun

The Oric-l and how to get the most from it
by Ian Sinclair
Published by Granada Publishing.
135 pages paperback, £5.95.
ISBN 0 246 12130 0

IAN SINCLAIR seems to be sending manuscripts to his publisher faster than some computer manufacturers get their products to the people who have paid for them. So it's no real surprise that the first independent Oric book to arrive is Ian Sinclair's 'The Oric-l and how to get the most from it'. What is surprising is that it is not really dreadful — he must dash them out. The Oric book joins his two Spectrum books and Dragon, Lynx and Commodore 64 offerings on the Granada list.

For £5.95 you get a 136 page paperback, quite nicely produced and with a few helpful illustrations scattered throughout. It includes a good index.

The book is not an attempt to cover everything about the Oric, or even Oric Basic. It is not an 'Oric revealed', delving into obscure machine code Pokes, or a one-to-one replacement of the rather poor official Oric programming manual. Ian Sinclair's book is a guide to programming for the newcomer to computers, and it covers a reasonably broad selection of Oric Basic instructions.

Here is what is in each chapter. I give his heading and sometimes my comment.

1. Connections. A straight-forward setting up guide.
2. Numbers and strings. Covers Print and Input statements and variables.
3. Disorderly behaviour. Looping, with Goto, If. . .Then, For. . .Next statements. A bit sudden for the beginner I suspect.
4. String handling.
5. Arrays and files. Also subroutines.
6. Rolling your own. Covers program design, somewhat boringly.
7. Graphics 1. Constructs a space invader rather laboriously with teletext graphics.
8. Graphics 2. Better chapter, on plotting.
9. New characters and sounds. Covers user defined characters, and sound. The best chapter in the book.
10. Odds and ends.

There are then four appendixes and an index.

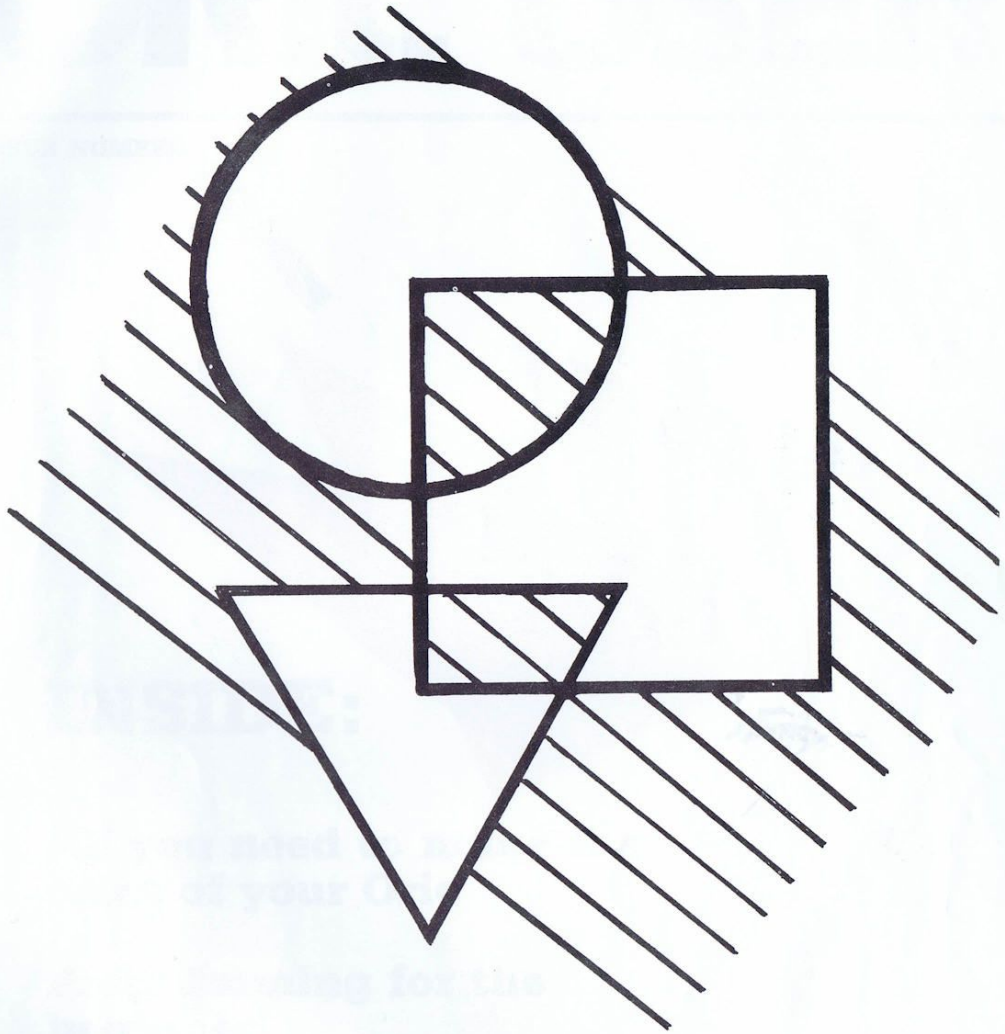
The tone of the book varies. Chapter one is ultra-simple, and includes a fully illustrated guide to tuning in your TV set. When the programming starts, in chapter two, the reader's intelligence is assumed to go up slightly, and by chapter three the information is coming thick and fast. The book does not include any long programs. The short programming examples illustrate Sinclair's points well, but don't do anything very interesting.

On the positive side the nine pages on sound in chapter nine are really good. Sinclair is good on sound in his other books as well. I suspect this is the thing he is really interested in. With a brand new Oric and a book to write on a tight deadline, sound is the thing he really seems to have got into, and it shows. These pages are interesting, and the examples are a good deal better. If you don't think you need a book to supplement the Oric manual do have a look at this section, as it does make things much clearer. The best part to read in the bookshop, in other words.

'The Oric-l and how to get the most from it' is overall a curiously insubstantial book, considering how wordy and information packed it seems to be if you open it and look at a page at random. I think it shows the weakness of the formula publishing approach. It doesn't take advantage of the things the Oric is particularly good at, or the things that are particularly easy to program on the machine. It ends up making rather heavy work of the Oric, a nice easy machine.

But Sinclair has the first book in the field, and it is better than, or at least different from, the Oric manual. At least he doesn't use Basic instructions in his programming examples and then forget to explain them, like the Oric manual does.

Ian Williams



- £10 FOR A YEAR'S SUBSCRIPTION
- OVERSEAS RATES ON REQUEST

WRITE TO:

ORIC USER

20 WYNFORD HOUSE
WYNFORD ROAD
LONDON N1 9QY

ORIC USER

ISSUE NUMBER 1

INSIDE:

All you need to make the most of your Oric

Programming for the beginner

News and reviews

Oric v. the rest

Six pages of software

NEW!