SOFT OPTIONS OPEN UP

Little software was available for the Jupiter Ace for several months after last October's launch, but now there are a number of programs to choose from.

A salesman at Lasky's in London suggested the original software gap was a result of people buying the Jupiter because they want to learn to program in Forth. They want to be able to use a bigger machine which uses Forth for business purposes, create their own graphics or just to take on the challenge of mastering another language.

Micro Marketing, a London firm which distributes the Jupiter Ace, agreed that people with a knowledge of Forth often buy the Jupiter Ace to experiment with programming, because it is the only native Forth micro.

The company also felt that the need for software was great, and now offers ten games marketed under its own name. These consist of four at 3K for £4.50, one combined 3K program for £6, and five 19K tapes for £6. They include Acevaders, Swampmonsters — which has little boats running around dangerous swamps — and Jupiter Says, the Jupiter Ace version of Simon Says.

Now that Micro Marketing has started the ball rolling, others have followed suit. Titan Programs of Manchester is currently developing a Space Invaders type game for the Jupiter Ace, and Computer Services, Leicestershire, is working on some educational programs.

Technical hitches and a limited user base have led to a delay in Computer Service's production, but the first education programs — mathematics, French and Spanish — will be ready by August. And a Christian education series, giving Biblical knowledge and quizzes for use on Jupiter Ace and other machines, is being considered.

The firm had hoped to keep prices to £10, but escalating developing costs may hike these up a bit.

After the educational programs are successfully launched, Computer Services will turn their chips to Home Process Control, to enable the Jupiter Ace to be used for operating burglar alarms, lights, cookers, etc. This should not prove difficult as Forth was originally designed to operate equipment, especially radio telescopes.

The Jupiter Ace Users Club has started its own software company, Remsoft, which currently produces ten games and utility tapes as well as an adapter kit to modify the Ace so that it can take Sinclair ZX81 add-ons. Remsoft also sells both 16K and 32K RAM packs to extend the limited memory of the unexpanded Ace.

The games include Night Rider — a challenge familiar to certain drivers, Appleater — another version of the Apple Panic game long available on the Apple II machine, Frogger — a black and white imitation of the arcade game, and Maze, a simple maze game.

Prices on the games range from £3.50 up to £6.50, although you will need an expanded machine to play some of the more complicated games. The 16K RAM pack sells for £30, and the 32K pack for £50.

Computer Services and Remsoft will have some strong competition from Jupiter Cantab itself, which has finally found the time to develop its own software.

The first batch of nine tapes — three 19K games, five tapes of 3K games, two games per tape, and a monitor program — will be out in a couple of weeks. They — like the rest of the software, will be priced from £6-£10.

The range of games includes Greedy Gobbler — a Pac Man variation that keeps track of high scores, plays a fast game and uses only 3K, Blowing Up The World — a rather odd game whose main objective is to bomb as much of the world as possible, and an Ace version of Space Invaders.

After that, a new program should be available almost every week. More games are planned, as well as home budgeting and club records. Even a chess program is on the drawing board.

Planned educational programs include graphics, maths for statistics and geometric calculations, morse code training and a music program. Eventually a full tax and accounting package will also be available, and software accessories, such as an assembler and a tool kit to help design and SAVE graphics, are being developed.

Stephen Vickers, a director of Jupiter Cantab, explained that the firm always intended to supply software with the Ace. 'It's been a matter of sorting out time to do it. The first batch is being issued later than we would have wanted it.'

But with Jupiter Cantab's 30-40 new programs, supplemented by other new offerings from Micro Marketing, Titan, Computer Services and REM software, Jupiter Ace users should be well served for software.
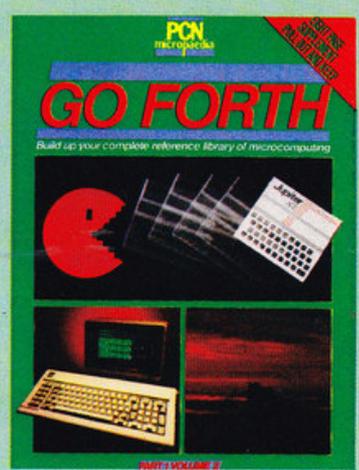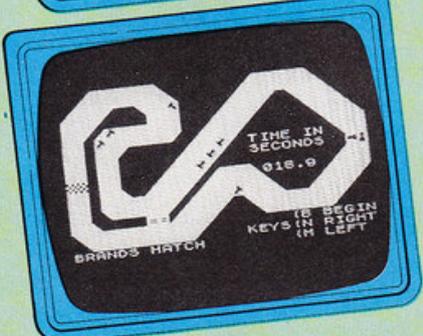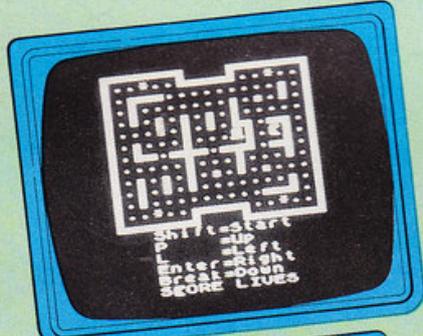
Pictured on the left is what you'll see when you load five of Jupiter Cantab's new games for the Ace.

The Pac-Man look-alike is called Greedy Gobbler, while the world map is part of a program Jupiter Cantab calls Blowing up the World. Your objective in the second game is to 'see how quickly you can get rid of the lot.'

In the race track game, called Brands Hatch, you are a racing car driver trying to complete as many laps of the course as quickly as possible. In the last, most familiar game, you are defending your home base in a game called Missile Men.

EVERYTHING YOU NEED TO KNOW ABOUT FORTH . . . (BUT WERE ASHAMED TO ASK)

Basic programmers coming to Forth for the first time often find the environment confusingly different. Certain simple facts about Forth systems are often not explicitly explained in the manuals and books, because the authors consider them too obvious.

And most users will probably be ashamed to ask, revealing their 'ignorance'. Here then is a selection of dumb questions — with the answers your mother never told you.

Q Where do programs live in a Forth system? Can I list them the way I do in Basic?

A Programs (usually called definitions) live most of the time on tape or disk. Forth is in this sense more like a compiled language than interpreted Basic. You write your 'source' (*ie*, not yet compiled) programs using a built-in editor and store them as numbered 'blocks' or 'screens' on tape or disk.

In most systems you start to write a source program by typing EDITOR and then whatever the number of the block you want is — '26 EDIT', for example. Often low numbered blocks will already be full of source code supplied with the system. You must find an empty one for your programs, so try 100 to start with. Looking at a screen with the Editor is the equivalent of listing a Basic program.

When you want to use a program, you compile it by LOADING the relevant blocks of source code into RAM. You do this by typing '26 LOAD', or whatever. The compiled program now lives in RAM in the 'dictionary', which is a linked list of all the words Forth understands.

Your program is LOADED at the top of the list, and down below it are the words that make up Forth itself. This basic dictionary is LOADED into RAM when you start up your Forth system and your programs are added on top as you compile them.

Of course, when you switch off the power your compiled program is gone, and you must LOAD it again next session (though many systems will let you SAVE the whole dictionary as a new, bigger Forth system if you wish).

Once the program is compiled into the dictionary you cannot LIST it or change it at all, unless you own a Jupiter Ace. To change it you go back to the source code using the Editor and then LOAD again. You can LIST the names of the words in the dictionary though by typing VLIST.

Q But isn't it possible to type definitions straight in from the keyboard?

Arcade games must move fast, so the programs that make the games work are often written in Forth. Forth is different in that its programs live for most of the time on tape or disk, and in this sense it is more like a compiled language than interpreted Basic.

A It certainly is. But definitions typed this way go straight into the dictionary, so you can't edit them if they are wrong or LIST them (again, except on a Jupiter Ace). And of course since the source code doesn't get onto tape or disk your definition will be lost once you turn off the power or crash the system — you'll do quite often at first.

Incidentally, Forth doesn't care at all where its input comes from — it treats a block which is being LOADED from tape or disk exactly as if you were typing it from the keyboard.

Q So does everything I type from the keyboard go into the dictionary?

A No. If you type a word, HELLO, for example, Forth will search the dictionary to see if the word is already there. If it is, then the word must *do* something, and Forth will proceed to do it. If it isn't then Forth does not understand it and will spit it back at you ungraciously thus: HELLO?

For Forth to put something into the dictionary, it must be preceded by one of Forth's compiling words. The most common of these are colon (:), CONSTANT and VARIABLE — later you may create your own compiling words.

If a compiling word is there, then Forth will try to add your input to the dictionary. But if you've done something illegal, eg a syntax error, it might fail and give you an error message.

If you type a number, Forth will still try to look it up in the dictionary. When it fails to find it (and before spitting it out) Forth checks to see if it has been given a number rather than a word.

Since it has, it puts it on the 'stack', which is where Forth keeps and manipulates numbers. This isn't the place to go into the stack and reverse polish and all that stuff — this is well explained elsewhere.

Unfortunately in most Forth systems you can't see the stack. When you type 123, Forth swallows it with a polite 'OK' and nothing more. It is very helpful when learning Forth to have a permanent display of the stack contents on the screen, and the lack of this facility is a major omission in most systems (designed by people who have forgotten what it was like to learn).

Some systems have a word like .S, which non-destructively prints the stack contents, but you have to type it out every time you change the stack.

Modify your system so the stack is automatically displayed before the 'OK' prompt returns, but to do this you will need to have either the source code of your system to edit, or better still a 'vectored' prompt routine which allows you to substitute new code without recompiling the system.

The stack can be permanently displayed on the bottom line of the screen by using Sharp's split-scrolling window facility on the MZ80B.

Q What is the stack for, apart from storing numbers?

A All arithmetic in Forth is done on the stack, and the answer is left on the stack. It will only appear on the screen if you request it by typing a dot (.) to print the top item. But more important the stack is the means by which Forth words pass values to each other. In Basic a value is communicated to a subroutine by putting it in a variable, eg LET A=123 : GOSUB 2000.

Though Forth has variables too it is more usual for a Forth word which produces a numeric result to merely leave it on the stack; the next word to be executed finds it there, uses it, and perhaps leaves a new number for the next word to find.

The reason for the 'backwrads' appearance of many Forth commands is just this. For instance, when you typed 26 LOAD what you really did was put the number 26 onto the stack; the word LOAD is defined so that it expects to find a number on the stack and uses it.

If you type merely LOAD then you will either get a 'stack empty' error or worse, if there was a number you had forgotten already on the stack, you will LOAD a block you didn't intend to.

Stack discipline is the hardest part about learning Forth. Most of the goofs you make to start with will be because a word leaves something on the stack which it wasn't supposed to. In addition the reason many people say that Forth programs are unreadable is because the source code gives very little clue to what is happening on the stack.

In fact, Forth programs can be made quite readable by intelligent choice of word names — in most languages you don't even have a choice — and by scrupulously commenting the effect of each word on the stack. When developing Forth programs you may find it more important to write down the stack effects than the actual definitions on paper.

Q How do you read a Forth program?

A Backwards, as usual. The Forth dictionary has a time dimension, in as much as words can only be defined in terms of words lower down, ie earlier, than themselves. New words are added to the top.

When a series of blocks is compiled into the dictionary the lower numbered are compiled first and the highest last, so that the last word defined sits on top of the dictionary. This imposes a 'bottom up' order on the way programs (a program is just a set of word definitions) are written.

The lowest level words are defined first, then higher level words are defined in terms of these, and so on until you end with the words that you will actually type to use the program.

But programs should be designed and read from the top down, ie from the highest level. This means that to understand a Forth program you should start to read at the last block ie the highest level definition, and work back.

Q Is Forth as addictive as people say?

A Yes and No.

You read a Forth program backwards, as words in the Forth dictionary can be defined only in terms of words lower down than themselves. New words are added to the top.

When a series of blocks is compiled into the dictionary the lower numbered are compiled first and the highest last, so the last word defined sits on top of the dictionary.

Laboratory Microsystem's implementation of Forth on the IBM PC is an excellent advertisement for the language.

Forth formats fingered

Forth can be installed on your micro; how and when it's done depends simply on how much money and/or time you have to deal with the installation.

Over the next three pages we present views and reviews of installations on three popular micros. We range from one of the most expensive machines — the IBM PC — through a middle-priced machine — the Apple II — to the lowest price micro on the market — the Sinclair ZX81.

The prices on these installations are reflected by the machines they're expected to run on. The least expensive is the installation on the ZX81, a mere £15 (still a bit dear on a micro that only costs £50), while the less expensive PC Forth and Apple Forths are about equally priced at around £100, with the IBM Forth slightly on the more expensive side.

And while you're getting your installation ready, it's a good idea to work with reverse polish notation for a while on a calculator. Old HP calculators operating on this system are available at cheap prices from secondhand shops if you don't want to spend the £100 or so to get a full running native Forth micro such as the Jupiter Ace.

Using Forth is simply a matter of changing the way you think about computing operations. As Harry Katzan says in *Invitation to Forth:* 'In a Forth expression such as 4 3 +, it is executed from left to right. When a data value is encountered it is placed in the stack by "pushing down" values that are below it. This is why a stack is commonly referred to as a "pushdown stack". When an operation is performed, it uses up the needed values from the top of the stack and "pushes" the result back on the stack.'
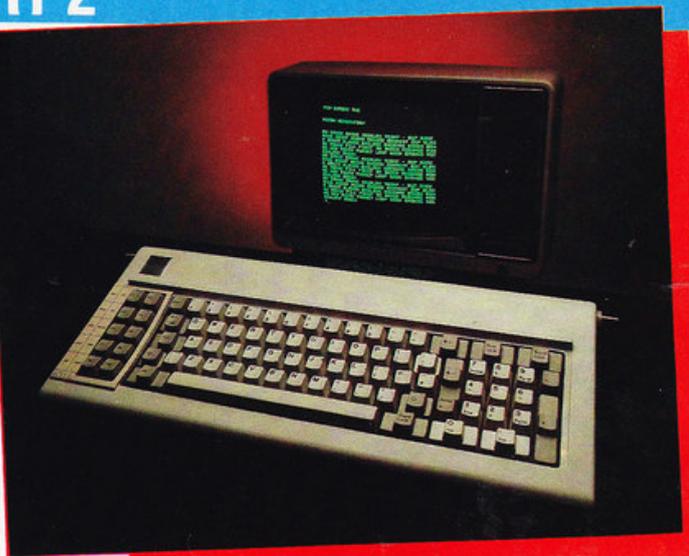
FORTH WITH OUT TEARS

Implementing a new language on your micro can be a traumatic experience, even if it is a language with Forth's attractions. But the Laboratory Microsystems PC/Forth user manual goes a long way towards getting Forth on your IBM PC without tears along the way.

The manual comes in a white A4 sized ring binder, which also contains two floppy disks holding the PC/Forth. The disks do not have write-protect flaps on them, and this is a pity because re-configuring PC/Forth is quite easy to do — as is ruining the master disks through accidental writes. There is also a small PC/Forth quick reference card. This contains the Forth words used frequently.

The package, reviewed included floating point and graphics facilities, which are available as optional extensions. The extensions come in separate transparent folders hooked in the front of the manual, and come with all the necessary documentation.

Laboratory Microsystems gives details of how you can copy the software from the master disks to a new systems disk, minimising the danger of accidentally overwriting one of the master disks.

Once a new systems disk has been configured, and the masters stored away in a safe place, the manual gives details of how the systems disk can be reconfigured (again) into one arge Forth.COM file. If you take this option you'll have a very powerful Forth system on one disk.

The main part of the manual is divided into sixteen sections. The main section is the users guide, while the others give a more detailed description of the system.

There are just 17 single sided pages in the user guide section, but it's surprising how much information is there. This section allows you to set up the system and get things going immediately, without having to wade through masses of paper to get to the information you need.

The user guide gives details of how PC/Forth has been configured to run on the IBM Personal Computer, along with a description on how to get Forth.COM RUNning.

GRAPHIC GAIN FOR PC

The software for Laboratory Microsystems' implementation of PC/Forth uses the PC-DOS operating system together with a high-resolution graphics terminal. Either PC-DOS or C/PM-86 can be used with this package, as PC/Forth has words that allow the system to communicate with either operating system. PC/Forth files can also be transfered from one system to another by using commands in the operating systems not from within PC/Forth itself.

Once LOADed, the Laboratory Microsystems logo appears on the screen with information as to what screen source file is attached with PC/Forth on boot-up. Different screen source files may be attached to PC/Forth on booting up the system. Entering A:FORTH B:DEMO.SCR from PC-DOS would attach the file DEMO.SCR on drive B:, subsequent LISTs, LOADs etc would be within the file DEMO.SCR.

It is a pity that PC/Forth uses the standard ROM video drivers used for text output. Any output to the screen is at the speed the video drivers can manage, and this isn't fast enough.

Laboratory Microsystems does give details at the back of the manual on how you can reconfigure the video drivers 'for special applications' but it might have been better if this had been handled by the software.

The PC/Forth system is very robust, and it took some devious manipulation of the stack to crash the system. Error messages are actually in English, and if you need a fuller explanation of what is going on a section in the manual gives the error message, together with a complete explanation.

As is the case with many Forth packages, an Editor is not present in the vocabulary, but the systems disk has three which can be permanently attached to PC/Forth. This idea of keeping the frills outside the system is a good one, and Laboratory Microsystems follows it up by giving details of how the extras — non standard commands — can also be removed from the main Forth.COM file.

Demonstration software is provided with the package, the games being of particularly high quality. Eratosthenes Sieve program, which originally appeared in the September 1981 issue of BYTE magazine, is an impressive demonstration. This program is executed in only 50 seconds, and shows the speed of Forth.

There are also some graphic demonstration programs, and these are also of high quality. In fact, if you opt for the graphic extension to PC/Forth, a very powerful graphics library can be built up.

References are made to the demonstration programs on the systems disk as the description continues.

The remainder of the users' section gives details of the PC/Forth compiler and the extra words that had been added to PC/Forth to allow it to communicate with the PC-DOS system.

Sections three and four give descriptions of Forth.SCR and Demo.SCR, which are two demonstration files on the PC/Forth systems disk. Forth.SCR contains various programming utilities for the Forth programmer, and some of these are of a very high standard. Kim Harris's Case and the Forth decompiler are particularly good.

The PC/Forth package contains three screen/line editors of one type or another. The main editor is a full-type screen which has been customised for the IBM PC. As is the case with the whole of the manual, no details are left out, but it isn't boring.

There is a full description of the screen editors' functions together with the name of the commands, which are logically sub-sectioned into cursor, line, word, string as well as some others.

For the Forth die-hards who think a screen editor is too easy, there is a version of FIG-Forth's original line editor by W Ragsdale. This and the other line editor are fully documented.

Forth for the Apple comes in varied versions, from graphics through to number-crunching for the US Navy.

APPLE ANIMATION — OR OTHER OPTIONS

The Apple II was one of the first affordable micros, and therefore has probably the widest range of Forth implementations of any personal computer.

GraForth, Microspeed Forth, MicroMotion Forth-79, Apple II Forth by Kuntze, TransForth II, and many others are available, and when you have as many Forths as this, the one you choose will depend on what you want to do with the language.

GraForth, as is suggested by the title, is well suited for graphics on the Apple. And because the graphics are RUNning in Forth, the increased speed of the language allows you smoother animation of images than is available with Basic programs.

The most exciting aspect of this animation is 3D, which is available using the Image editor. Up to 16 3D objects can be manipulated by the program at once, and they can be rotated, scaled, translated or positioned.

Insoft, which distributes GraForth, brags that its version can count to 30,000 in less than three seconds. While you might never want to count that quickly, it gives you a good idea of how fast the package will handle animation.

At only £60, GraForth looks to be an economic and useful implementation of forth for the graphically-minded.

A more expensive Forth package is MicroMotion's Forth-79, which sells for between £95 and £124, depending on whether you want floating point and high resolution Turtle graphics. Compared to the GraForth package and the PC-Forth at around the same price, this one just doesn't make the grade.

Part of the problem is documentation. This becomes immediately apparent when you try to LOAD what is essentially an 80-column package on a 40-column screen, and the manual doesn't tell you any reason why you shouldn't make this attempt.

You get as far as paragraph two only to be told to turn to Appendix A to start up the package. From that point on it goes from bad to worse. In the package's floating point arithmetic, for example, a simple question like the square root of 16 is answered with 3.999990. It seems the point floats just a little bit too much.

Microspeed is more expensive, and is a combined hardware/software package which transforms the Apple into a very fast number-cruncher. So fast in fact that it will outpace a PDP11/34, and fast enough for the US Navy to use the system to flight-test new aircraft.

The language is a heavily-modified and extended version of Forth, which is why we've included it here. Like all Forths, Microspeed is programmed in RPN.

Its main feature is the high-speed maths processing chip. This is on a plug-in card, and takes care of all the really complicated floating-point and trigonometrical calculations.

The 6502 chip which is the heart of the Apple is not particularly good at doing calculations on large numbers, so this scheme takes the weight off it and lets a real specialist do it. The 6502 is thus free to concern itself with flow control.

With Artic's implementation of Forth on the Sinclair ZX81, you get about the cheapest Forth machine available. Perhaps the only way of running anything close to Forth at a lower cost is the use of a reverse polish notation calculator.

The word is spreading...

Low-cost Forth implementations have been developed for many other micros than the IBM, Apple and ZX81.

Recently Artic computing, which produced the ZX81 implementation reviewed here, released an implementation of Forth for the Sinclair Spectrum and a toolkit for the BBC Forth implementation.

The Spectrum implementation is similar to the one developed for the ZX81, and does not have the disk handling facilities which might make it more powerful. If it had, it could be used in conjunction with the much-touted Sinclair microdrive. The package sells at about the same price as its ZX81 progenitor — £15.

Microprocessor Engineering offers Forth for the TRS-80 machine, the HP-85 micro and the Atari home computers. All these implementations sell for about £87.

There is also a version of Forth available for the Dragon 32 computer. Available from Oasis Software, it's called DragonForth and promises to be a FIG-Forth implementation for the Dragon that can also execute Basic statements. On cassette and selling for £18.95, it allows use of the Dragon sound and colour commands as well as floating point.

There are also several versions of Forth available for the BBC micro on both cassette and on EPROM.

FORTH CHANCE FOR ZX81 USERS

ZX Forth is a good implementation of FIG-Forth for the 16K ZX81. It comes with an Editor, which is on three screens next to the compiler, and should provide the FIG-Forth fan with a reliable version of one of the original Forths on the ZX81.

The Forth comes in a small black box which contains the tape, two instruction manuals and an installation note. LOADing time is approximately six minutes, and the tape is LOADed in the same way as any other tape for the ZX81.

Also supplied is a 'peel-off' keyboard overlay, which should be stuck to the ZX81's keyboard whenever ZX Forth is to be used.

The reasons for this are twofold: first, Forth words such as ! , @ and ' are printed on the overlay, and secondly, special graphics characters replace the normal ZX81 characters.

ZX Forth uses the ASCII character set, and if the user wishes to get the extra characters such as graphics and upper and lower case printed out then either a Q.S. Chr$ board or dk'Tronics Graphics ROM with extra RAM should be used.

You don't have to use other extra, but if ! was entered on the keyboard then some other character would be printed. However, there would be no adverse effect on the program.

The display on the ZX81 is slightly jittery when the ZX Forth is being used. Every time a key is pressed the screen 'dips' by a small amount, then returns to normal. This may sound bad, but the effect is not as annoying as the ZX81's display in FAST mode.

The Forth is very fast, of course — character output is in the region of four to five times that of Sinclair Basic.

The compiler itself is fast, and adds two new words — FAST and SLOW — to the vocabulary. A simple DO...LOOP of 32767 to 1 takes 12 seconds to complete in SLOW mode, and three seconds in FAST.

The addition of FAST and SLOW in the ZX Forth makes good sense, as compilers should to some extent be tailored to the machine they are to run on.

There is just over 7K free when the compiler is LOADed, and once the screens for the Editor are added this amount is reduced to 2K on a 16K machine. Therefore, once editing has been complete it is wise to forget the Editor altogether.

Unlike the Jupiter Ace, which is a dedicated Forth micro, ZX Forth uses the standard Forth words to LOAD and SAVE screens to and from tape by using the Editor supplied with the tape.

At first it seems a lot of trouble to load four screens of Editor before you can EDIT your own screens, but when you realise that memory is at a premium it makes sense to have the Editor separate from the compiler — this is standard Forth anyway.

No extra words have been added to the standard vocabulary to allow the use of the ZX printer, but it shouldn't be a problem to write the necessary routines to achieve this. Forth provides many facilities to improve access to the system in general, so filling the printer buffer with the necessary data and executing the necessary routines in ROM might be one way to achieve this.

It is also possible reset the boundaries of the dictionary and then exit to Basic where the modified Forth could be saved to tape.

Overall, the compiler is suitable for anyone who owns a ZX81 and wishes to learn Forth. After self-tuition you can easily progress to another Forth system with confidence, and without the fear of having to learn another new set of words.

THE STANDARD BEARERS

Many versions of Forth are currently available for micros, but they all derive from two dialects that were defined in the 1970s. These are FIG-Forth and Forth-79 (see part 1).

When the language first gained recognition the Forth Interest Group (FIG) was set up to promote it. This group developed what it called FIG-Forth as its standard version. The 'standard' itself is embodied in the FIG-Forth installation manual as written by William F Ragsdale.

The European Forth Users Group (EFUG) was founded in 1976. The Forth Standards Team was formed to set a standard for all Forths, and this task force came up the standard known as Forth-79, actually released in October 1980.

Programmers interested in implementing Forth from scratch or wishing to learn more about the two dialects should read the FIG-Forth and Forth-79 manuals — both available from FIG.

Each standard has a good following, and there is much debate between the devotees of FIG-Forth and Forth-79 as to which is worthy of being called *the* standard.

The dialects differ both in content and in the way they operate. The reader should be aware of these differences before obtaining a Forth compiler, as some of them — double number extensions, for example — are considered of such paramount importance by either camp that they wouldn't dream of resorting to the other standard.

ZX SPEX

Name ZX Forth
Applications Language implementation
System Sinclair ZX81
Price £15
Publishers Artic Computing
Formats Cassette
Outlets Artic Computing, 0482-75284.

STACK MANIPULATION
The word column gives the name of the word.
Action gives a description plus effect on stack ie (before --- after)

FIG-FORTH / FORTH-79

WORD	ACTION	WORD	ACTION
PICK	Copy n1-th number to top (n1---n2)		Not supported
ROLL	Rotate n-th number to top (n---(n))		Not supported
DEPTH	Count numbers on stack (---n)		Not supported
	Not supported	SP@	Leaves address of top of stack (——addr)

COMPARISON

WORD	ACTION	WORD	ACTION
0≥	True if top number is greater than zero (n---flag)		Not supported
NOT	Reverse value of flag (flag1---flag2)		Not supported
D<	True if d1 is less than d2 (d1 d2---flag)		Not supported
U<	Compare two top items as unsigned integers (un1 un2---flag)		Not supported

ARITHMETIC AND LOGIC

WORD	ACTION	WORD	ACTION
1-	Subtract 1 from top number (n---n-1)		Not supported
2-	Subtract 2 from top number (n---n-2)		Not supported
	Not supported	DABS	Absolute value of a double number (d---d)

MEMORY

WORD	ACTION	WORD	ACTION
	Not supported	ERASE	Fill n bytes in memory with zeroes. Start at address (addr n ——)
	Not supported		As ERASE but with blanks

TERMINAL I/O

WORD	ACTION	WORD	ACTION
U.	Print top number as unsigned integer with black (un---)		Not supported
	Not supported	.r	Print number right-justified in field (n fieldwidth ---)
	Not supported	D.R.	As .R but for double number

Forth makes great use of the stack, and there are a number of differences between the two standards. The table shows some of the differences then, but not the entire set of words required for both.

The tables are in no way complete, as there are different versions of both dialects. Anyone interested should consult the appropriate installation manual.

The reader shold not decide what compiler to invest in merely on the basis of these tables. Control structures, mass storage and definition words have been left out for the sake of brevity.

Such is the flexibility of Forth (both dialects) that even with a certain word that one dialect may have and yours hasn't, it is possible to give the definition of the missing word (if you know it) and to have it compiled as part of your Forth dictionary.

This can even apply to control structures that another dialect may have, although it is not recommended that you start messing about with them.

A LOOK AT THE WORD SET

Books are more important for Forth implementations than for almost any other language. They give you much needed background, and assist in-depth implementations.

Here we look at three books about Forth, each dealing with the language in increasing levels of complexity. The first, *The Complete Forth*, by Alan Winfield, is also the thinnest (and the least expensive) of the books.

But this is one of those cases where you can't tell a book by it's cover. In addition to the Forth-79 handy reference card at the back of the book, there is an index which lists the complete standard Forth-79 word set. There are also 127 pages of tightly packed and well-presented information about the language.

The book starts from scratch, telling you how to read the book and what both novice and computer expert alike will get from reading it. The author then starts on the fundamentals of Forth, bringing in the notion of reverse polish notation, the stack and then a series of pieces about using numbers in Forth and learning stack manipulation.

In the second chapter he jumps into an explanation of the dictionary, Forth error messages,
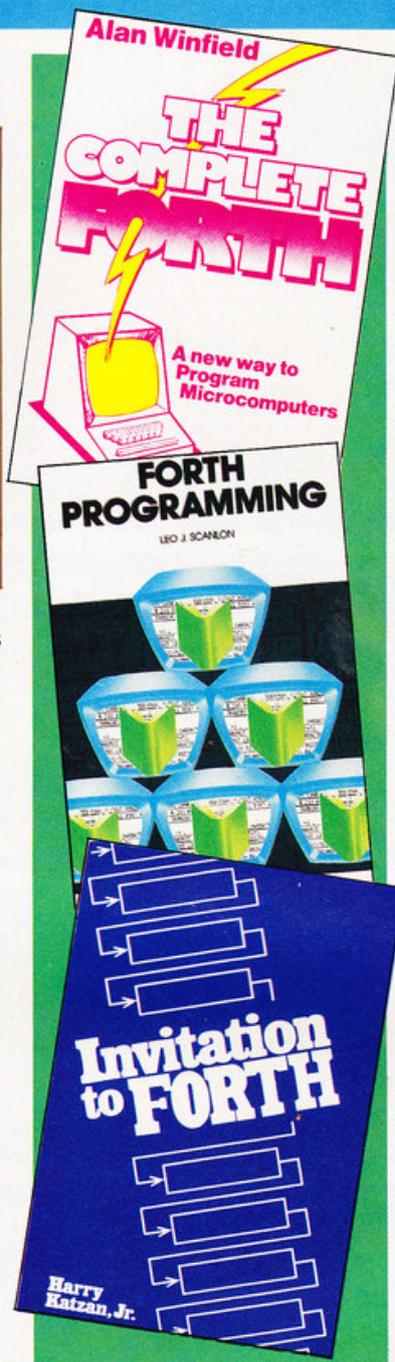
and how to use Forth variables. Mr Winfield continues in this vein throughout the book. The very fact that he is so thorough, however, tends to make the book a little grey, and might deter some novices who'll be attracted by the zappy illustration on the cover.

The book is published by Sigma Technical Press and sells for £6.95.

The other two books are printed in the US. Leo J Scanlon's *Forth Programming* is sightly more technical, although its extensive use of diagrams and tables makes it look less chock-a-block than the Winfield book. The structure is similar, although Mr Scanlon uses the greater length of his book (246 pages) to go into more detailed discussions of stack manipulation, do-loops, control structures and the differences between Forth-79 and FIG-Forth.

Forth Programming includes a useful appendix of double-number extension words to process larger values on 32-bit machines, and there is also the obligatory comparison between Forth-79 and FIG-Forth words, as well as a hex-decimal and ASCII conversion table.

An Invitation to Forth, by Harry Katzan Jr, is the most technical of the lot — and unfortunately the hardest to read. The dot-matrix printer used to typeset the book on an Apple II is an admirable attempt to bring micros to the publishing world, but it still doesn't make the dot-matrix output any easier to read.

The book is divided into ten chapters — starting oddly enough with chapter 0 — and explains most of the Forth concepts with clearly labelled flow-chart diagrams and control structure models.

Apart from the standard group of Forth chapters common to the other two books, there is also a chapter on double precision Forth and information management using Forth.

There is also a rather erudite chapter on computer philosophy, which details the differences between the Harvard and Princeton architectures. It doesn't necessarily bear any direct relation to Forth, but it makes interesting reading.

The three books about Forth we've selected for review may well whet your appetite for more information on this increasingly popular language.

And so on and so forth

Once you've read through the Forth Micropaedia and you've read through the books reviewed here, you may want to get more involved in Forth. If you do, there are plenty of groups you can turn to, and the numbers are growing all the time.

A good place to start is either your local Forth Interest Group (keep an eye on PCN's Clubnet for names and addresses in your area) or the Jupiter Ace Users Group.

The Jupiter Ace Users Group is a relatively new group that publishes its own quarterly newsletter, runs its own software house and has developed a number of discounts for club members wanting to buy Ace software and peripherals.

If you're running Forth on the Jupiter Ace, they are probably your best independent reference point for getting more out of your machine. They can be reached at the address given on page nine for Remsoft.

NEXT WEEK

Next week we begin our massive five part series on graphics for six of the country's most popular micros. We'll have hints and programming lists (in living colour) for the BBC Micro, the Atari 400, the Oric 1, the Sinclair Spectrum, the Commodore Vic-20 and the Dragon 32 machines.

Over the five weeks of this series you'll get forty pages of PCN's best for the best of Britain's low-priced micros. Colour will flow off our pages and onto your screens with the mere press of a few keys (well, actually quite a few).

We'll also be featuring some of the programs discussed in the Micropaedia in our weekly ProgramCards selection, so you can clip and keep them with your other PCN ProgramCards.

Stay tuned, it's going to be a colourful five weeks.

Contributors: David Janda, Marcia Macleod and Geof Wheelwright.