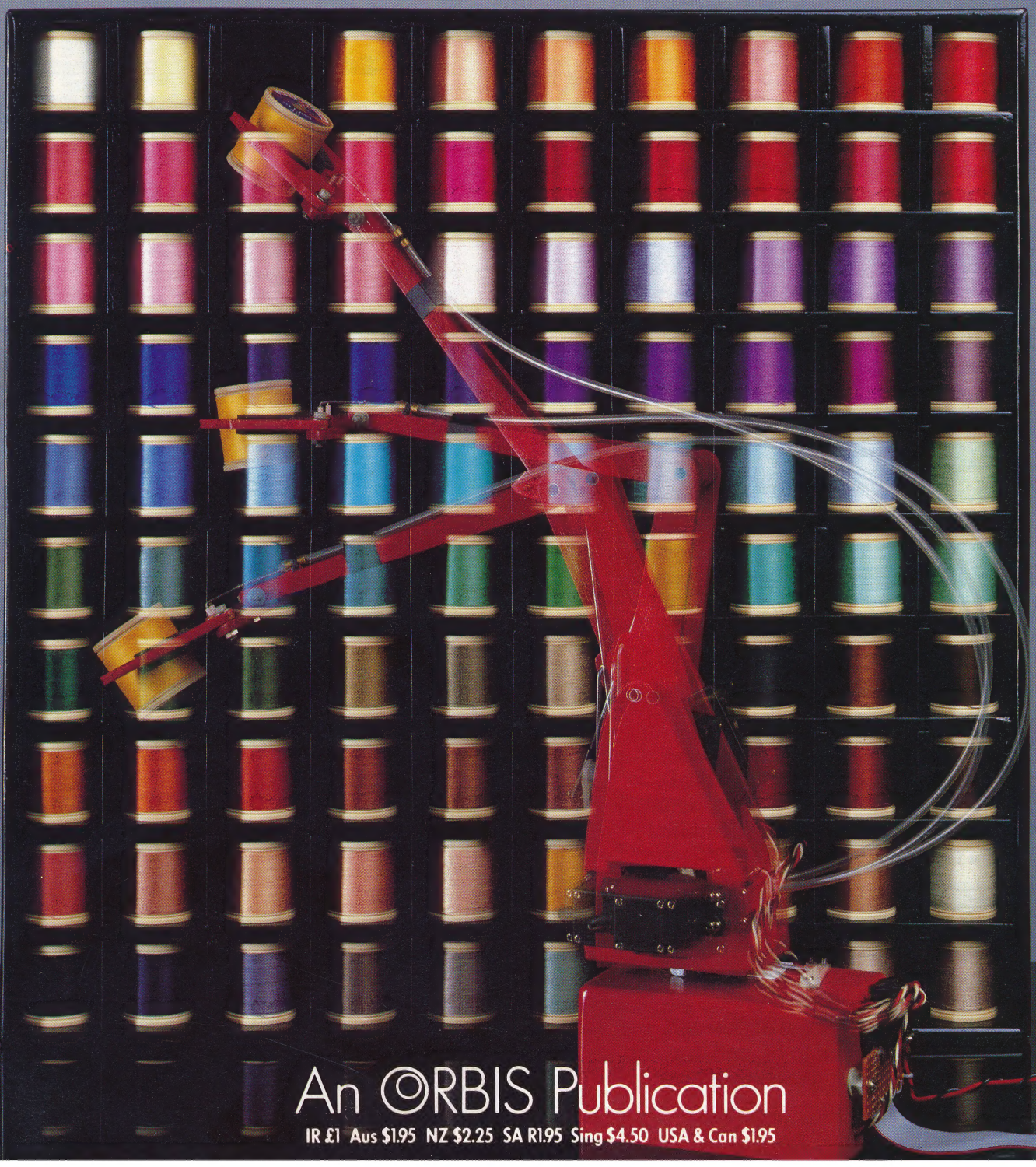
ISSN 0265-2919

90p 54

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO



An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

APPLICATION

FIELD TRIPS TO OTHER WORLDS Computer simulations and adventure games may seem mere 'play' when set against traditional educational techniques, but they provide invaluable learning experiences — 1061

HARDWARE

ACORN ADD-ON The Acorn Plus 3 provides the Electron with a disk drive and disk filing system — 1069

SOFTWARE

DOWN ON THE FARM We look at a suite of farm management programs for the Dragon micro — 1064

SPACE FORTRESS With its excellent 3-D graphics and fast joystick response, Zaxxon will particularly appeal to arcade enthusiasts — 1080

COMPUTER SCIENCE

1 + 2 = 20? We explore aspects of PASCAL'S mathematical abilities, and provide a program that generates successive terms in the Fibonacci sequence — 1074

JARGON

FROM MATRIX TO MIDI A weekly glossary of computing terms — 1068

PROGRAMMING PROJECTS

SUPPLY LINES Having taken on a crew for our 16th century New World trading voyage, we now develop the module that organises the data relating to the ship's provisions — 1066

MACHINE CODE

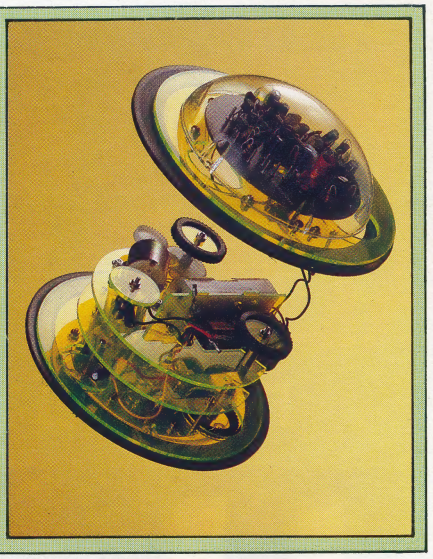
WE INTERRUPT THIS PROGRAM . . . The BBC Micro's use of controlling signals called 'interrupts' is the subject of this first article in a two-part series — 1077

WORKSHOP

BLUEPRINT In our Workshop series over the coming weeks, we will show you how to construct a robot arm. But before we get down to the nuts and bolts, we consider some of the options open to us, and draw up a sketch of the machine we will build — 1072

Next Week

- **One of the most important applications of the computer is the database. In our first article in a new series, we examine what databases are and how they work.**
- **The Acorn Music 500 produced by Hybrid Technology turns the BBC Micro into a complete synthesiser. We look at this new peripheral for the computer musician.**
- **Many people in the micro industry predict that home robots will shortly be a major growth area. We take a look at Movits — low priced robots for the home hobbyist.**



QUIZ

1) What is the value of simulations in education?
2) How can a computer improve the 'calving index'?
3) What is the purpose of the *MOUNT command?
4) What is generally considered to be the first microprocessor?

Answers To Last Week's Quiz

1) 14 address lines are needed by a 16 Kbyte ROM to enable the processor to address all 16 Kbytes. An eight-bit processor uses two bytes to call addresses above 255.

2) The AMX mouse software closely resembles that on the Apple Macintosh.

3) CASE allows a variable passed to it to select one of a series of options that are available within the statement.

4) The robot would no longer register any difference between the ambient light and the line it was following and would tend to stray off course.

Managing Editor Mike Wesley; **Editor** Stephen Cooke; **Art Editor** Claudia Zeff; **Production Editor** Bobby Pickering; **Technical Editor** Steve Colwill; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Staff Writer** Stephen Malone; **Sub Editor** Jonathan Kaye; **Contributors** Geoff Bains, Nick Walsh, Joe Pritchard, Steve Malone, Karl Dallas, Anthony Ginn, Steve Colwill; **Software Consultants** Pilot Software City; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Maurice Geller; **Production Manager** Peter Taylor-Medhurst; **Subscription Manager** Christine Allen; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1985; © Orbis Publishing Ltd 1985: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Heanor Gate Printing Ltd, Derby**

HOW TO OBTAIN ISSUES AND BINDERS FOR THE HOME COMPUTER ADVANCED COURSE – Issues can be obtained by placing an order with your newsagent or direct from our subscription department. If you have any difficulty obtaining any back issues from your newsagent, please write to us stating the issue(s) required and enclosing a cheque for the cover price of the issue(s). **AUSTRALIA** – please write to: Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G, Melbourne, Victoria 3001. **MALTA, NEW ZEALAND & SOUTH AFRICA** – Back numbers are available at cover price from your newsagent. In case of difficulty, write to the address given for binders. **UK/EIRE** – Price: 90p/IR£1.15. Subscription: 6 months: £26.00. 1 Year: £52.00. Binder: please send £3.95 per binder, or take advantage of our special offer in early issues. **EUROPE** – Price: 90p. Subscription: 6 months air: £44.72. Surface: £36.14. 1 year air: £89.44. Surface: £72.28. Binder: £5.00. Airmail: £8.25. **MALTA** – Obtain binders from your newsagent or Miller (Malta) Ltd, MA Vassalli Street, Valetta, Malta. Price: £3.95. **MIDDLE EAST** – Price: 90p. Subscription: 6 months air: £50.18. Surface: £36.14. 1 year air: £100.36. Surface: £72.28. Binder: £5.00. Airmail: £8.25. **AMERICAS/ASIA/AFRICA** – Price: US/CAN$1.95/90p. Subscription: 6 months air: £59.54. Surface: £36.14. 1 year air: £119.08. Surface: £72.28. Binder: £5.00. Airmail: £9.50. **SOUTH AFRICA** – Price: SA R2.45. Obtain binders from any branch of Central News Agency or Intermag, PO Box 57394, Springfield 2137. **SINGAPORE** – Price: Sing $4.50. Obtain binders from MPH Distributors, 601 Sims Drive, 03-07-21, Singapore 1438. **AUSTRALASIA/FAR EAST** – Price: 90p. Subscription: 6 months air: £64.22. Surface: £36.14. 1 year air £128.44. Surface: £72.28. Binder: £5.00. Airmail: £9.75. **AUSTRALIA** – Price: AusS2.15. Obtain binders from First Post Pty Ltd, 23 Chandos Street, St Leonards, NSW 2065. **NEW ZEALAND** – Price: NZS2.65. Obtain binders from your newsagent or Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington.

ADDRESS FOR BINDERS AND BACK ISSUES – Orbis Publishing Limited, Orbis House, Bedfordbury, London WC2 4BT. Telephone 01-379 5211. Cheques/postal orders should be made payable to Orbis Publishing Limited. Binder prices include postage and packing and prices are in stirling. Back issues are sold at the cover price, and we do not charge carriage in the UK.

NOTE – Binders and back issues are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK and Australian markets only. Binders and Issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

ADDRESS FOR SUBSCRIPTIONS – Orbis Publishing Limited, Hurst Farm, Baydon Road, Lambourn Woodlands, Newbury Berks, RG16 7TW. Telephone: 0488-72666. All cheques/postal orders should be made payable to Orbis Publishing Limited, Postage and packaging is included in subscription rates, and prices are given in sterling.

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH

FIELD TRIPS TO OTHER WORLDS



Good At Games
Educational simulations and adventure games make it possible for children to explore otherwise inaccessible, or even potentially dangerous, situations from within the classroom environment. Far from being a waste of time, experience suggests that such games can add considerably to the learning process

Just as schools are microcosms of the larger world of learning, so a microcomputer can serve in a school as an environment within itself. We examine the enormous learning potential that has developed with the advent of simulation and adventure games, which for the student open doors to otherwise inaccessible worlds.

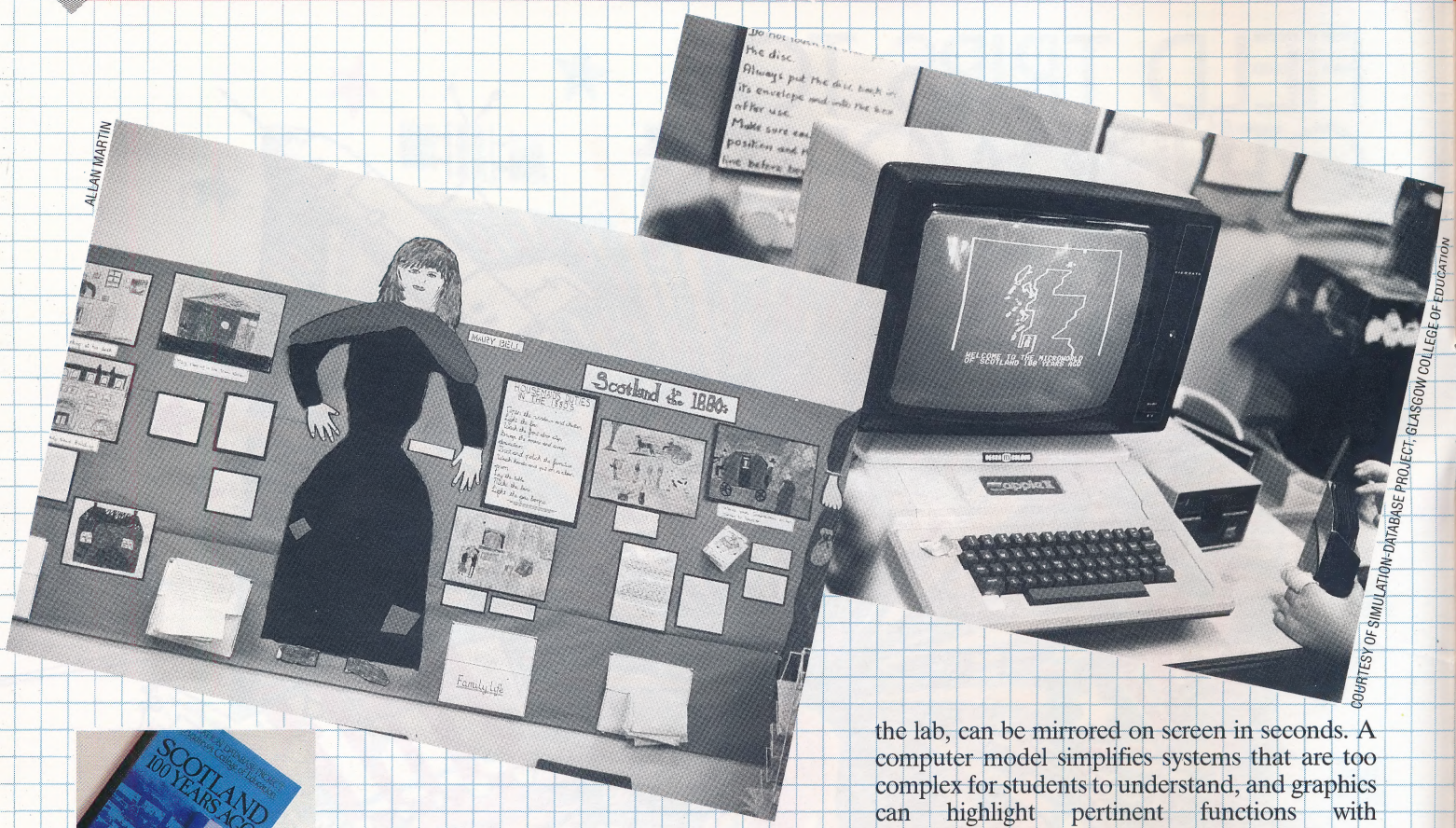
Games are fun, and all children and most teachers agree that learning should also be enjoyable. The use of games in general, and role playing in particular, has acquired a popularity in recent years concomitantly with the growth of drama-in-education and the development of 'game theory' in psychology. These developments lend even more credence to the computer's potential in schools.

The original space invaders-type game and the more recent adventure/simulation variety represent the two predominant genres of computer games. As software develops, the distinctions between the two become increasingly difficult to maintain, but most games can still be recognised as drawing their themes from one or the other.

Those of the space conflict variety have made little impact on educational software despite their popularity. While writing their book, *Mind at Play, The Psychology of Video Games,* Elizabeth and Geoffrey Loftus questioned a number of psychologists about the educational effectiveness of this style of game — the answers they received were not encouraging. As can be expected, it is the entertainment value of 'shoot-em-ups' that far surpasses anything educational, although research does show that this style of game, because of the co-ordination and reflexes required, can have therapeutic effects on sufferers of various types of mental disorders and brain damage.

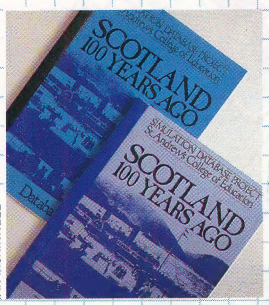
Programs like Spell Invaders and Maths Invader have been attempts at presenting traditional subjects in space invader format, having the child 'shoot down' the correct answers. But these games have proven to be less fun than the originals, and their educational benefit is dubious at best.

Simulations and adventures, on the other hand,



ALLAN MARTIN

COURTESY OF SIMULATION-DATABASE PROJECT, GLASGOW COLLEGE OF EDUCATION



IAN McKINNELL

Journey In Time
The Simulation–Database Project, developed at St. Andrew's College of Education, uses the computer as a tool to supplement — rather than replace — other learning activities. The package breaks down into three parts: an 'adventure game', a collection of text-files providing additional background information and a database holding further references and sources. The class is divided into groups, with each group taking the role of a character who (in this example) makes a long journey across Scotland in the 19th century. Additional activities springing from and accompanying the use of the database include mapwork, creative writing, drama and field trips

demonstrate a considerable potential as educational alternatives. Simulations use the computer to create learning environments hitherto unavailable to the child, and adventure games can replicate the fantastic scenarios of science fiction and myth, both without the expense and danger implicit in reality. These principles have already been employed in industrial training schemes and many of them are now being employed in schools.

Ballooning, a program by Hill McGibbon for the Spectrum and Commodore 64, is a typical simulation game, providing children with the opportunity to learn through playing with something they would not otherwise have access to. A fuel gauge, altimeter, temperature gauge and a rate of descent/ascent dial are enclosed in an instrument panel at the bottom of the screen, which act accordingly as the burner is turned on or off, and the gas valve is opened and closed. As the balloon descends to a certain height, the terrain becomes visible, and should refuelling be necessary there are several landing stations to which the craft must be brought down.

EDUCATIONAL ADVANTAGES

Besides being inexpensive and safe, simulations offer other educational advantages. Represented on a computer, the details of setting up an experiment or exercise are already taken care of. Thus children are left unencumbered to concentrate on the situation, what is happening, and why. Time is no longer an obstacle preventing complex and lengthy projects to be completed in class. Scientific experiments that can take days in the lab, can be mirrored on screen in seconds. A computer model simplifies systems that are too complex for students to understand, and graphics can highlight pertinent functions with representative data from relevant stages in the process. Because the simulation can be repeated under various conditions, the students can experiment indefinitely. The direction towards which the experiment travels, and the variable factors implemented, since they are all under the student's control, allow a greater understanding of the results and conclusions, and because the simulation is presented as a whole and not in parts, a proper perspective is attained.

There are certain limitations, however, inherent in simulations. Because a computer can store only a predefined number of variables, there is the danger of a simulation presenting an over-simplified view of its subject. Thus the ability to portray real-life situations is both their greatest attribute and perhaps their greatest flaw. In reality, it is the unforeseen that surprises us and forces a reassessment of our understanding of what is happening. For this reason, a program like Ballooning, with its fixed number of variables, will have its greatest benefit if used in a broader context, perhaps as a single aspect of the science and history of flight. Unfortunately, this limitation is often overlooked; a simulation is no substitute for the real thing. Their strength lies in stimulating further enquiry into associated subjects; not in providing ready-made responses, or as a simple extension of programmed learning techniques. There is, for example, a science program written for the Apple, which simulates a project connecting light bulbs and batteries in different ways. Presumably it has been written for the school that can afford an expensive computer but mysteriously has no access to batteries and bulbs.

The possible shock from a 6v cell can hardly justify the simulation on safety grounds.

CREATING MICROWORLDS

Most people, children and adults alike, enjoy indulging in fantasy. In recent years, books like *The Hobbit, 2001: A Space Odyssey,* and Robert E. Howard's Conan series, have proved immensely popular. Role-playing games like Dungeons and Dragons have not only been a tremendous commercial success but have also coincided with the widespread recognition of the value of role-play in therapy and education. Its interactive nature makes the computer an ideal medium for fantasy and role-playing games. Graphics and sound capabilities can more fully define a historical or imaginary world. Educationalists have been quick to grasp this potential of creating 'microworlds' on the computer, seeing them as a quantum leap in teaching practices.

There would seem, for example, no better way for a child to learn history than by travelling back in time and experiencing an adventure. But the child will need information to survive and so the program must contain a database that can be consulted during the game and be used as reference work away from the keyboard. The Simulation-Database Project was set up by Allan Martin at St Andrews College of Education in Glasgow 'to explore the way in which computers can contribute meaningfully to the work of the primary school, by creating a new kind of computer-based pedagogic object, the simulation database'. The simulation's structure is based on an adventure game: a child enters a historical world in which, when travelling between different zones, various situations and incidents have to be dealt with. Text files in the database relate to each of the zones and they can be accessed at any time. The programs are designed to be used as part of a large project in which most learning would take place away from the computer, which itself is used as only one of many resources.

A second database containing references to the zone or subject can be searched for sources of further information in books, films, and other media. To help integrate the program with other classroom activities, the software includes a handbook, map of the area, simulated letters, pictures and other reference material. Written in LOGO, the programs can be adjusted to local conditions — zones and data can be added, removed or altered to fit the particular objective of the individual school or class.

The first simulated database was based on Scotland in the 1880s. The package was used with a class divided into six groups, each taking an identity of a fictitious but authentic character. The characters represented each stratum of society from aristocrat to house-maid and each group was given a scenario requiring them to make a journey across Scotland. Once they had planned their journey with the map and moved into the first zone on the computer, the characters encountered various incidents. After each phase of movement, the children, using two databases and other reference sources provided by the teacher, investigated the newly arrived at zone. Activities generated out of these investigations included mapwork, diary writing, writing newspaper reports and other descriptive work. The children produced plays, taped 'interviews' with the characters, made visits to the areas covered in the game and invited outside speakers to the school. With discussion being encouraged throughout, as well as drawing and painting their new surroundings, the students and the teacher found it a most enjoyable exercise.

The simulation database project is an example of how a computer game can be used to interest children and stimulate work in associated studies. Two further packages from the same group are forthcoming: Palestine at the Time of Christ, for use in religious education, and The River, as an aid in environmental studies and science. A simulation like the one just described uses the unique qualities of the computer to produce an exciting new educational resource. It is an ideal example of the use of the micro as a medium through which a child can explore a subject and gain valuable experience, as opposed to its use as a tool, such as in word processing.

Man The Player

Many educators and parents pale at the thought of students spending large amounts of time 'playing' instead of 'working'. A recent television programme showed a group of young children exploring turtle graphics on the floor of a schoolroom. The commentator remarked that viewers should not be deceived by the unstructured nature of the exercise, saying: 'In fact, there is real learning going on here.'

However, many parents and teachers are beginning to believe that 'real' learning occurs only through play. There is a clear basis for such a radical viewpoint, not only in educational theory, but also in philosophy and history. A book by J. Huizinga — first published in 1944 when he was a professor of history at the University of Leyden, Germany — provides a strong argument that play is not only a learning process but in fact has substantial value in human culture on its own merits. The book, called *Homo Ludens* (or, *Man The Player*), shown here on the right, makes several points about the importance of play:

- Play is a significant function: there is a sense to it
- All play means something — it is far from a wasteful activity
- Play is both voluntary and instinctive. Children play instinctively to learn about the world around them, but they continue to play because it is fun
- Contemporary civilisation suffers from the formulation of play: the more structured play becomes, the less it is play



DOWN ON THE FARM



GEOFF ROWLEY

Farmers have for many years now been incorporating the use of computers into their farm management systems. The Farmfax suite of programs, in cartridge format and designed to be used on the Dragon microcomputer, presents a fully comprehensive suite for every major aspect of the modern farm.

Assuming that computers are being used by all sectors of society and business, the introduction of a selection of programs for farmers is not in the least surprising. What may be surprising is that the machine most of these programs are intended for is the previously foundering Dragon, which is now produced in Spain and is finding increased popularity amongst Europe's agriculturally based workforce.

One such group of programs is the Farmfax series, which is the brainchild of Geoffrey Paterson, himself a farmer. The practical application of each package is geared toward a specific farming task and is testimony to the fact that the people who wrote the specifications, and translated them into Assembly language knew what they were doing. The programs are designed to be easily understood by people unfamiliar with computers, and are based on ROM cartridges that are ready to use the instant they are plugged in. A number of them also have extra RAM on the cartridge for quick and easy storage of data, though users are advised to keep tape back-ups for safety's sake.

One example of a farmer taking advantage of the micro's inclusion in farm management is Robin Dunkerley, who farms 500 acres on the outskirts of Oxford. About half of his land supports his herd of 150 Holsteins and Friesians, the remainder being mostly arable. The sort of problem a Farmfax program has helped him to solve is determining the 'Calving Index' — the number of days a cow goes without giving birth, and therefore continuing her milk yield. 'It's been reckoned that if a cow doesn't calve at least once a year,' he explained, 'it costs you about £2 for every day over the 365 that she hasn't done so. In the past, we've slipped back to 390 days and worse, and 375 has been the average. The computer helps us to keep track of any that haven't been served to the bull, and so far we haven't missed one. Last year our Calving Index was only 366, and I doubt if we'll ever do better than that. It's as near perfect as we're likely to get.'

INDIVIDUAL COW RECORDS

As an indication of the scope of a program, the Individual Cow Records can hold all the records of up to 240 head of cattle within the cartridge; a cheaper version stores them on tape. Since all data is downloaded into memory and manipulated there before being SAVEd again, both versions work identically. However, though many would regard tape as a rather slow medium for data storage, to farmers who have had to cope with the tedium of calculating decisions such as when to sell a cow whose yield is dropping off and measuring what she is costing in foodstuffs against what she is bringing in by milk sales, it's time well spent. And, most significantly, they find it more efficient than renting time on a mainframe.

'We used to pay the Milk Marketing Board about £40 a month to keep our cow records on their computer,' recalls Dunkerley. 'We would send them the information at the end of the month, and we would get the printouts about three weeks later, by which time it was too late to act upon the information. Also, a lot of stuff they were

telling us might have been valuable to someone with a top pedigree herd, but it was absolutely irrelevant to a commercial dairy herd like mine, which is just geared up to producing milk.' Thinking for a long time that buying his own machine would make a lot of sense, Dunkerley wasn't, however, prepared to pay £2,500. But the Dragon, the printer and Farmfax programs, at between £600 and £700 complete, was within his budget. His outlay was quickly recouped and now he can access the information he needs whenever it's required, instead of paying for data that was often late and irrelevant.

At the end of each month now, Dunkerley spends about an hour organising and balancing his information. He first runs through the Dairy Management Program, which gives him financial results, the margin of income from milk over expenditure on feed, his gross income, how many litres of feed he is using to generate each litre of milk, the total production of milk and other relevant data. He then runs the Dairy Ration to ensure that he is producing the amount of milk he should be. He finds it very useful because it performs a complete feed analysis. By entering the amount of megajoules of energy, and how much digestible crude protein and dry matter is present in each kilo of food, and inputting the price and whether or not it is produced on the farm, Dunkerley can determine the total cost and bought cost of each cow per day.

FARMFAX IN ACTION

By entering the daily milk yield of each cow, the Individual Cow Records program will provide an absolutely accurate record at the end of each month. 'If any of the factors change', Dunkerley explains, 'the computer will recalculate everything for me automatically. That's the beauty of a computer: as a result, the time this sort of work takes is cut to less than a third. And it also tells me things I would always have liked to have known, like how my bank overdraft is going. As a tenant farmer, I am very vulnerable to that dreadful day, twice a year, when my rent falls due. I can also do an accurate forward budget, which makes my bank manager very happy.'

Each of the Farmfax program modules operates on a very user-friendly basis, and includes a small, simple manual requiring just a few minutes' study — usually a pocket-size fold-out sheet of paper. They all begin with a simple menu. For instance, Cullcow, which aids the farmer in decisions regarding cows whose milk yields drop below a certain level, first asks the farmer to choose between:

PAGE 1 — PRICE DETAILS
PAGE 2 — BEST BUY/SELL

On page 1, the farmer first has to enter the month and year under consideration, then the bank interest rate applicable, the milk price, the milk yield in litres per day from each cow, and the cost per month. As soon as these last four items have been entered, the same figures are transferred to all the other entries to save keying-in time, though they can be altered manually if it is thought, for instance, that interest rates may rise in subsequent months.

On page 2, the farmer is asked whether he is buying or selling cows, the number in the batch, the expected price and the sale month. The computer then compares all the figures and places a star beside the cow for the best offer. Pressing the ‹BREAK› key at any time will return the user to the index, to re-key any data if necessary.

The other packages operate on a similar basis, though obviously with the more complex programs the opening menu will be more complex. For instance, the Dairy Ration Formulation opening page offers:

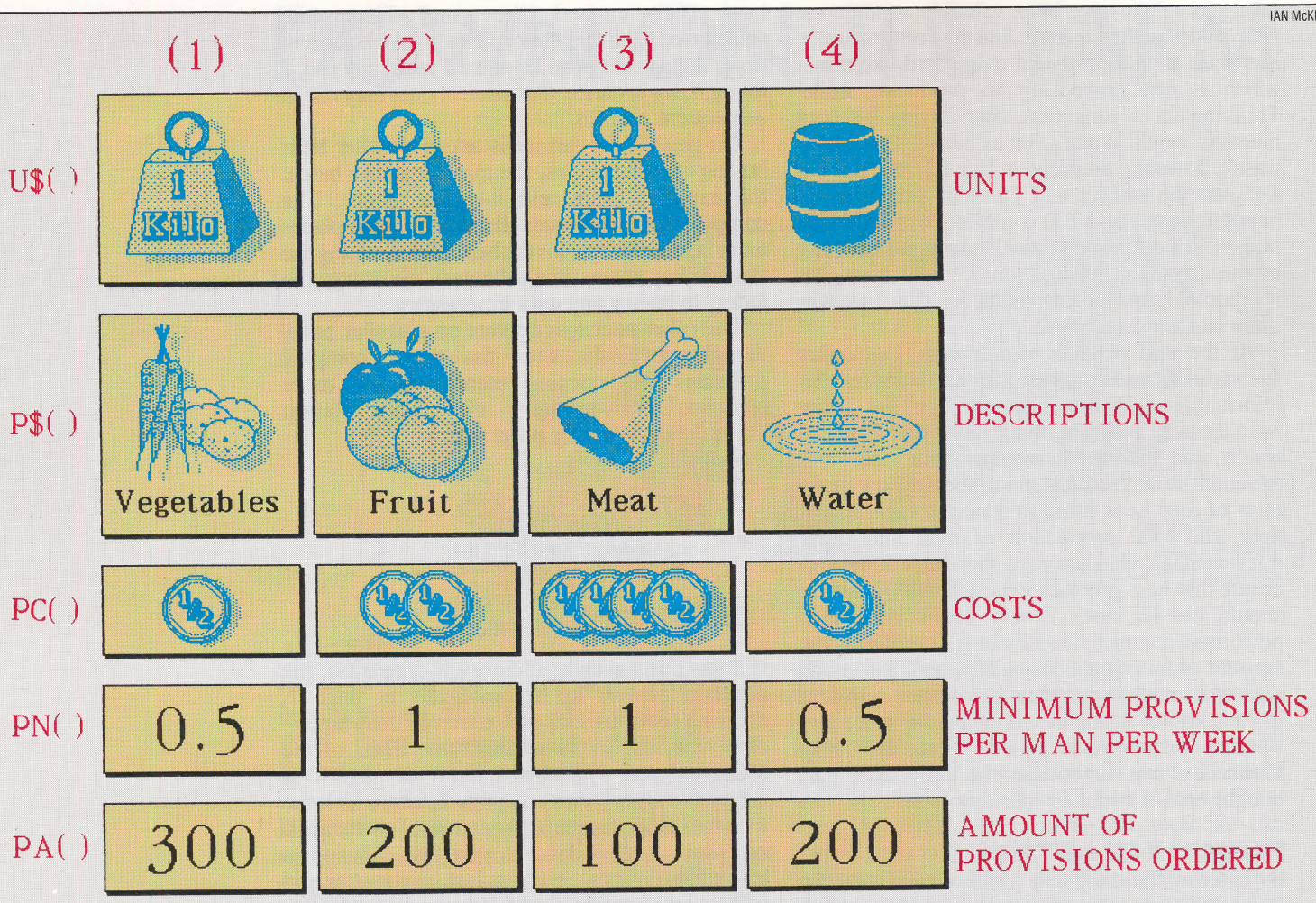
PAGE 1 — FOOD TABLE
2 — MINERALS
3 — COMPARABILITY
4 — GROUP DESCRIPTION
5 — FORMULATION
6 — FEED QUANTITIES
7 — FILING/RETRIEVING

If a cartridge without memory is employed, the program powers up automatically at page 7, allowing previous data to be retrieved from tape — otherwise, the menu is displayed. Each of the foods entered with their prices on page 1 is automatically repeated on page 2, into which the user can enter the mineral content of each, while on page 3 the data is analysed, issuing a breakdown of how much energy, dry matter and protein each supplies for £1. If the farmer enters on page 4 his production goals in terms of the cow's body weight, days in calf, litres of milk yield and butterfat content, it is then possible, with the program's aid in formulating his feed ration, to equate instantaneously the total food costs for a given number of cows for a given number of days.

Clearly, this is the sort of program written by specialists for specialists. Farmers like Robin Dunkerley appreciate the assistance it gives them in an increasingly industrialised agricultural world wrapped up in governmental and supra — governmental red tape. 'With the EEC quotas and the pressures increasing upon us all the time,' he concludes, 'we've got to run the farm as tight as we can, and the computer helps us to do that. I'm not sure it helps us to make more money, but it certainly saves a lot.'

Farmfax Suite: For the Dragon
Price: Buysell, Cullcow and Fertbuy, £56.35 each; Cash Accounts, £286.35/£401.34 with RAM; Super Fertbuy, £113.85; Cash Planner, Dairy Management, Dairy Prediction, Dairy Rotation Formulation, Field Records, Individual Cow Records, each £113.85/£171.35 with RAM.
Distributor: Farmfax Computer Systems, PO Box No. 2, Stockbridge, Hampshire SO20 6LE
Format: Cartridge

IAN McKINNELL



Provisional Requirements
The player can select four types of provision: vegetables, fruit, meat and water. The descriptions of these four types are held in the array P$() and corresponding units are held in U$(). To make decisions on the amount of provisions to buy, the player must be given two items of information: the cost of each provision and the minimum requirements for each crew member per week. This data is held in the arrays PC() and PN(). These four arrays are set up at the beginning of the program. A further array is filled as provisions are ordered. PA() holds the amount of each provision

SUPPLY LINES

We continue our simulation series by developing the second module of the program for the New World trading game. This subroutine is concerned with ordering food provisions for the crew on their voyage across the Atlantic.

In the first stage of the New World trading game the player hired the crew, and at the end of the hiring stage the instruction PRESS ANY KEY TO CONTINUE appeared. Pressing a key brings the second stage to the screen, at which provisions must be purchased. The player knows that the voyage will take around eight weeks, so sufficient food and water must be brought.

The program calculates the total requirements for eight weeks and checks that sufficient supplies have been purchased. If they have not, it prompts the player, saying someone may get hungry or thirsty, and provides an opportunity to re-order. If the player does not have enough gold to pay for the order then the program asks the player to try again (the program displays a running balance of money throughout the provisioning stage.)

As in the hiring stage, we need to set up a number of arrays for the provisioning stage.

```
19 REM **** PROVISIONING ARRAYS ****
20 DIMU$(4):U$(1)="KILO":U$(2)="KILO":U$(3)="KILO"
:U$(4)="BARREL"
21 DIMP$(4):P$(1)="VEG":P$(2)="FRUIT":P$(3)="MEAT"
:P$(4)="WATER"
22 DIMPA(4)
23 DIMPC(4):PC(1)=.5:PC(2)=1:PC(3)=2:PC(4)=.5
24 DIMPN(4):PN(1)=2:PN(2)=1:PN(3)=1:PN(4)=.5
```

The provisions are supplied in different units — vegetables, fruit and meat are measured in kilos and water is measured in barrels. Line 20 sets up an array, U$, with four elements to hold these four units. Recording the units in an array rather than continually printing them saves memory. If you wish to extend the list to include sacks of flour, for example, this array could easily be extended.

The descriptions of four types of provision are represented in line 21 by four elements of the array P$(). The program sets up an array PA() to store the amount of each provision purchased. A kilo of vegetables costs half a gold piece, a kilo of fruit one gold piece, a kilo of meat two gold pieces and a barrel of water half a gold piece. This information

is stored in the array in line 23. The order of provisions is not stated in this line, but set up by default. The array could easily be extended to include extra provisions.

To maintain good health each crew member needs a certain amount of each provision. The requirements are set up in an array, PN(), in line 24. For example, a member of the crew needs two kilos of vegetables per week, so the first element, PN1, is set to 2.

Line 500 in the first program module sends the program to the subroutine at line 1000 for hiring the crew. After hiring, it returns to line 500. To call the provisioning routine we must insert a GOSUB call after line 500, as follows:

```
550 GOSUB 2000 : REM PROVISIONING
```

A screen of text explaining the provisioning is printed. The variable CN, set up during the hiring stage, represents the number of crew hired. It is used in line 2040 to remind the player how many mouths there are to feed.

The programming for the provisioning is enclosed within a loop and is used four times, once for each type of provision. The value of subscript T increases by one each time. This is set up in line 2050. The screen prints the amount of each food type that each crew member requires per week. The requirements are determined using the relevant element of PN() as defined earlier.

As we have seen, the food is measured in kilos and the water in barrels. The required units are determined by the relevant element of the array U$(), defined in line 20. If more than one kilo is required, line 2075 adds an S to kilo; if it is singular, line 2080 replaces S with a space. Line 2085 prints OF and the name of the provision held by P$(), followed by FOR EACH WEEK OF THE JOURNEY.

The player is asked in lines 2100 and 2110 how many units of each provision he wants to buy. The requirements are typed in and represented by P$() in line 2130. The number is put in the first element of the amount array by line 2140. If the amount ordered is zero the program prints IF YOU DON'T BUY ANY followed by the provision array ;P$(T) at line 2160. The program then works out whether the amount ordered and entered in the PA() array is greater than that required for an eight-week journey, using the formula CN * 8 * PN(T). CN is the number of crew, PN(T) is the requirement of a particular provision, determined by T. This takes place on line 2170. If the supplies are insufficient a warning message is printed to remind the player that someone may go hungry or thirsty. The program checks whether it is going through the loop for the fourth time. If so it prints THIRSTY rather than HUNGRY.

If the player has been warned that the provisions are insufficient the program (at line 2230) asks: DO YOU WANT TO TRY AGAIN (Y/N)? If the player is short on funds and still accepts the entry, the program goes to 2400. If you want to make a different order the relevant value in the array PA() is set to 0. In this case the program has to go back to the same provision type. It does this by subtracting 1 from the loop count with T = T-1 in line 2250.

If the player is happy with the order made then the amount of money remaining is printed and a list of provisions ordered so far is displayed. The player may have bought sufficient provisions for the voyage but may not have enough gold left to pay the bill. Line 2260 checks whether this is the case. If so, lines 2270-2290 indicate there is not enough money to buy that amount of provisions and the player is asked to try again. If the purchase was valid the cost is subtracted from the remaining gold, held in MO.

A list of provisions bought so far is printed at line 2410 in another loop, controlled by TT, using variables U$(), PA() and P$(). The amount of gold remaining is printed at line 2480. The loop ends at 2490. The player moves on to the next stage by responding to PRESS ANY KEY TO CONTINUE. At this point control will return to the main program.

ERRATUM
In the listing for Module 1 on page 1053, the following changes to variable names should be made. Replace CD$() by C$(), ST$ by S$, IP$ by P$ and PK$ by K$ throughout the module

Module Two

```
 000 REM **** PROVISIONING ****
2005 PRINTCHR$(147):REM CLEAR SCREEN
2010 S$="    STAGE 2 - PROVISIONING*"
2015 GOSUB9100
2020 S$="    -------   ----------*"
2025 GOSUB9100
2030 GOSUB9200:PRINT
2040 PRINT"YOU'VE HIRED A CREW OF ";CN;"."
2045 GOSUB9200:GOSUB9200
2050 FORT=1TO4
2055 PRINT
2060 PRINT"EACH CREW MEMBER WILL NEED "
2070 PRINT"AT LEAST ";PN(T);" ";U$(T);
2075 IFPN(T)=1THENPRINT" ";:GOTO2085
2080 PRINT"S";
2085 PRINT" OF ";P$(T)
2086 PRINT"AT";PC(T);"GOLD PIECES PER ";U$(T)
2090 PRINT"FOR EACH WEEK OF THE JOURNEY."
2095 GOSUB9200:PRINT:GOSUB9200
2100 PRINT"HOW MANY ";U$(T);"S OF ";P$(T)
2110 S$="DO YOU WANT TO BUY*":GOSUB9100
2120 PRINT
2130 INPUTP$
2140 PA(T)=VAL(P$):GOSUB9200
2150 IFPA(T)>((CN*8*PN(T))-1) THEN2260
2160 IFPA(T)=0THENPRINT"IF YOU DON'T BUY ANY":GOTO
2180
2170 PRINT"IF YOU ONLY BUY ";PA(T);U$(T);
2175 IFPA(T)=1THENPRINT" OF":GOTO2180
2176 PRINT"S OF"
2180 PRINTP$(T);",":GOSUB9200
2190 PRINT"SOMEONE MIGHT GET ";
2200 S$="HUNGRY"
2210 IFT=4THENS$="THIRSTY"
2220 PRINTS$;"!":GOSUB9200
2230 S$= "DO YOU WANT TO TRY AGAIN (Y/N)":GOSUB910
0
2240 INPUTP$:P$=LEFT$(P$,1)
2242 IF P$<>"Y" AND P$<>"N"THEN 2230
2245 IF P$="N"THEN2400
2250 PA(T)=0:T=T-1:GOTO2410
2260 IFPA(T)*PC(T)>MOTHEN2270
2265 GOTO2400
2270 S$="YOU DON'T HAVE ENOUGH MONEY FOR":GOSUB910
0
2280 PRINTPA(T)
2290 PRINTU$(T);"S OF ";P$(T):GOSUB9200
2300 S$="PLEASE TRY AGAIN*":GOSUB9100:PA(T)=0:T=T-
1:GOTO2410
2400 MO=MO-(PA(T)*PC(T))
2410 PRINT:S$="PROVISIONS SO FAR:*":GOSUB9100
2412 GOSUB9200
2415 FORTT=1TO4
2420 PRINTPA(TT);U$(TT);
2430 IFPA(TT)=1THENPRINT"  OF ";:GOTO2440
2435 PRINT"S OF ";
2440 PRINTP$(TT)
2450 GOSUB9200
2460 NEXT
2480 PRINT"MONEY LEFT = ";MO;" GOLD PIECES"
2485 GOSUB9200:GOSUB9200
2490 NEXTT
2500 GOSUB9200:PRINT:S$="END OF PROVISIONING*":GOS
UB9100:GOSUB9200
2510 PRINT: S$=K$:GOSUB9100:PRINT:   GOSUB9200
2520 GETP$:IFP$=""THEN2520
2999 RETURN
```

Lines Of Supply
This subroutine deals with the provisioning for the voyage and should be added to the first module given on page 1053. The program loops four times, allowing the player to order one of the four provisions on each pass

Basic Flavours

Spectrum:
Make the following changes:
2005 CLS
2240 INPUT P$: LET P$ = P$ (1 TO 1)

BBC Micro:
Make the following changes:
2005 CLS

M

MATRIX

A *matrix* is a multi-dimensional array that is set up within an area of the computer's memory for the storage of data. In BASIC, matrices are generally defined by means of the DIMension statement. Each dimension of the matrix will have an associated index. This means that each cell within the matrix can be easily defined by the number of each index. The rows and columns in a two-dimensional matrix, for example, are defined as the x and y indices; and the expression (2,3) will refer to the cell positioned in the second row of the third column.

Matrices are widely used in computing for the fast storage and retrieval of information. Data organised within a matrix is obviously more easily accessed and manipulated than data that is scattered throughout the computer's memory. Matrices are particularly useful when dealing with databases or spreadsheets, where it is often necessary to perform arithmetical computations on entire rows or columns.

MEMORY HIERARCHY

Each kind of memory takes a different length of time to access. Sometimes a fast access speed is not important to the performance of an operation — sometimes it is critical. Thus, memory storage is organised into a *memory hierarchy*. Data is stored in the various types of memory depending on how fast the computer needs to access that data at a particular moment in time. If data held in a lower part of the memory hierarchy needs to be accessed at a faster rate, the computer will transfer it to a higher level.

In a microcomputer, the memory hierarchy is set up as follows. At the top of the hierarchy is the word (a few bytes) that is held within a register of the processor. Below that are the bytes held on the area of memory currently regarded to be zero page, and the rest of the addressable RAM follows. The information held on the track below the current head position of a disk or cassette is followed by the data stored on the rest of the disk or tape currently within the drive. Finally, the lowest level of a memory hierarchy is data held on disk or tape that is not currently in the drive.

As can be seen, access speed diminishes as data moves down the hierarchy. A byte of memory held within a processor's register takes 0.5 nanoseconds to be accessed, whereas a byte held on a tape not currently in a cassette deck could take several minutes. However, although information held on the lower strata of the hierarchy may take much longer to access, this is compensated by the enormous quantities of data that can be stored at the lower levels. This also means that the cost of data storage is cheapest at the hierarchy's lowest levels. For example, a processor register can typically store only a single word and the storage medium costs several pounds. On the other hand, a cassette or disk can store up to a megabyte of information at once, at a cost per byte that is only a fraction of that of a register.

MICROPROCESSOR

A *microprocessor* is the silicon chip, or group of chips, which comprise the microcomputer's CPU. At least two units are incorporated into the microprocessor: the Arithmetic and Logic Unit, which performs simple arithmetic functions, and the Control Unit, containing the stack and address registers as well as the program counter. The Control Unit also holds the instruction set — the machine's list of operating codes — and has the ability to both code and decode data (thus controlling the flow of information between the memory and other parts of the machine).

It is widely agreed that the first microprocessor was the Intel 4004. This was developed from an idea proposed by Edward Hoff in the late 1960s, and went into production in 1971. Since then, hundreds of different microprocessors have been designed and millions have been produced — the most successful being the Mostech 6502, the Zilog Z80 and the Intel 8088. Today, many of the world's top designers are working on inexpensive, sophisticated 32-bit microprocessors, which are remarkably more advanced than their four-chip ancestors.

MIDI

This term is an acronym for 'Musical Instrument Digital Interface'. *MIDI* is an internationally agreed configuration that allows computers and electronic musical instruments from different manufacturers to communicate with each other. This means that a wide variety of musical instruments and tape machines can be synchronised together using a computer or synthesiser as the control device.

The computer at the centre of the system assigns each instrument within the MIDI network an address. When the computer wishes to send a 'message' to a particular instrument, it transmits the message to the MIDI control interface. The interface attaches a start bit and an end bit to the eight-bit pattern. The ten-bit word thus created is then sent down the MIDI link and received by each of the devices in the chain. If the address in the message corresponds with the address of the receiving device, the MIDI will decode the instruction and execute it, otherwise the instruction will be ignored.



More Than The Minimum
An increasing number of MIDI interfaces are now available that enable a wide variety of micros to be used as control units. Although the basic specification for a micro to control a MIDI unit is simply the provision of MIDI-IN and MIDI-OUT ports, many manufacturers are exceeding this. The Roland MP401, shown here, features several more control ports than the necessary minimum

ACORN ADD-ON

The Acorn Electron has suffered from limited expansion potential with its lack of interfaces. The Plus 1 provided these but still prevented the incorporation of a disk drive system. The Plus 3 expansion module now provides disk capability and simultaneously elevates the Electron to the level required for serious micro users.

One of the major criticisms levelled at the Acorn Electron has concerned its lack of interfaces. Although the machine used the highly rated BBC BASIC, the fact that the Electron possessed only a pair of monitor sockets, an RF jack and a cassette interface (not even a joystick port was provided) meant the machine's expansion potential was severly limited. Acorn has always promised to provide expansion modules to enable users to upgrade their machines to the level of the BBC Micro in terms of available interfaces, but these add-ons were slow in arriving.

The first of the expansion modules to arrive was the Plus 1 (see pages 449-451), which did much to silence the critics of the Electron. But although it provided many interfaces not available on the original Electron, the Plus 1 did not, however, include a disk interface that would allow easily accessed mass storage of data.

This situation has now been improved with the arrival of the Plus 3, an all-in-one disk system for the Electron. The module is 'L' shaped with the disk filing system attaching along the back of the computer and the shorter disk drive unit bending around the right-hand side — the latter, unfortunately, covers the stock Electron power supply. The Plus 3 therefore comes with its own 36 watt power supply that plugs into the left-hand side of the unit and powers both the disk drive and the Electron. Like the Plus 1, the Plus 3 slides onto the edge connector at the back of the Electron, and to prevent the unit moving and possibly breaking the edge connector, two heavy screws affix the Plus 3 firmly into the Electron's base. Once fitted, the Electron and Plus 3 form a sturdy unit, which would be very difficult to break. Unlike the Plus 1, the new model also has its own edge connector fitted on the back to allow the Plus 1 — and possible future units — to be added.

Surprisingly perhaps, Acorn has chosen to fit one of the Sony $3\frac{1}{2}$ in disk drives to the Plus 3 as opposed to the usual $5\frac{1}{4}$ in disk drive format. There is also a second edge connector directly behind the disk drive to enable a second disk drive



CHRIS STEVENS

Building Blocks
Acorn's Plus 3 add-on for the Electron is designed to accommodate the Plus 1 as well. The provision of these interfaces allows the Electron user to upgrade the computer to a specification approaching that of the BBC Micro (with disk drive) for a total price of around £450

to be added, which Acorn claims will accommodate not only $3\frac{1}{2}$ in drives but also 3 in and $5\frac{1}{4}$ in drives. If the user wishes to add a BBC Micro $5\frac{1}{4}$ in disk drive, Acorn has made available an add-on interface to allow the unit to be fitted to the edge connector.

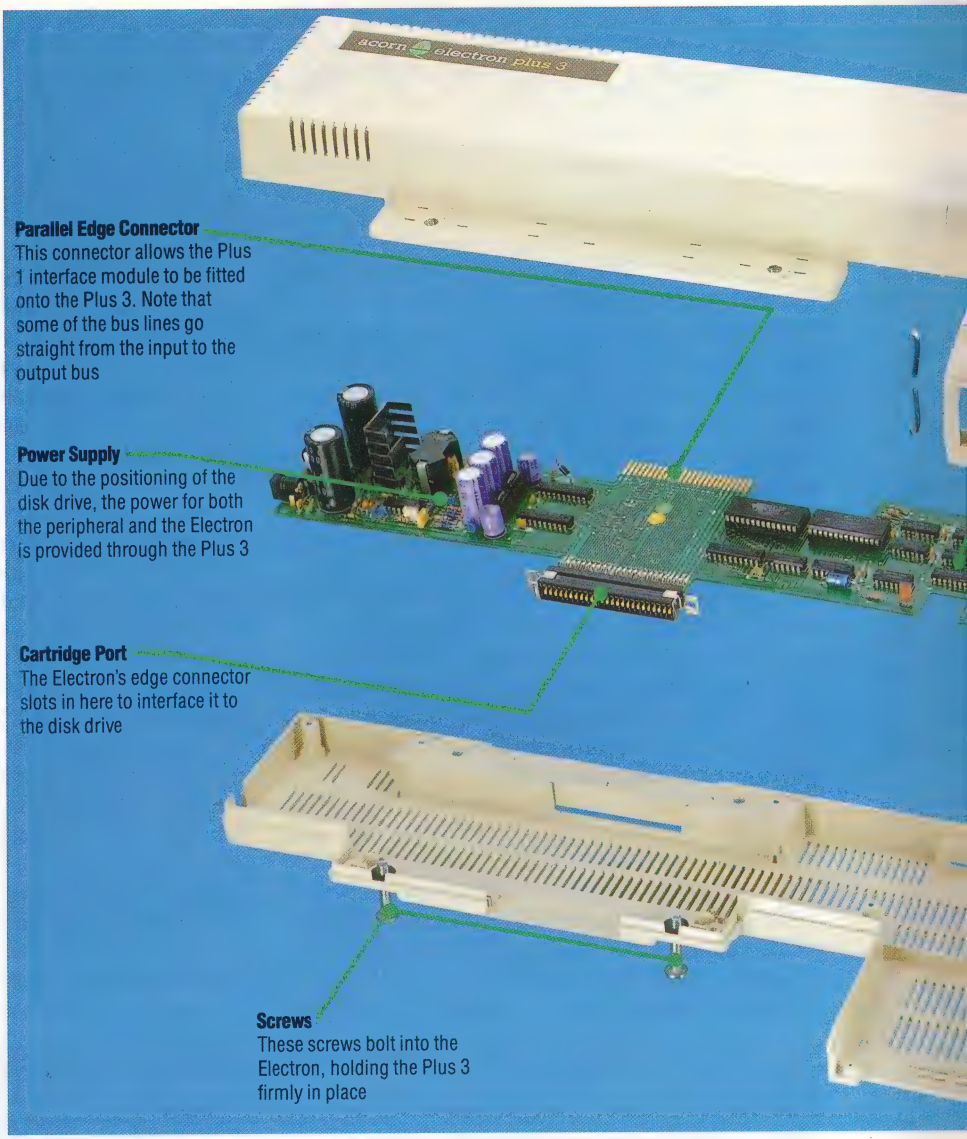
Inside the Plus 3, the device is divided into four parts. Beginning at the left is the power supply circuitry characterised by the large capacitors. Next to this are the buses leading from the Electron, some of them leading to the disk drive with others heading straight out again to the external edge connector that links to the Plus 1. The Disk Drive Filing system circuitry follows. The chips here contain the extra ROM-based commands enabling you to 'talk' to the disk drive and the disk management chips. Finally, on the far right, is the double-sided disk drive itself, encased in metal. This casing performs a dual role. Primarily it acts as a heat sink to prevent too high a temperature building up within the module, but it also acts as a Faraday cage to prevent stray magnetic fields interfering with the system.

The Sony disks are encased in hard plastic with a protective cover on the read/write window. When formatted, the disk is divided into 80 tracks of 16 sectors. As each sector can contain up to 256 bytes of information, the total capacity of a single-sided disk is therefore 320 Kbytes.

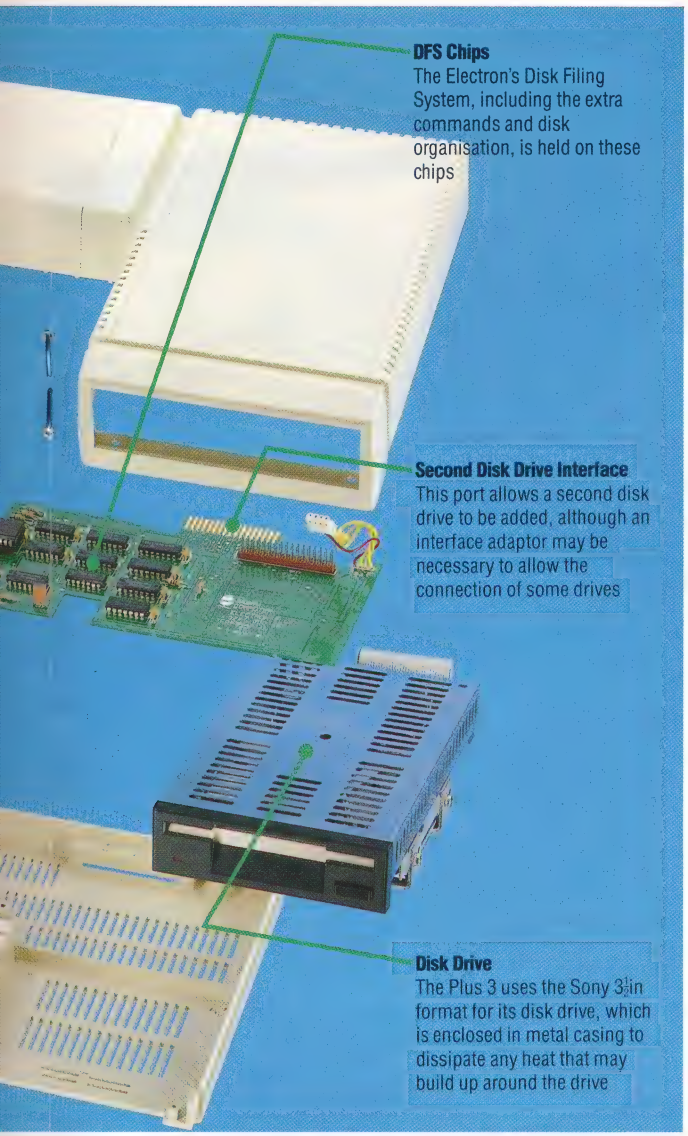
When the Plus 3 is fitted and the machine is powered up, along with the normal heading the message ACORN ADFS is displayed. This stands for Advanced Disk Filing System. Although the Electron DFS is very similar in many respects to the BBC Micro system — many of the commands are identical — there are subtle differences in the operation of the system. Provided with the Plus 3 is a Welcome disk containing demonstration programs similar to those on the cassette included with the Electron. However, the disk also contains a number of utilities invaluable to the operation of the disk filing system. Among the utilities provided are BACKUP, which allows entire disks to be copied, DIRCOPY, allowing only specific files to be copied from one disk to another, BUILD, which enables BOOT routines to be built in to the disk, and DUMP, displaying the contents of a file in hexadecimal and ASCII notations.

Like the BBC Micro DFS, the Electron's disk commands are prefixed with the * command, thus OS commands such as *LOAD, *RUN and *DELETE will be familiar to those who have used the disk filing system on the Micro. CATaloguing the disk will produce an identical display to that of the Micro, with the disk title, drive number, options, current directory name, and the current library name with the files, listed below.

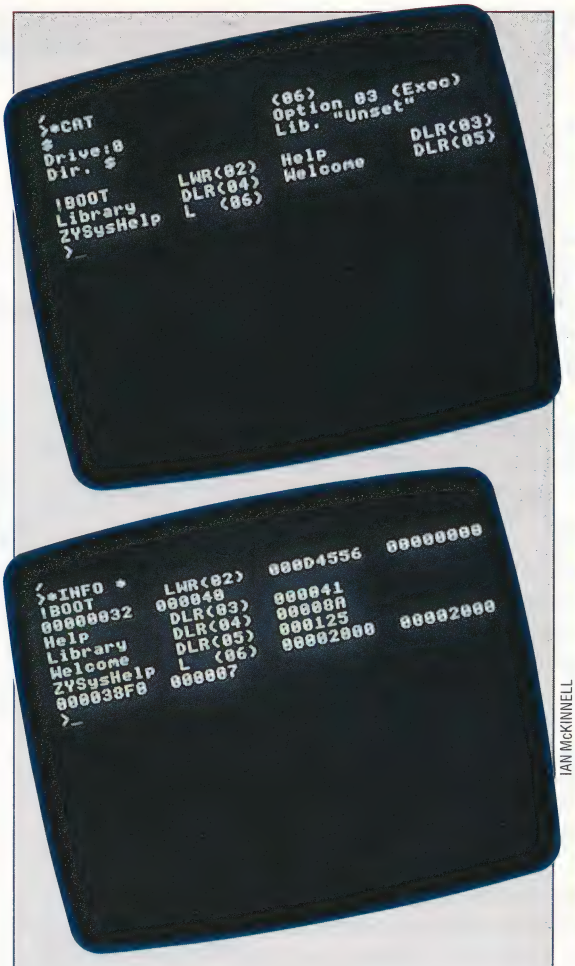
The primary difference between the two operating systems is the method of loading new disks. When the disk is inserted into the Plus 3 drive, the user has to inform the operating system of this by typing in the command *MOUNT — this has the effect of loading the Currently Selected Directory (CSD) and the Currently Selected Library (CSL) into memory. When the user wishes to change disks, the command *DISMOUNT is input. The effect of the *DISMOUNT command is to close all sequential files that may still be open, and to 'unset' the CSD and CSL. After the disk in the drive has been changed, *MOUNT is once again issued. Apart from the addition of these commands, the disk operating system appears to be identical to that of the BBC Micro.



The ADFS is organised in a series of file hierarchies. This enables files to be accessed quickly and simply by way of a series of pathnames. At the top of the hierarchy is the currently selected directory, which, on power-up defaults to the root directory $. To escape from the CSD to any other directory available on the current disk, the user simply types *DIR followed by the chosen pathname. Below the CSD can be a number of files, or a number of sub-directories known as Library files. Because the CSD can only hold up to 47 files at once, you have to organise them into libraries if you want to hold a very large number on the disk. These sub-directories can hold a number of files that may not be able to fit into the main CSD directory; thus, to access a file



CHRIS STEVENS



IAN McKINNELL

Acorn ADFS Screens
The Acorn Advanced Disk Filing System for the Electron is similar to that of the BBC Micro. The screen on the top shows the display in response to the *CAT command. *INFO displays the file's name, its access permissions, load, execution and disk addresses and the file's length

ACORN ELECTRON PLUS 3

PRICE
£229 inc VAT

DIMENSIONS
365×200×50mm (disk drive); 365×90×50mm (interface)

CAPACITY
320 Kbytes for a single-sided disk

INTERFACES
Power supply jack, cartridge-type input bus, edge connector output bus

DOCUMENTATION
The manual provided with the Plus 3 is thorough and gives the beginner a step by step approach to using the device. However, there are a few oversights that need correcting

STRENGTHS
At the price, the interface is good value and the user is provided with a disk filing system and a disk drive together

WEAKNESSES
As yet, there is little software available for the machine. Furthermore, to make full use of the disk drive, extras are needed such as a printer interface, which is only available on the Plus 1

called Fred on a disk with a number of directories with libraries, the format would be $.library1.fred. Although at first this might seem a long-winded method of accessing a file, it does have the advantage of allowing each of the libraries available on the disk to have a file named Fred. Provided the user remains in the correct library, reSAVEing the file will not overwrite the other Fred files held on the disk.

Like the BBC Micro, there are a number of useful utilities provided on the Electron's ADFS. There is a series of commands, for example, enabling you to name and rename disks, directories and files. There are also several commands available to handle file access permissions. This is performed by the command *ACCESS, which limits the accessibility of a file. For example, issuing the command *ACCESS file1 RL 'locks' the file, preventing it from being accidentally deleted, but also allows the file to be read. On the other hand, *ACCESS file1 WR permits the file to be freely written to and read from. To prevent anyone from copying a machine code file from the disk, suffixing the file with E will restrict the number of commands that can be used on the file to *RUN, and also limits the various methods of deleting the file.

The manual included with the Plus 3 is thorough, with a full tutorial course provided, and is up to the standard of documentation that has come to be expected of Acorn. There are, however, a few oversights in the manual, which, although minor, are nevertheless slightly annoying. As an example, the abbreviations CSD and CSL are mentioned on page 18 of the guide, but are not explained until pages 22 and 26 respectively.

Ever since the Electron was launched, it has generally been regarded as the poor relation, in the Acorn stable, to the BBC Model B Micro. The lack of interfaces, particularly regarding the omittance of a disk drive port, led many to conclude that the Electron was little more than a games machine that could not be regarded as an investment for serious applications. However, with the drop in price of the Electron and the availability of the Plus 1 and Plus 3 interfaces, the machine now looks a much more interesting proposition. The Electron has now become a force to be reckoned with.

BLUEPRINT

We start a new project in Workshop: to design and build a small machine capable of picking up and accurately moving objects around, to be run from the Commodore 64, Spectrum or BBC Micro. We will also develop sophisticated controlling software for each computer. In the first part of the project we consider the range of design options available to us.

Analysis of design options usually starts with defining the task we want performed. For the purposes of our project the nominal task is for the machine to be able to move from a starting point, pick something up (say, an object the size and weight of a cotton reel) and put it in a box, before returning to within 2.5mm of its starting point. We will assume that all the positions are known in advance. An industrial extrapolation of this task is a machine that packs items as they arrive along a production line.

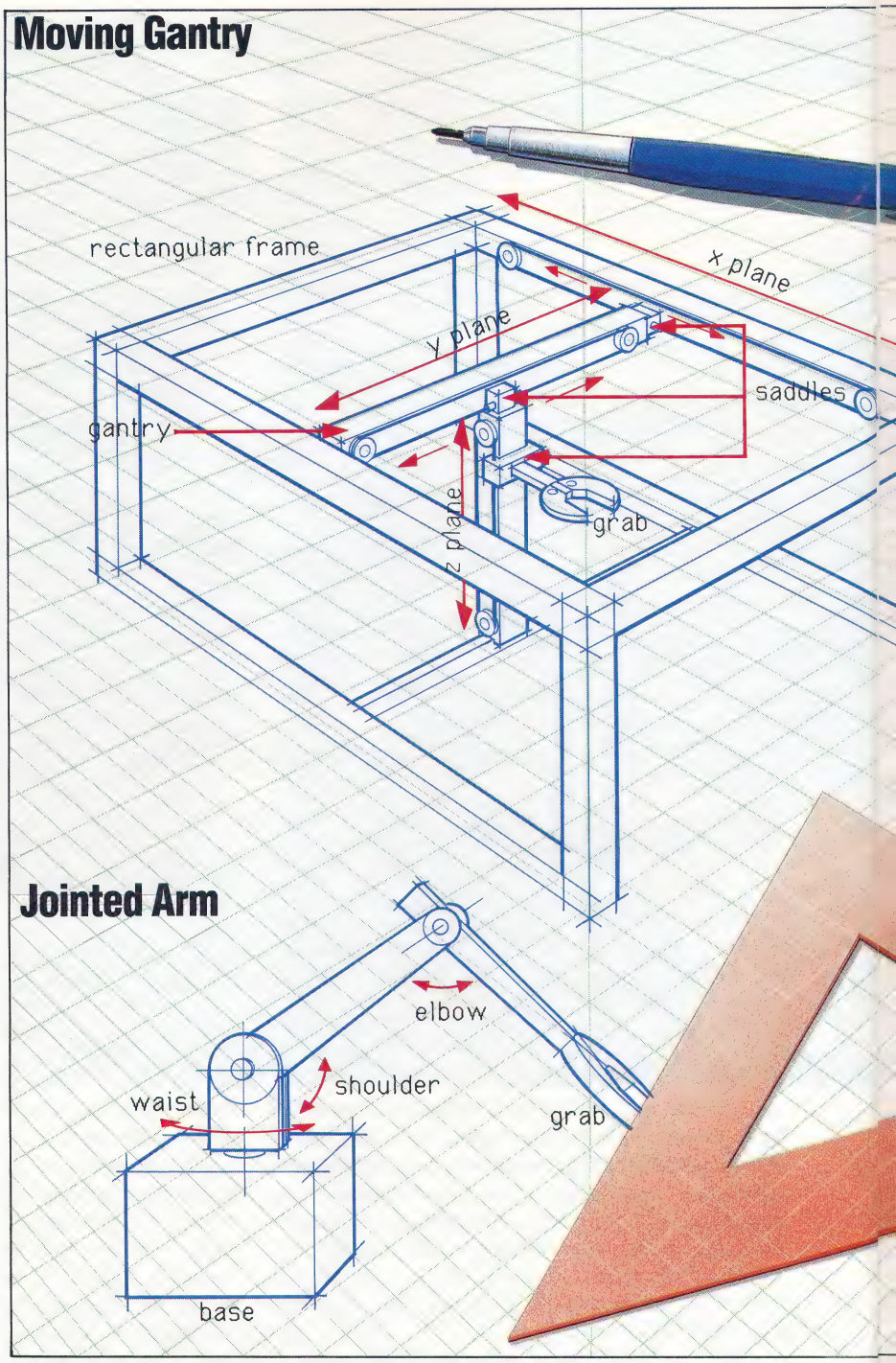
What sort of machine would we require to perform this task? Clearly it would need several motors. These motors could be hydraulic or pneumatic, but we shall consider only an electrical system of motors.

Three major electrical motor types suggest themselves, the most simple of which is the normal DC type. Although such motors are cheap it is difficult to control them accurately from a digital device, such as a home computer. To achieve the required degree of accuracy using this type of motor we would have to add shaft encoders to the motor spindles (see page 642). Each motor would then require decoding logic, either from hardware or software, to allow accurate positional control.

Stepper and DC servo-controlled motors are much easier to control by computer. However, stepper motors (see pages 835-837) are expensive, require all sorts of driver circuitry and are generally too large for our purposes. For a small-scale system like the one envisaged, servo motors have two important advantages: because they are mass-produced for the radio-controlled model market, they are widely available and relatively cheap.

We looked at how a servo motor works earlier in the Workshop series (see pages 912-914 and 922-925); to recap briefly, a servo motor system consists of a small 5v DC motor, controlled by a digital feedback loop. The angular position of the motor spindle is monitored by a miniature potentiometer and compared with the command signal in the built-in chip. The command signal is a 5v pulse of between one and two milliseconds' duration, the length of the pulse determining the angle that the motor spindle takes up. The command pulses must be repeated once every 50Hz (cycles per second).

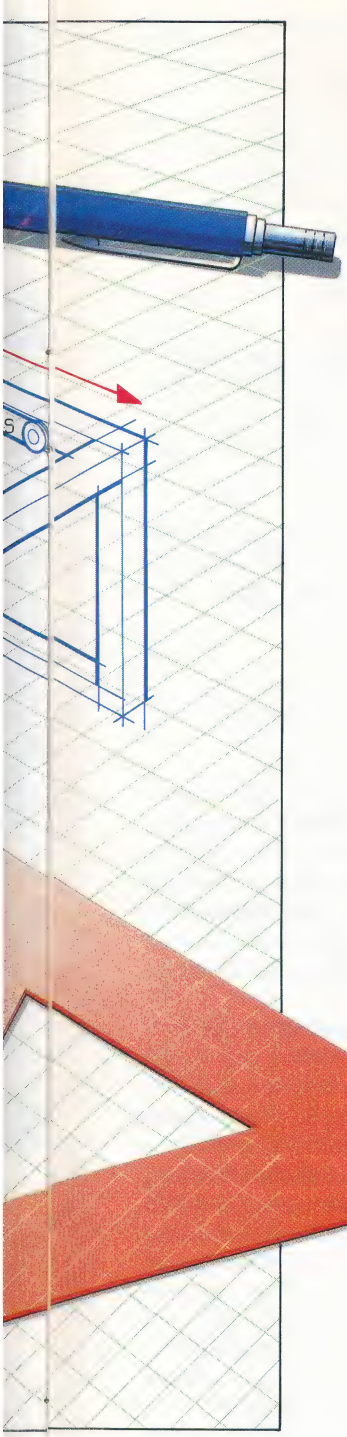
The driving software for both single and multiple servos has also been described in Workshop. The program is interrupt-driven, allowing BASIC software to run in the foreground and breaking in at regular intervals to send a new set of pulses to the motors.

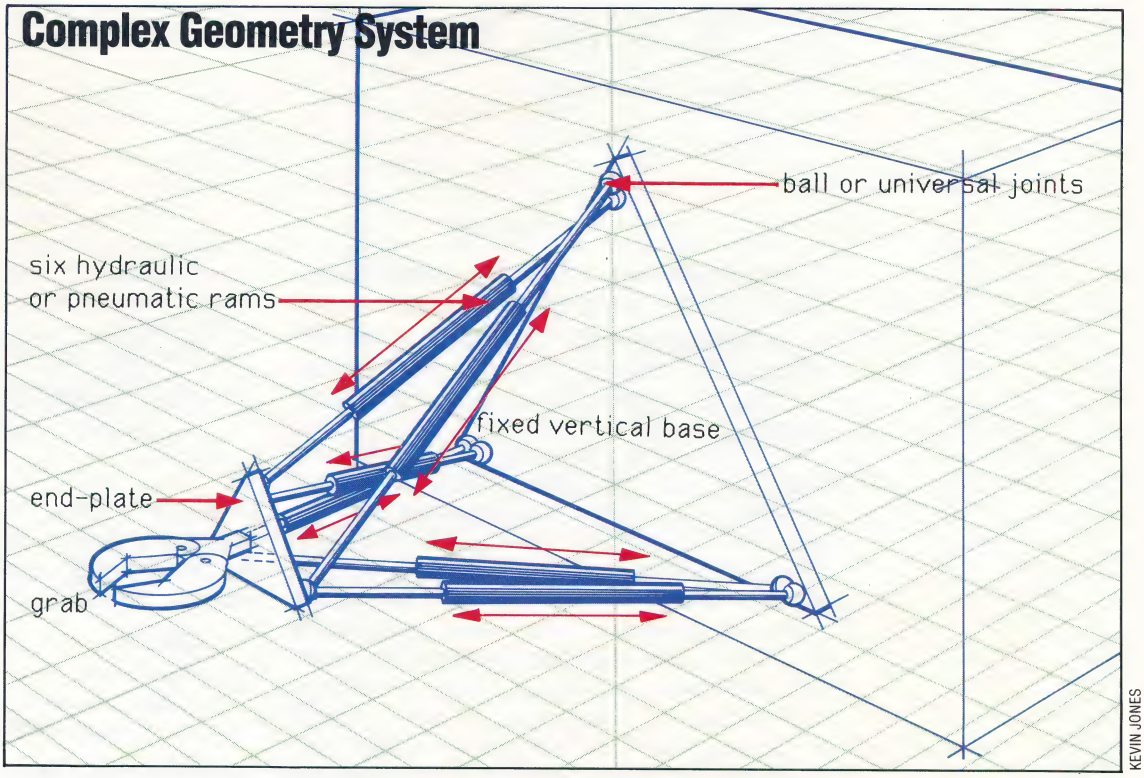


Possible Prototypes
Here are two possible designs for machines that can pick up a cotton reel and place it in a box or cup. The moving gantry cannot perform the more complex tasks that the robot arm is capable of. It does have the advantage of simple movement geometry; the robot arm motors require much more complex control to achieve simple linear movement of the grab. However, computer control means that these complex calculations are easy to handle

GEOMETRY OF THE MACHINE

Many possible designs suggest themselves for a machine to perform the task we have set, but first we must consider the design constraints. The first





KEVIN JONES

Complex Geometry
Computer control of electrical, hydraulic and pneumatic motors means that machines can be devised that have great flexibility. The complex interplay of the various motors that move sections of the machine can be handled by sophisticated software, which is, to a large extent, hidden from the machine user. Imagine the complex manipulations that are possible with this six-ram system, all six extending rams being under software control from a computer.

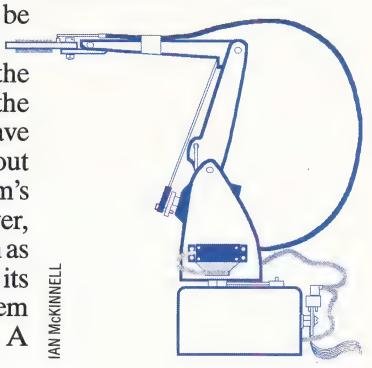
consideration is that to pick up an object such as a cotton reel we require a grab mechanism. This must use a minimum of one motor. Secondly, the 'grab' must be moved in three planes. The machine must be capable of moving at least 10cm from left to right (the x plane), a few centimetres back and forth (the y plane) and at least 10cm up and down (the z plane). Each plane of movement will require at least one motor, and the grab must be manoeuvred by a combination of these motors with an accuracy of at least 2.5mm to enable it to return to its starting point. The grab must also be able to pick up a weight of at least 20 grams, and its jaws must be able to open at least 3cm to accommodate an object the size of a cotton reel.

One design that would meet the constraints would be a small gantry system based around a rectangular frame. Gantry and grab could be moved by attaching them via saddles to wire pulleys, which could in turn be connected to servo or stepper motors. This system would require four motors, three to control movement and one to control the grabber action.

A second possibility, and one that more closely models the human form, is a jointed arm. The rotation of each joint could be controlled by a single motor. Digital servo motors are well-suited to this application, as the motor angle is easily directed from computer software, allowing accurate control over the arm members. Moving the grab by means of a robot arm requires more complex control than making the same movement with a gantry. For example, to move the grab in one plane only — say, the x plane — requires simultaneous control of at least two motors in the jointed arm system, this movement could be achieved using only one motor in the gantry system. However, as the system will be controlled from a computer, complicated control sequences do not present a great problem: they can be accomplished by use of clever software.

The jointed arm has one advantage over the gantry — it is a more flexible system. While the gantry could perform the specific task we have outlined it could not perform other tasks without physical rearrangement of the system's components. The jointed arm system however, could perform modifications of the set task such as placing a cotton reel in a box that was lying on its side. We will therefore select the robot arm system as the basis for our constructional project. A diagram sketching the basic design is given.

Colour By Numbers
In each instalment of the project, we will colour the relevant section of this line drawing to indicate the part of the robot arm being dealt with



IAN McKINNELL

1+2=20?

The discipline of PASCAL's data classification and the great variety of possible data types are features that make the language ideally suited to the clear-cut logic of mathematics. We look at aspects of PASCAL's arithmetic abilities.

What is one plus two? Without further information we might reasonably venture the answer: three, of course. The implicit assumption is that we are dealing with pure natural numbers, but suppose we put one 10p coin and two 5p coins in a vending machine? The result could be 20 pence — or, viewed from a different angle, a cup of coffee and two pence change. This sort of data classification is fundamental to any problem description, and PASCAL helps us to organise and describe our data in a clear and logical manner.

As we have already seen, we can create new and special categories of data so that we can think about the solution to the problem and design algorithms based on the processing of each data object in a manner appropriate to its type. If we inadvertently try to do something silly, like reading a Boolean truth value from the keyboard, the PASCAL compiler will trap this logical error immediately. You can appreciate the hours of frustrating debugging that can be saved if a language compiler won't even let you execute a single instruction until every error has been purged from the source program! The incompatibility between different data types and the richness of the ways in which we can describe them is one of the most valuable qualities in PASCAL. Far from feeling restricted in what we can do, we shall very often wish to take further advantage of PASCAL's 'strongly typed' data descriptions, and deliberately impose some additional constraints of our own on variables.

Now that we have used all the simple types of variables (those that may only have one value), we can fully summarise the rules for manipulating them. None of the scalars are 'compatible' with a variable of any other scalar type, though the two numeric types have many characteristics in common, and are allowed to 'rub shoulders' on occasion. As a real number can approximate a whole number, assignment is possible from integers to reals, but never vice versa. Here are some examples:

```
Program Compatibility (input, output);
VAR
   intA,
   intB  :integer;
   Xreal,
   Yreal :real;
BEGIN
   read (intA, Xreal); {read an integer, then any legal
      number}
   Yreal := intA; {real := integer is OK}
   intB := Xreal; {** ERROR : illegal **}
   {. . etc.
```

Arithmetic operations are, not altogether surprisingly, defined only on the numeric types. For both integers and reals, the usual four symbolic 'operators' may be used:

+ addition
− subtraction
* multiplication
/ floating point division

In this context, they are 'dyadic' or binary operators, as they always require two 'operands' of either numeric type. Whenever one of the operands is real, the result of the expression will be real—so that: 2 + 2.0 is 4.0 (not 4). In the case of the division symbol, the expression evaluates to a real even when both operands are integers: 3/5 is 0.6, 8/4 is 2.0.

When we are dividing integral values, a 'real' answer often makes no sense. Twelve chocolates divided among ten people gives one each with two chocolates left over, for example. The integer result and remainder can be obtained with the two integer division operators, DIV and MOD (both of which are reserved word systems in PASCAL): 15 DIV 5 = 3 and 15 MOD 5 = 0; 31 DIV 7 = 4 and 31 MOD 7 = 3. Be careful if your data is likely to be negative, as integer division is not defined for any negative values of denominator. With both types of division, real and integer, any attempt to divide by zero will result in a run-time error—at least until someone invents a way to evaluate infinity.

Multiplication and division operations are, of course, evaluated before addition and subtraction in an expression. The usual bracket notation must be used to override this logical 'precedence'. For example: (8 + 4) DIV 2 = 6, but 8 + 4 DIV 2 = 10.

TRANSFER FUNCTIONS

Although we cannot make direct assignment of a real value to an integer, PASCAL provides very useful 'transfer functions' with the identifiers trunc and round, which may be used to perform this mapping. Trunc simply truncates a real number disregarding its fractional part, whatever it may be, so that trunc (3.999) = 3 and trunc (−123.456) = −123. The function round performs intelligent rounding, towards zero if the fractional part is below 0.5 in

value and away from zero otherwise: thus, round (1234.5) = 1235 and round (–0.49237) = 0. Naturally, an error will occur if the real argument to either of these functions would give an integer result that lies outside the scale of the type integer (–MaxInt to MaxInt), so always check with an IF statement first. Also, the argument *must* be real, not integer (they are already truncated and rounded, after all).

There is one nicety that will be unfamiliar to programmers who have agonised over expressions using the INT function in BASIC. PASCAL has a predefined function that returns a Boolean result: false if its integer argument is an even number and true if it's odd. The identifier used is, not surprisingly, called odd.

```
IF odd (N)
   THEN
      WriteLn ('That's odd!')
   ELSE
      WriteLn ('That makes us even.')
```

or:

```
IF odd (N) THEN
   IF N MOD 2 = 0 THEN
      WriteLn ('Zap the compiler!')
```

The table shows all of PASCAL's arithmetic functions. They may all take any single numeric argument, real or integer, and the first two, abs and sqr, will return a value compatible in type with their argument. Thus: abs (– 19.372) is 19.372, and abs (255) is 255. The other functions will, however, *always* give a real result, so that sqrt (16) is 4.0, not 4. This is because the algorithms used to evaluate them are modelled on the summation of a series of terms, all of which are fractional.

There is an important lesson in style here with regard to the comparison of reals. Notice that we used the word 'is' rather than an equals sign when referring to the real results of the arithmetic functions. We mean 'is the same as' rather than equal to, because it is *never* safe to compare real values for exact equality. The slightest error in calculations will mean that two nominally 'equal' reals may in fact differ by, say, 1.0E-27. Insignificant, of course, but the computer doesn't know that. Instead of using IF X = Y THEN .. we can use PASCAL's predefined abs function:

```
IF abs (X – Y) <SmallestSignificant THEN {etc.}
```

Here we test for when the difference between X and Y is extremely small. Exactly how close we regard as insignificant could usefully be specified in a CONST definition at the head of the program. Logarithms are given to the natural base (e), not 10, and exp raises e to the power of its argument (that is, it 'exponentiates'). There is no actual exponentiation operator in PASCAL, which was a deliberate design decision by Wirth. How often have you seen BASIC programs doing silly things like saying:

```
500 LET D = B^2 + 4 * A * C
```

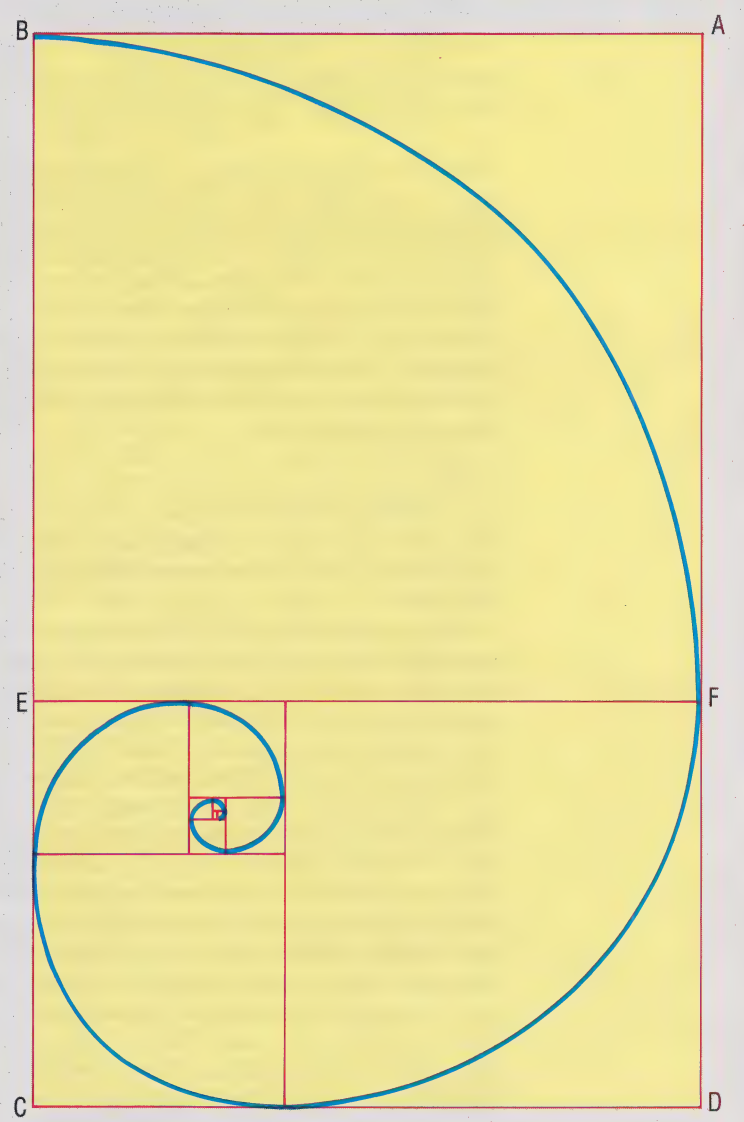
which uses the slowest and most inaccurate way

The Golden Section

There is only one point on a line which bisects it so that the ratio of the length of the smaller section to the larger is the same as the ratio of the larger section to the length of the whole line. This ratio is called 'sectio aurea', or the 'golden section', and like pi is an irrational number with an infinite number of digits



The golden section was used by Greek artists and architects (notably in the design of the Parthenon) and is often revived in theories of proportion. However, some of its most familiar manifestations are to be found in nature. The 'sea-shell' spiral is based on the golden section, since its radius vectors, separated by 90°, are in this section ratio



The rectangle ABCD has sides in the golden section ratio. A square ABEF can be delineated within this, such that a rectangle with the same ratio of sides as ABCD is formed (EFDC). The process can be continued ad infinitum and the corners of the squares are points on the 'golden section spiral'

LIZ DIXON

possible of evaluating the square of a number? B * B is far better, of course. Notice that PASCAL's sqr (N) returns the square of N directly as a true integer (assuming N is an integer); sqr (X/3) would give a real. Notice also another BASIC habit to start breaking: sqrt (it) *not* sqr (it) if you want the square root of a number.

SUBRANGE TYPES

All the scalar types have a defined and ordered range of values, and they may be used as a 'host' type from which we can derive new subrange types. These are composed, as the name suggests, of a contiguous portion of the host type. The lower and upper bounds required to delimit the subrange are listed between the 'ellipsis' symbol (..) in the definition of the subrange type identifier; for example:

```
TYPE
  byte = 0 .. 255 {subrange of integer}
  alpha = 'A' .. 'z'; {subrange of char}
  suit = (club, diamond, heart, spade); {a new type}
  major = heart .. spade; {subrange of suit}
```

Apart from the advantages we discussed earlier, a good optimising PASCAL compiler will store variables of the type byte, for instance, in eight bits instead of a possible 32 (if four-byte integer representation is used). Thus a large array of this type will take up only 25 per cent of the memory that would be used with unqualified integer elements. Notice that the type major has two possible values, rather like thinking of Boolean values being defined as:

```
TYPE
  boolean = (false, true);
```

All subrange types inherit both the data classification and the operations defined on their host type.

The previous definition of alpha may allow more than just letters of the alphabet, of course. In particular, the ASCII character set has six other symbols in the gap between 'Z' and 'a', including the square bracket symbols ('[' and ']'). These are also among PASCAL's 23 reserved symbols, and are used to delimit the components of some structured data types, as in many languages. BASIC also used these symbols originally for enclosing array indices, but the syntax was changed to round brackets because older computers sometimes had 'half-ASCII' character sets that did not include lower case letters, square brackets, the braces used for delimiting comments in PASCAL, and some other symbols. The unmodified Apple II also suffered from this restriction. PASCAL provides the following optional alternatives for cases where this problem arises: (. and .) are allowed in place of [and], and (* and *) may be used instead of { and }. The second alternative, for comment delimiters, is quite common; however, only ISO-validated compilers are likely to support the square bracket alternatives. Either, or both, substitutions may be made, so that: (* this is a legal comment}.

Function	Returned Value
abs (K)	absolute value of K
sqr (K)	square of K
sqrt (K)	square root of K
sin (A)	sine of A radians
cos (A)	cosine of A radians
arctan (T)	angle (in radians) with tangent T
ln (K)	natural logarithm of K
exp (L)	e raised to the power L ('anti-ln' of L)

Band Of Gold

The program Golden, given here, generates Fibonacci terms and prints them together with their ratio. Using the program, it can be seen that one of the interesting properties of the Fibonacci series (1, 1, 2, 3, 5, 8, 13, etc.) is that the ratio of each successive pair of terms increasingly approximates to the value of the golden section.

The program demonstrates some of the principles already discussed in the course, including further examples of PASCAL's REPEAT structure. As an exercise, you could try to alter the loop's terminating condition so that it stops when the next Fibonacci term to be calculated exceeds MaxInt

```
PROGRAM      Golden     ( output );

CONST
        Epsilon     = 1.0E-08;

TYPE
        Fibonacci   = 1 .. MaxInt;

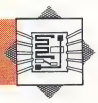
VAR
        first,
        second,
        next     : Fibonacci;
        ratio,
        Gold     : real;
        count    : integer;
BEGIN
  WriteLn ( 'The Golden Section' : 30 );
  WriteLn;
  WriteLn ( 'Fibonacci Series :' );
  first := 1;   { by definition }
  second := 1;
  ratio := first/second;
  count := 0;

  REPEAT
    IF count MOD 10 = 0 THEN
      BEGIN { Heading every 10 lines }
        WriteLn;
        WriteLn ( 'First' : 10,
          'Second' : 10, 'Ratio' : 14 );
        WriteLn
      END;

    WriteLn ( first : 10, second : 10,
                ratio : 16 : 8 );
    count := count + 1;
    Gold := ratio;   { remember old GS }
    next := first + second;
    first := second;     { move up . . }
    second := next;      { the series }
    ratio := first / second

  UNTIL abs ( ratio - Gold ) < Epsilon;

  WriteLn;
  WriteLn ( 'The Golden Mean is : ',
        100 * ratio : 10 : 5, '%' )
END .
```



WE INTERRUPT THIS PROGRAM...

Interrupt handling allows the OS to manage the processing resources of a computer, using signals called 'interrupts' and 'events'. This aspect of the operating system's activity is the last major area that we will look at in our series on the BBC Micro.

When you are running a BASIC program on the BBC Micro, it's easy to believe that the entire resources of the machine are dedicated to running that program. This is, however, not so; other tasks, such as reading the keyboard, are constantly being undertaken by the BBC Micro's operating system. This apparent 'time sharing' of the computer's resources is made possible by the use of *interrupts*. Essentially, an interrupt is a signal to the CPU that instructs it to stop what it's currently doing and perform another task. Once this task is completed, the CPU returns to its original job and continues with it as if nothing had happened. Interrupts on the BBC Micro are responsible for such diverse activities as keyboard reading, flashing colours, ENVELOPE command processing and A/D converter readings. Let's delve a little into how the machine actually does this.

Interrupts on the BBC Micro are generated by a variety of different hardware devices in the computer system; these include the Serial Interface chip, the User and System VIAs (Versatile Interface Adaptors), and the Econet and Disk Interface hardware. These devices give two distinct types of interrupt to the 6502 processor. Both *cause* the system to stop what it's doing, but you can instruct the CPU to ignore one type of interrupt if you want. These are called *maskable interrupt requests*, or IRQs. The other type of interrupt will never be ignored by the CPU, and is thus a high priority interrupt. This type is called a *non-maskable interrupt*, or NMI for short.

NMIs on the BBC Micro are reserved for devices that need immediate attention from the CPU, whereas IRQs are generated by devices that want the CPU's attention, but can afford to be a little more patient! The only two devices in the system that use NMIs are the Econet network hardware and the disk filing system. On receipt of an NMI, the OS makes use of a small machine code routine in page &0D. This is one of the reasons page &0D is out of bounds to the user once a disk interface is fitted. Because of these special properties of the NMI, we won't give it any further consideration. The programmer is specifically warned off using it by Acorn.

This leaves us with IRQs, and so let's look at how these are dealt with by the BBC Micro. Maskable interrupts on the 6502 processor can be disabled by the SEI instruction from assembler programs; they can be re-enabled with the CLI instruction. When an IRQ is received by the 6502, an action is taken that depends upon the source of the interrupt. The 6502 finds out the origin of the interrupt by 'asking' each possible interrupt source if it was the cause of the interrupt. As soon as the source has been found, the interrupt can be acted upon, and the device that generated the interrupt is effectively told to forget the interrupt until another event happens that warrants another interrupt being sent to the 6502. This is called *clearing* the interrupt. The machine code routine that deals with an interrupt is called an Interrupt Service Routine, or ISR.

WHAT DOES AN ISR DO?

As the program that has been interrupted mustn't detect any changes in the values held in the 6502 registers, the first task of an ISR is to preserve the values in the registers. These are saved on the 6502 stack by a series of push operations. They can then be restored after the interrupt has been dealt with. The ISR must then do whatever is necessary to service the interrupt, and it must clear the interrupt condition before leaving the ISR and returning control to the interrupted program. The reason why it is vital that the interrupt is cleared is fairly obvious; if it were not cleared, then as soon as control was returned to the interrupted program the 6502 would receive an IRQ from that source again—putting the CPU into a never-ending loop.

Now we've looked at the general principles involved in interrupt handling, let's consider the specific case of the BBC Micro. This computer is heavily dependent on interrupts, using them for a great variety of operations. For this reason, care must be taken not to disable the interrupts using SEI for too long. If we keep the period of time for which interrupts are disabled below a few milliseconds, we won't detect any difference. Any longer than that and the OS will start misbehaving. The following program demonstrates what happens to the BBC Micro when you disable the interrupts for any length of time:

```
10 MODE 2
20 DIM C (100): REM set up space for machine code
30 FOR I%=0 TO 2 STEP 2
40 P%=C
50 [OPT I%
60 .code SEI / turn off interrupts
70 RTS
80 ]:NEXT
90 COLOUR 14
```

```
100 PRINT "Hello!"
110 ENVELOPE 1,1,4,-4,3,10,20,20,127,0,0,-5,126,126
120 SOUND 1,1,160,200
130 TIME=0
140 REPEAT
150 PRINT TIME,I%
160 I%=I%+1
170 IF TIME > 100 THEN CALL code
180 UNTIL FALSE:REM go forever
```

RUN the program and you will see that as soon as the interrupts are disabled by calling the machine code routine code, the variable TIME is no longer altered and both the flashing colours and the ENVELOPE processing stop. However, the machine does not 'crash', because the routine continues to print I% out correctly. Be sure to press CTRL BREAK when you're finished with this program.

Garçon!!
It's not hard to imagine the BBC Micro's CPU as performing a similar role to a waiter in a busy restaurant. Just as the 6502 must constantly divide its attention between servicing peripherals, managing data transfer and program execution, so the waiter must attend to important customers (who demand priority over others), clearing tables and dealing with the occasional crisis.

An efficient waiter will be able to divert his attention to each of these tasks when required and yet still be able to return to where he left off, ensuring a steady flow of food from the kitchen to hungry customers. The 6502 operates in a similar fashion, responding to 'interrupts' or 'events' generated either by hardware or through software control. Once the interrupt has been serviced, the CPU can return to its previous task

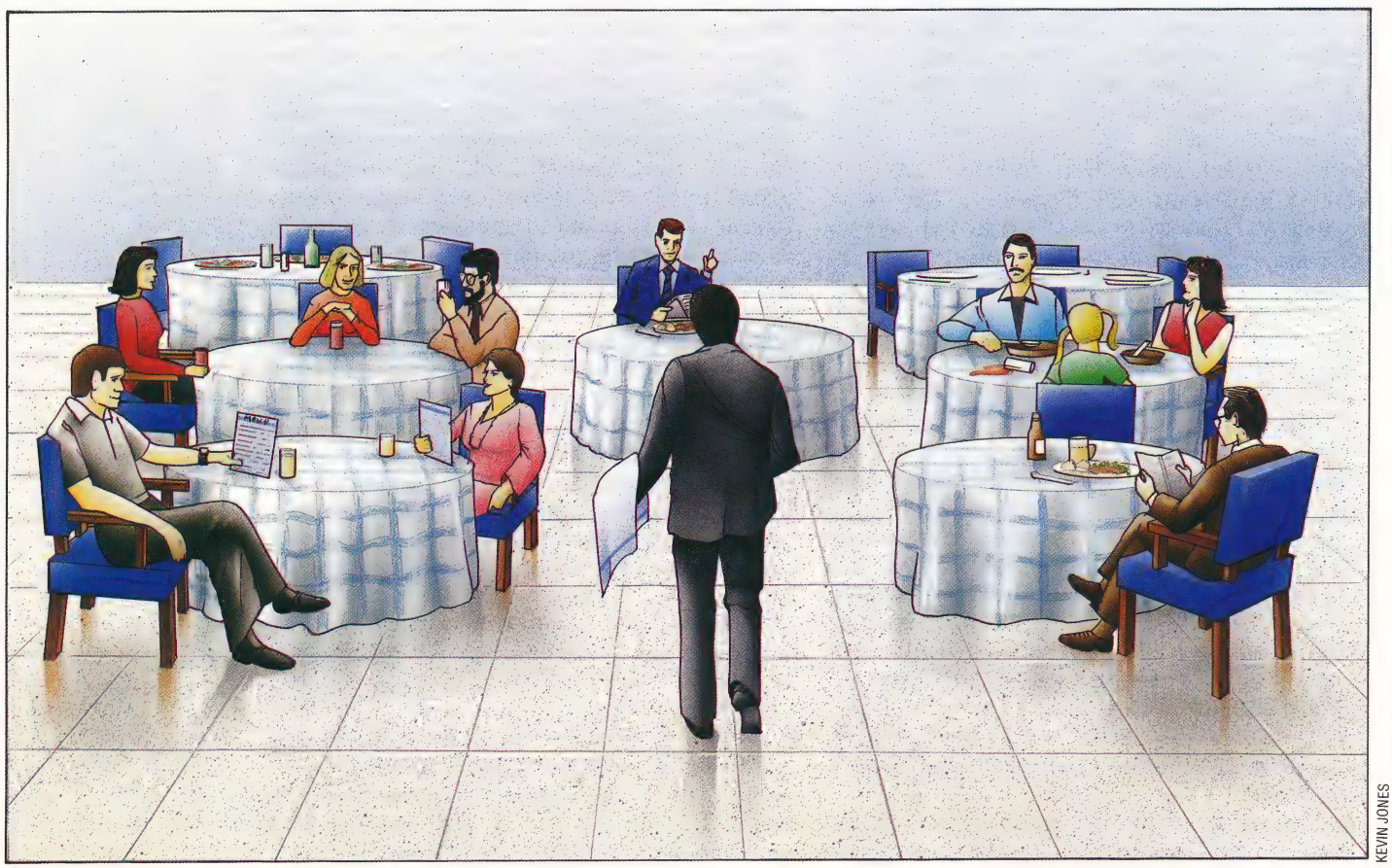
THE IRQV1 VECTOR

When an IRQ is received by the 6502, control is passed to an ISR whose address is held in a vector called IRQV1. This is held at address &204 and &205. The routine entered via this vector checks for interrupts from the hardware that the OS expects to receive and that it needs for its normal functioning. The ISR entered through this vector deals with interrupts from three hardware devices — the Serial Interface chip, the System VIA and part of the User VIA. They take the order of priority in which they are listed.

The Serial Interface chip is the piece of hardware that is responsible for the RS423 interface and the tape interface. The interrupts that come from this source are, obviously, concerned with these interfaces. The chip is the highest priority device that uses the vector.

Next in the 'pecking order' are interrupts generated by the System VIA. This extremely useful device interfaces the CPU to a variety of other devices in the machine, and also provides some timing facilities. A System VIA interrupt is produced when a key is pressed, thus indicating to the OS that the key must be processed in some way. An interrupt is also produced by the chip when a full frame of information is sent to the monitor or television screen. This interrupt is sent once every 50th of a second, and is called the 'Vertical Sync Interrupt'.

The A/D converter also interacts with the System VIA, and an interrupt is sent whenever it has finished converting a signal. This particular interrupt causes the OS to update the values that are used to return a number to the ADVAL function. The other important interrupt provided by the System VIA is generated once every 100th of a second. This causes TIME to be incremented, the 'interval timer' (used in the generation of events) to be updated, the INKEY timer (which times the delay for an INKEY instruction) to be decremented and part of an ENVELOPE to be processed. It also causes the OS to continue the processing of any keys that have been pressed. Other interrupts provided by the System VIA concern the speech synthesiser system and a light pen if fitted. OSBYTE calls exist that enable us to alter the way in which



KEVIN JONES

the OS responds to interrupts from these two sources. However, as we'll soon see, it's rather unwise to mess around with IRQV1.

The lowest priority interrupt dealt with by this vector comes from the User VIA, which is the device that controls interaction with the user port and the printer port.

Any interrupt that cannot be dealt with by this vector is passed on to an Interrupt Service Routine entered via a vector called IRQV2. Most interrupts that cause this vector to be entered come from the User VIA, which can be programmed by the user to provide a range of interrupts to the CPU. However, these interrupts would not normally be recognised and acted upon by the CPU.

THE IRQV2 VECTOR

IRQV2, held at locations &206 and &207, can be altered by the user to point to an ISR provided by the user. You should always use this vector to add your own Interrupt Service Routines, even though it is possible to alter the contents of IRQV1 to point to a new routine. This process of changing the contents of a vector and making it point to a new routine is called *intercepting* the vector. If you wish to intercept the IRQV2 vector in order to add your own interrupt routines, then on entry to the routine pointed to by the vector the following conditions must be in force:

- The CPU must have already stored the contents of the A register in location &FC, and pushed the processor status register (PSR) on to the stack.
- The X and Y registers should be in the state that they were in when the interrupt occurred.

The second point means that you must save the X and Y registers on entry to your ISR by using the following statements:

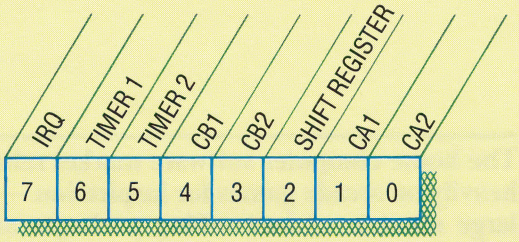
```
TXA:PHA
TYA:PHA
```

Your routine must then clear the interrupt condition. This depends upon the source of the interrupt; if it was the User VIA, this usually involves reading or writing certain User VIA registers. Before returning to the interrupted program, remember to restore the registers and use an RTI instruction to leave the interrupt routine. One final point; although the A register is stored in location &FC on entry to IRQV1, a further interrupt might cause it to be overwritten before we finish processing our user interrupt. There are two solutions: either execute an SEI instruction on entry to your routine and a CLI before leaving it; or push the A register on to the stack on entry to your routine and pull it back before leaving. If you do this, then you must also store the A register value back in &FC before executing the RTI instruction. Users with little knowledge of the User VIA are advised to tread with caution.

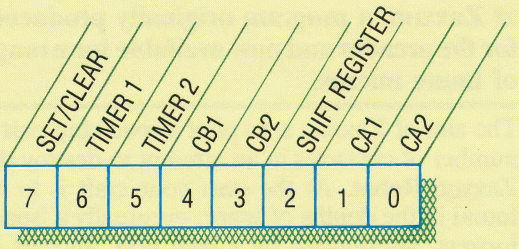
The BBC Micro uses 'event' signals as an alternative to the use of interrupts. These signals — which some regard as 'interrupts without tears' — are the subject or our next instalment.

VIA Chip Interrupts

Interrupt Flag Register



Interrupt Enable Register



LIZ DIXON

Although the BBC Micro's 6502 processor has only two interrupt lines — IRQ and NMI — the peripheral interfacing VIA chips can be programmed so that many devices can share the IRQ interrupt line to the processor. Each VIA chip has two eight-bit I/O ports, each of which has eight data lines and a pair of signalling lines. In port A these are referred to as CA1 and CA2; the corresponding lines in port B are called CB1 and CB2. These lines are often used for handshaking — that is, for synchronising data transfer between the micro and a peripheral device — but they can also be programmed to act as interrupt lines. In addition to four possible interrupt lines coming into each VIA there is an internal interrupt that can be generated by the chip's shift register. This register is used for serial transmission through the CB2 in response to pulses from an external source on CB1. Finally, there are also two 16-bit interval timers that can also generate interrupts. All of these interrupts have to share the single IRQ line to the processor.

When an interrupt is generated, the processor may well wish to know the source of the interrupt. It can do this by interrogating the interrupt flag register in the VIA. Each bit within this register corresponds to one of the possible interrupt sources, and is set by an interrupt request from that source. The exception to this is bit 7, which will be set if any device requires an interrupt. Effectively, bit 7 is an IRQ line.

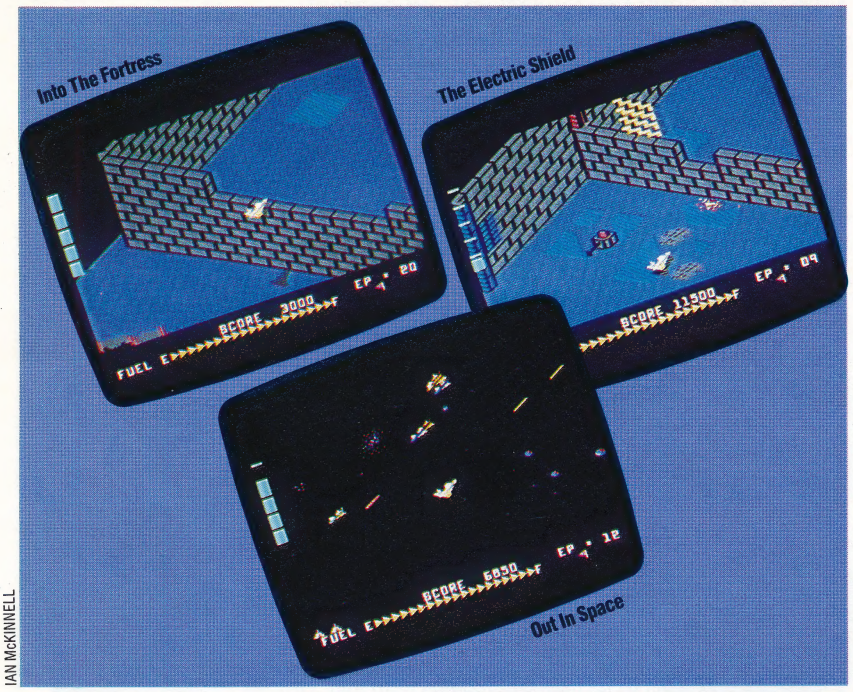
There may be occasions when we might want to stop one device from interrupting while the processor is servicing another. We can do this by programming the Interrupt Enable Register (IER). This register has eight bits that correspond to the same devices as the bits in the Interrupt Flag Register (IFR). If a bit is set to one then interrupts from the corresponding device will be recognised, but if the bit is set to zero then interrupts from the device will be ignored

SPACE FORTRESS

The Final Frontier
Shown here are four different stages of Zaxxon. The player enters the game by flying a spacecraft into the first of the two fortresses. The craft must be manoeuvred at the correct angle and altitude to clear the wall. Within the fortress itself is another wall, which is protected by an electric shield. Extra points can be scored by destroying the gun turrets. Once the first fortress is cleared, the craft moves into space, where it must shoot, or avoid, the swarms of enemy craft that fly past. Finally, after overcoming the defences of the second fortress, the craft confronts the Zaxxon robot itself

The home computer software market relies heavily on arcade games for inspiration — a large number of best-selling packages are adaptations of themes developed on 'dedicated' arcade machines. Here we look at Zaxxon, a program originally produced for the arcades and now available for a range of home micros.

The aim of Zaxxon is to pilot a spaceship past a number of obstacles in an attempt to destroy the Zaxxon Robot. At the start your craft is to be found in the depths of space; eventually a 'space fortress', surrounded by a high wall, appears in front of you. To proceed further, you must pass through a gap in this barrier, before confronting the first of the many obstacles that bar your path.



IAN McKINNELL

These include missiles, radar dishes and fuel dumps — all depicted in realistic 3-D. The fuel dumps are extremely important — destroying these adds to the fuel carried by your spaceship.

The game's graphics are beautifully designed. Three different sprites are used to depict the player's craft — one each for diving, climbing and normal flight. As the ship climbs and dives, these sprites increase and decrease in size in a convincing manner. The obstacles on the ground, too, are well-drawn and the perspective lends an air of realism to the game. A particularly nice touch is the spaceship's shadow; this grows larger as the ship loses height and passes across the objects on the ground.

If your ship is destroyed by enemy fire during this phase of the game, you must return to the start, but after sufficient practice you will find that most of the hazards can be dealt with. It is worth the effort involved in destroying the radar dishes, as these, if left alone, will cause horizontally moving missiles to home in on you. These each require three direct hits before they are destroyed. At all times, great care must be taken to gauge your craft's height correctly — it is all too easy to misjudge this and fly straight into an oncoming missile. Eventually you will be confronted by an electric force field — you should climb high to get over this, and then dive to a low level, where you will be able to obliterate the stationary enemy fighters that are waiting on the ground. It's advisable to get rid of as many of these as possible, as each one demolished at this stage means one less to deal with later in the game.

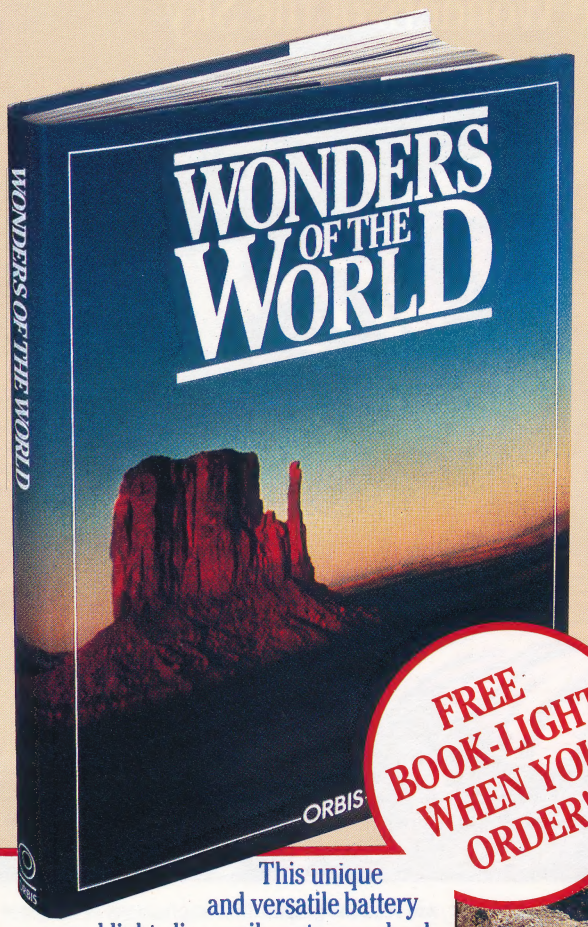
After successfully negotiating the fortress defences, your ship again heads into space, where it is confronted by a swarm of attacking enemy craft. At this point, the player faces the problem of playing a three-dimensional game on a two-dimensional screen. To attack the enemy craft it is necessary to climb or dive to the correct level; this involves trial and error, as the only indication given is the relative sizes of the various ships — when your ship is approximately the same size as the enemy vessels it is in a suitable firing position. A quick trigger finger is a decided asset during this phase of the game!

Leaving space again, your ship returns to the space fortress. This time the gap in the wall is smaller, and the targets are harder to hit. You will now find yourself confronting the Zaxxon Robot, which advances across a grid to meet you. At this point, your ship stops completely and your only defence is to get the robot before it gets you. This requires three direct hits on the automaton's missile launchers. While you try to achieve this, the Zaxxon Robot advances inexorably — if it manages to hit you, you must return to the start of the game.

Zaxxon: For Sinclair Spectrum and Commodore 64
Price: Spectrum cassette, £7.95; Commodore disk, £14.95; Commodore cassette, £9.95
Publishers: US Gold, Unit 10, Parkway Industrial Estate, Hineage St, Birmingham B7 4LY
Author: Peter Adams
Joysticks: Required
Format: Cassette (Spectrum, Commodore); disk (Commodore only)

WONDERS OF THE WORLD

A grand tour across arctic wastes and heat-soaked deserts discovering natural miracles that will astound you.



The Wonders of the Sky

● The Eclipse of the Sun ● The Midnight Sun ● The Polar Dawn ● Amazing Cloud Formations ● The Tornado ● The Rainbow

The Turmoils of the Earth

● Earthquakes ● Mount Etna ● The Giant's Causeway ● The Pools of Keli Mutu ● The Trinidad Pitch Lake ● The Sulphur Pools of Yellowstone

Terrestrial Architecture

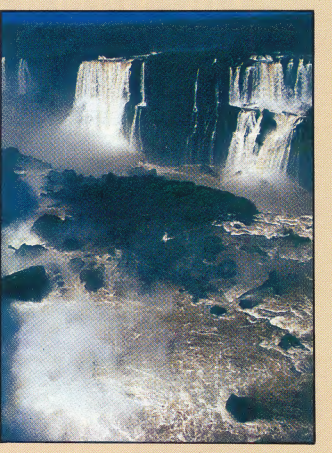
● The Sahara Desert ● The Grand Canyon ● The Stone Towers of Arizona ● The Great Caves ● Death Valley ● The Barringer Meteorite Crater

The Majesty of Eternal Ice

● Icebergs ● Avalanches ● The Polar Caps ● The Great Aletsch Glacier ● The Paradise Ice Caves

The Kingdom of the Seas

● The Dead Sea ● The Maelstrom ● The Great Barrier Reef ● The Amazon ● The Great Lakes ● The Iguaca Falls



▲ *The mighty roar of water as it tumbles 250 feet down South America's Iguaca Falls can be heard up to 15 miles away.*

The Plant Kingdom

● The Venus Flytrap ● Mangrove Swamps ● The Giant Sequoia Tree ● The Fairy Ring Mushroom ● Imitative Orchids ● Plant Defence Systems

The Miracles of Nature

● The Scorpion ● The Sloth ● The Hummingbird ● The Albatross ● The Snow Leopard ● Birds of Paradise ● The Killer Shark ● The Vampire Bat

'Wonders of the World' Dept. Mailwise, FREEPOST, Lambourn Woodlands, Hungerford, BERKS. RG16 7BR
Registered in England No. 1631038.

FREE BOOK-LIGHT WHEN YOU ORDER!

This unique and versatile battery powered light clips easily onto your book, car map, street guide or note pad. Your bed-time book-light is yours to keep absolutely free with our compliments whether or not you decide to keep the book!



▲ *The twin volcanic pools of Keli Mutu – a stunning contrast of brilliant green and sombre red.*



The regal bird of paradise. ▲



To:
'Wonders of the World'
Dept. Mailwise
FREEPOST
Lambourn Woodlands
Hungerford
BERKS. RG16 7BR

YES, please send me *Wonders of the World* on 10 days' approval, together with my free gift. I understand that, if I am not entirely satisfied, I may return this book within 10 days, and keep my free gift in any case.



ORDER FORM

Post in an envelope today. No stamp required.

Please indicate your chosen method of payment.

☐ I enclose my cheque/postal order for £16.95 (£15.00 plus £1.95 p+p) made payable to Mailwise Ltd

☐ Please debit my Access/Visa Card account no.

Expiry date of card _____

Mr/Mrs/Miss/Ms. _____ (Block capitals)

Address _____

_____

_____ Postcode _____

Telephone number _____
(in case we need to contact you for any reason).

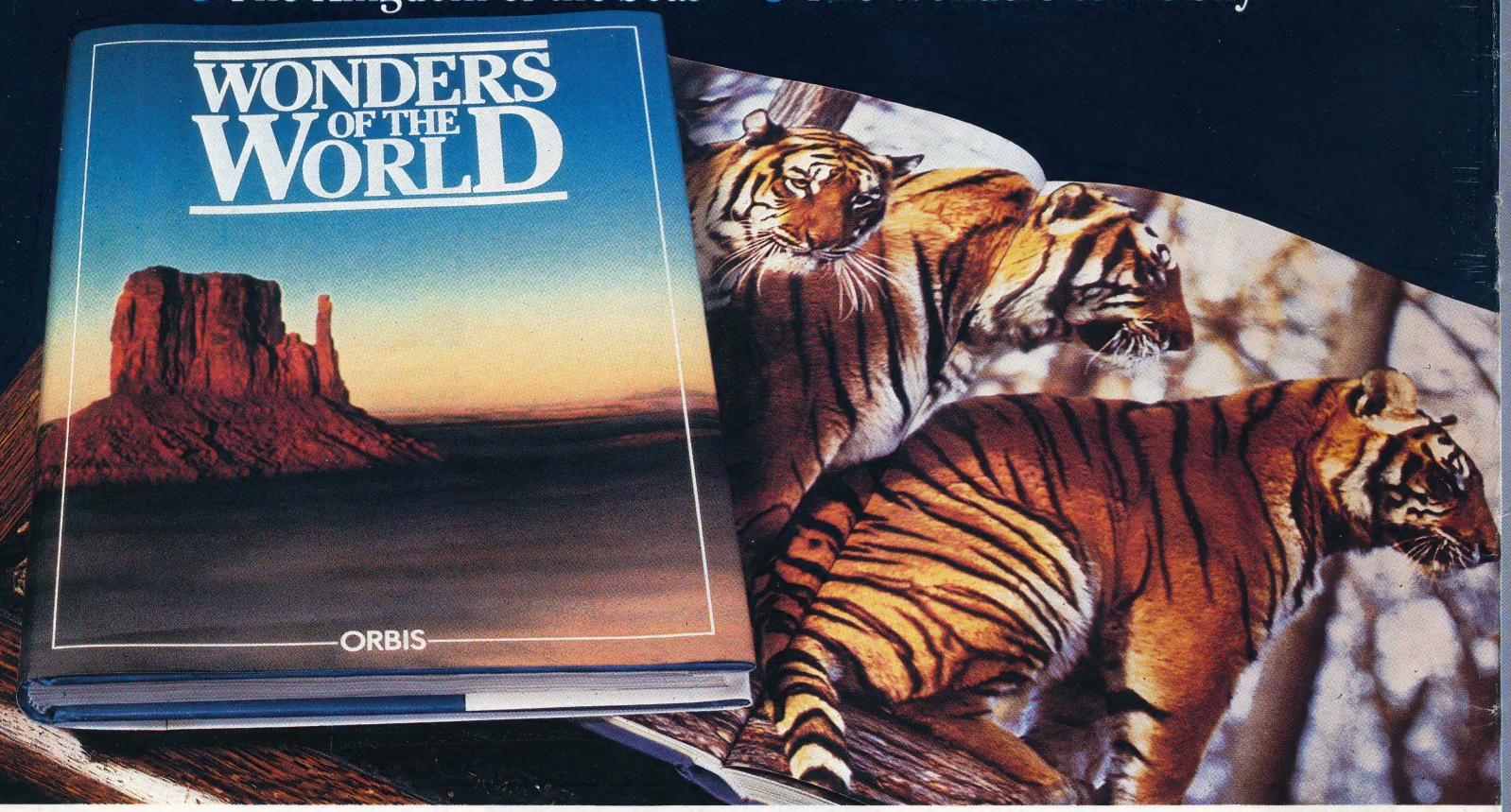
Signature _____
(Parent's or Guardian's signature required if under 18).

HC

Please allow 28 days for delivery. This offer is only open to residents of the UK (if not resident in the UK, please write for further details)

Venture into the world's unspoilt regions and find strange, exotic beauty

- The Miracles of Nature
- The Turmoils of the Earth
- The Kingdom of the Seas
- The Wonders of the Sky



Today, even in the age of space travel, man's technological achievements are still overshadowed by the wonders of the natural world. Throughout our planet, often in inaccessible regions, there are strange phenomena and extraordinary creatures which we are only now beginning to understand.

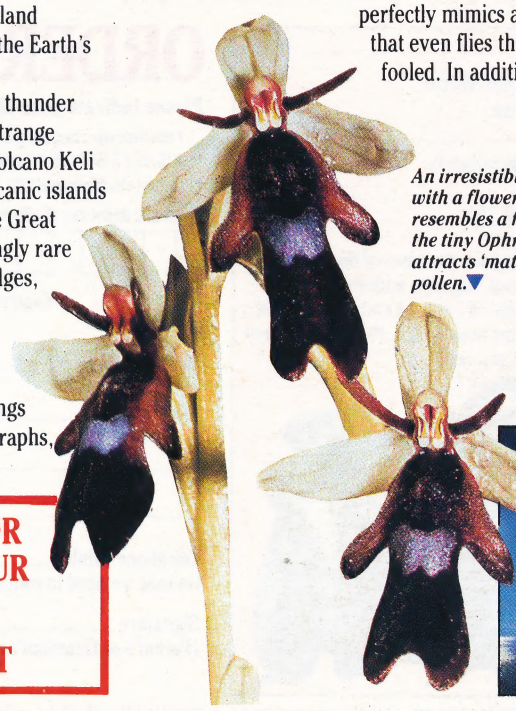
Wonders of the World takes you on a journey of discovery across the world, to desolate wildernesses, tropical jungles, remote island paradises, beneath the sea and beneath the Earth's surface.

Visit the majestic Iguaca Falls whose thunder can be heard 15 miles away . . . and the strange pools, one red, one green, of the giant volcano Keli Mutu, part of the 'ring of fire' of the volcanic islands in the Pacific. Come face to face with the Great White Shark . . . experience (an increasingly rare encounter) the ghost of the mountain ridges, the elusive Snow Leopard.

Nature's intriguing network

This superbly presented volume brings you over 300 magnificent colour photographs, capturing the rarely seen spectacles of Nature. You will debate the scientific explanations, and investigate the origins of these phenomena in the light of the latest theories about our amazing planet.

Breathtaking geological formations, like the mile-deep gash of the Grand Canyon and the towering peak of Everest, are contrasted with miniature miracles like the tiny orchid which so perfectly mimics a fly that even flies themselves are fooled. In addition you will read the graphic first impressions of the great explorers, and the myths and legends which have grown up around these marvels as a consequence of the profound impact they had upon their first discoverers.



An irresistible orchid – with a flower that exactly resembles a female insect – the tiny Ophrys muscifera attracts 'mates' to carry its pollen.▼

A permanent reference work for the whole family

Never before have all these natural wonders been brought together into a single important work. *Wonders of the World* will immediately become a source of wonder for the enlightenment and inspiration of your whole family.

Order your copy by phone or post today.

Simply send your payment in an envelope with the order form overleaf, and *Wonders of the World* will shortly be sent to you together with your free gift!

The Arctic sun stays above the horizon during the summer months, even at midnight.▼



SEE OVER FOR NEWS OF YOUR FREE BOOK-LIGHT