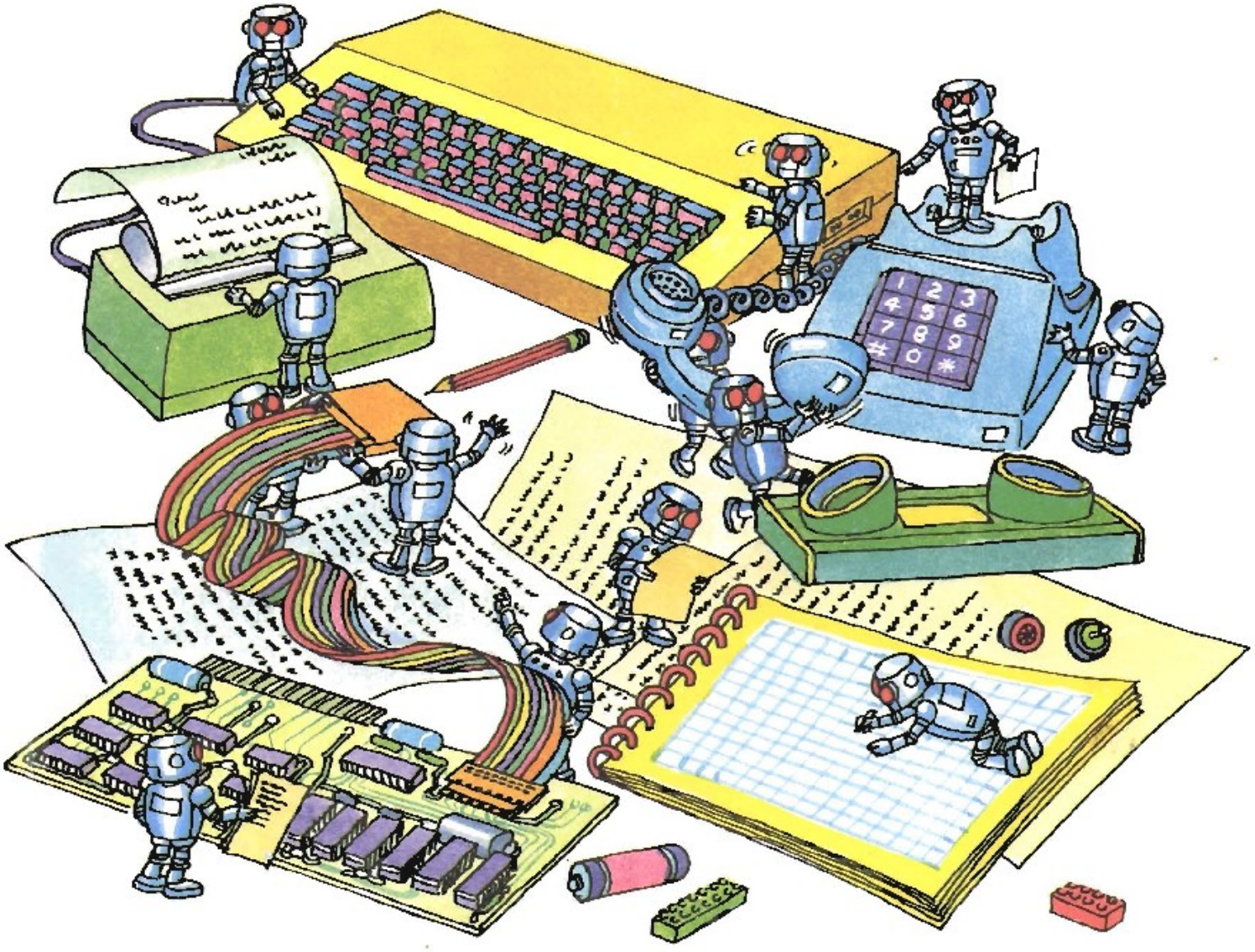
PRACTICAL THINGS TO DO with a MICROCOMPUTER

PROGRAMS AND PROJECTS FOR BEGINNERS

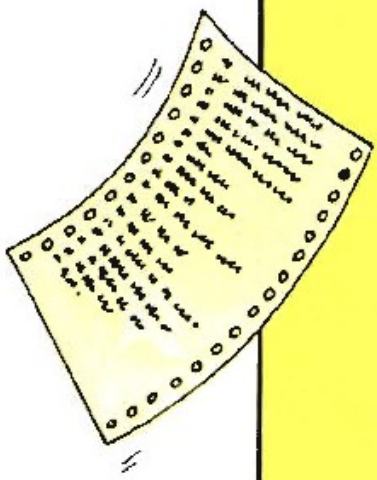
PRACTICAL THINGS TO DO with a MICROCOMPUTER

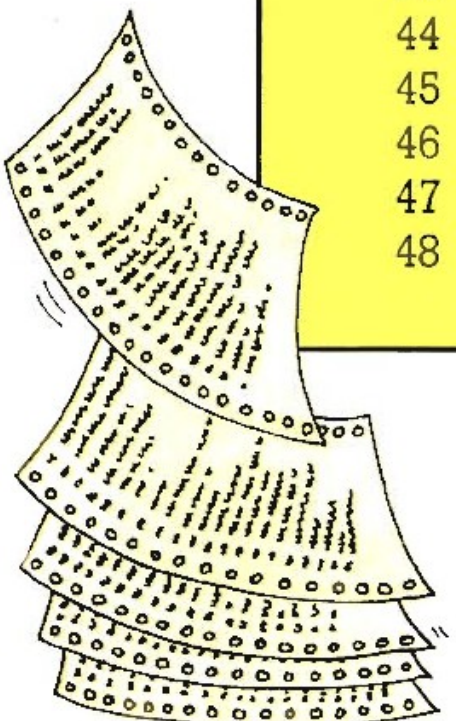
Judy Tatchell and Nick Cutler

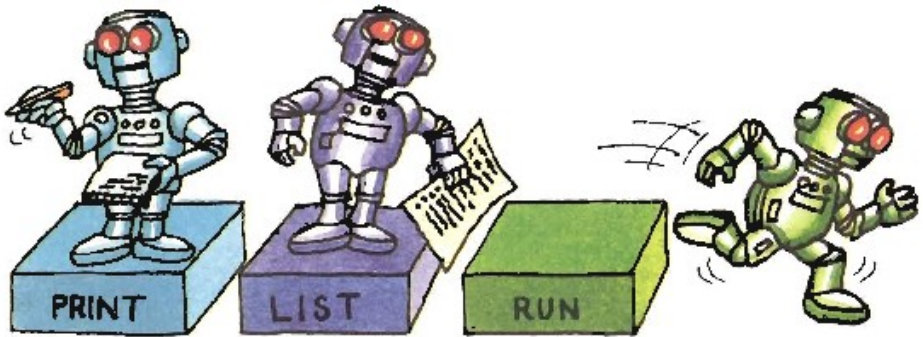


This book was designed by Graham Round and Iain Ashman and illustrated by Graham Round, Mark Longworth, Martin Newton, Graham Smith, Jeremy Gower and Sue Stitt.

Contents





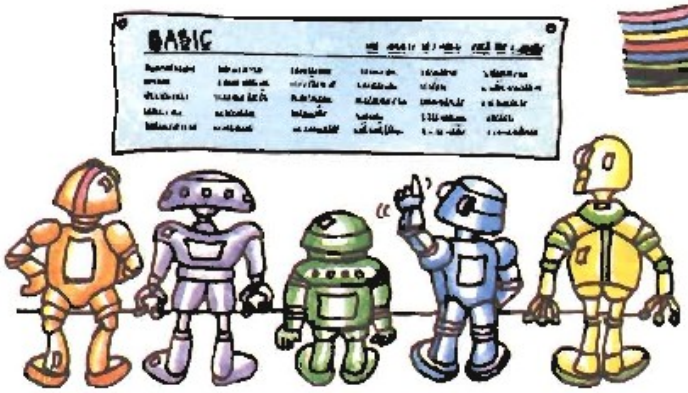


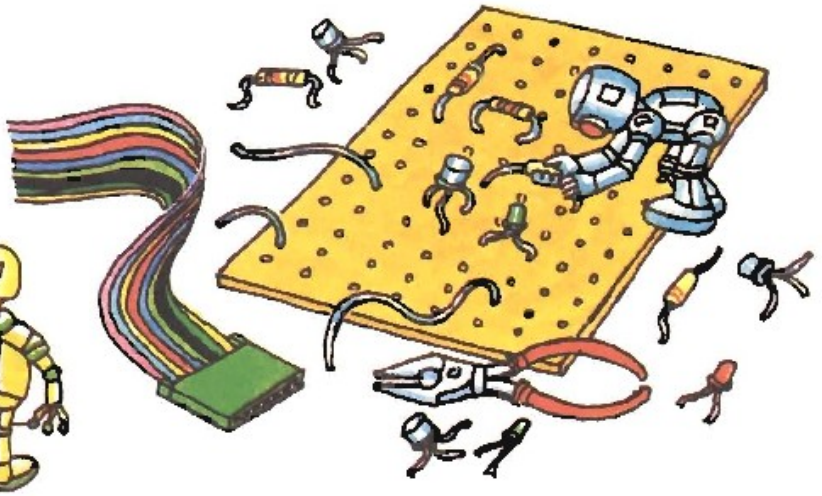
About this book

This book is about all the different things you can do with a microcomputer. There are lots of programs to run, as well as information about extra equipment you can use with your computer. There are also some projects which show you how to build simple electronic circuits to plug into the computer, and even a robot to build.

Some of the programs, such as the averages and sorting programs, make the computer work for you. Others are just for fun. Throughout the book there are lots of ideas for changing and adapting the programs to make them do different things. Short explanations show how the programs work, and typing them in, running and altering them will help you learn how to write your own programs.

You can find out about the different kinds of program you can run on a home computer and where to buy them, as well as about printers, disk drives, light pens and other things you can add on, and ideas for using them. There are hints on how to store programs successfully using a cassette recorder, and a useful program to save in this way which you can use to store a catalogue of information.





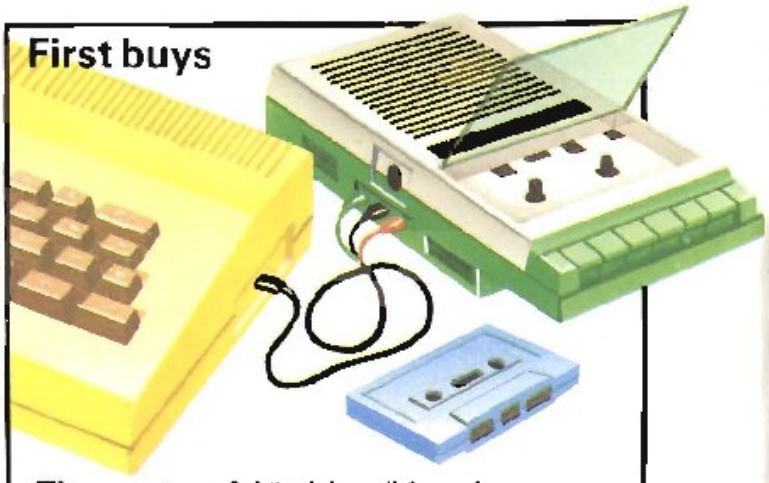
The programs are written in the programming language BASIC, but most home computers use their own version, or dialect, of BASIC. At the end of the book there is a BASIC conversion chart to which you can refer to find the correct command for your computer. The chart covers most of the popular home computers, but if yours is different, you may still be able to run the programs by looking up the commands your computer uses in your manual and substituting them if they differ from those used in the program listings.

For the projects at the back of the book, you need a special socket on the computer called a user port, or input/output port. If your computer does not already have one, there are some suggestions for how to find out if you can get one. Remember that although you cannot damage your computer by typing in the wrong commands in a program, you might damage it if you wire up the circuits in the projects wrongly. You should be all right if you follow the instructions carefully, and there is a list of places to go for help if you do get stuck.

What you can do with a computer

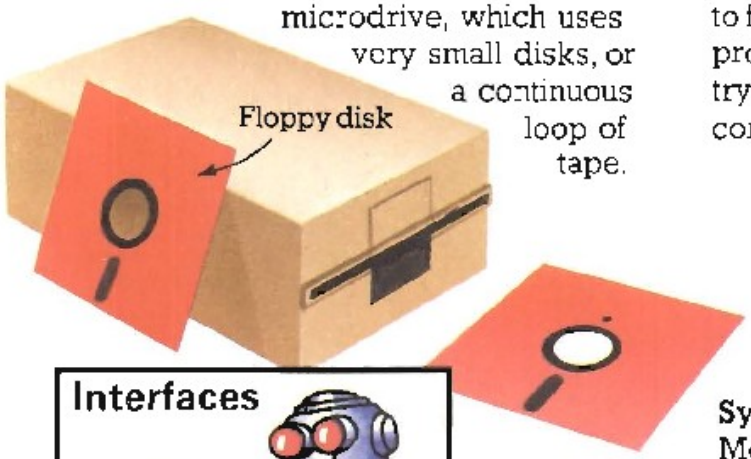
The first thing most people do with a home computer is type in programs from its manual or from magazines. There are lots of other things you can do with a computer, though, and these two pages show some of the extra equipment you can buy to make it do different things. You can find out more about most of it later in the book. Some of the items are not really necessary for home computing, but others, such as a cassette recorder and a printer, are really useful and worth saving up for.

First buys

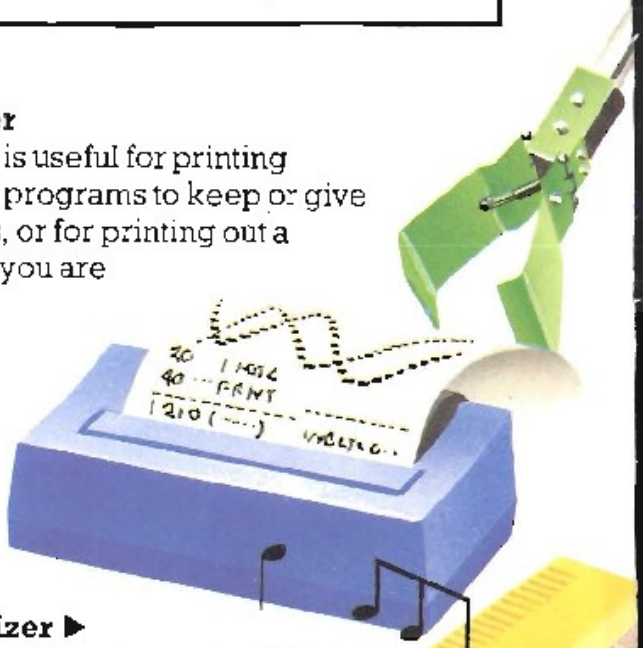


The most useful "add-on" for a home computer is a cassette recorder with tapes for recording, or saving, copies of programs. Most computers work with any cassette recorder, but a few need their own special make.

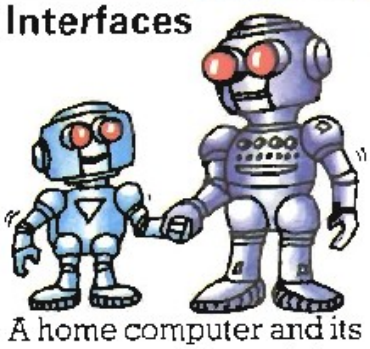
▼ Disk drive
A disk drive stores programs on "floppy disks". It works faster and can store more than a cassette recorder, but is also more expensive. A cheaper version is a microdrive, which uses very small disks, or a continuous loop of tape.



▼ Printer
A printer is useful for printing copies of programs to keep or give to friends, or for printing out a program you are trying to correct.

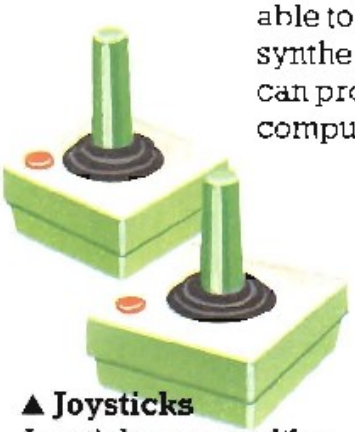


Interfaces

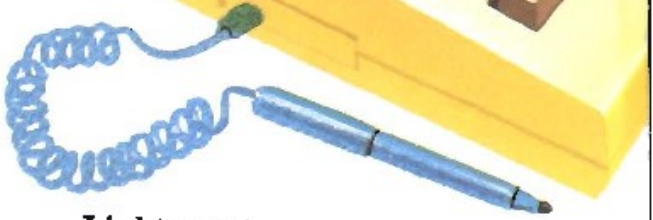


A home computer and its "add-on" need special matching circuits, called interfaces, to regulate signals between them. Most home computers have built-in interfaces for a TV and cassette recorder, but you may need to buy interfaces for other equipment from a computer shop or through advertisements in magazines.

Synthesizer ▶
Most computers have a built-in "sound synthesizer" for making sounds. You may also be able to buy a voice synthesizer, so you can program the computer to talk.

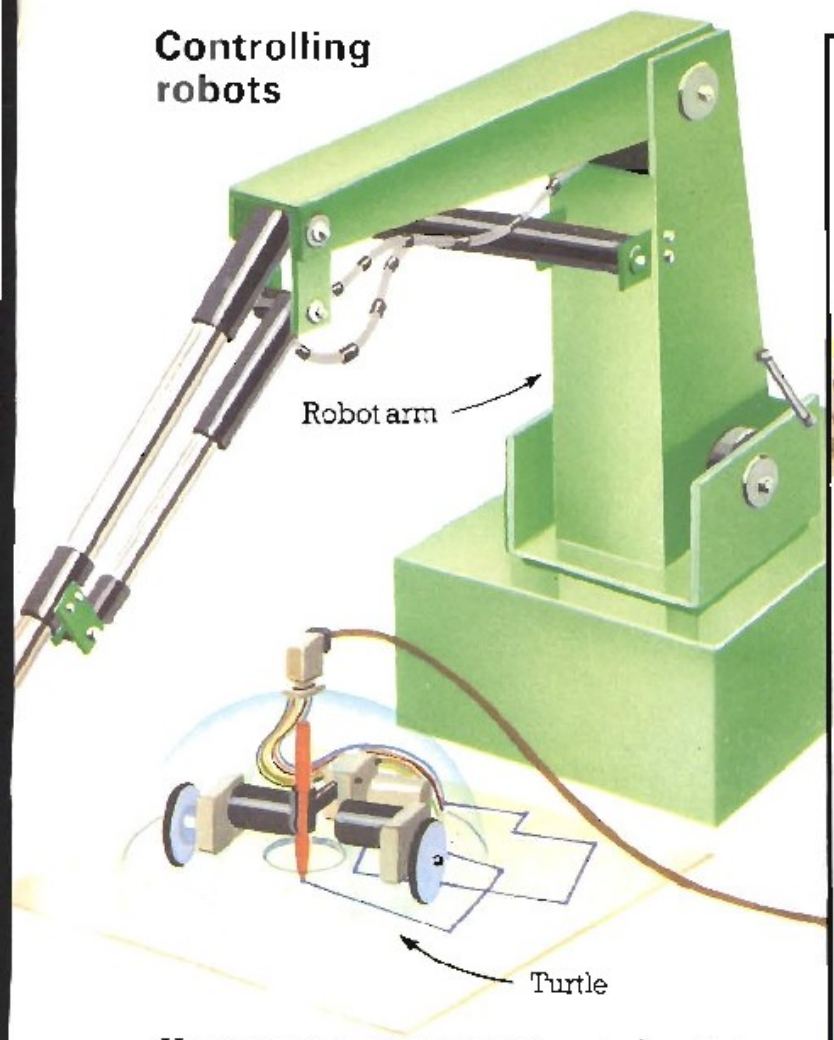


▲ Joysticks
Joysticks are used for playing computer games. Moving the stick tells the computer where to move a shape on the screen, and you fire missiles by pressing the button.



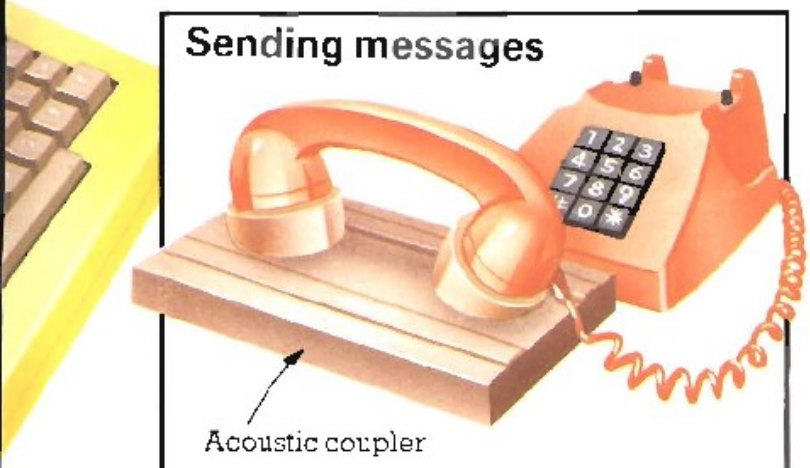
Light pen ▲
You can use a light pen for computer graphics, as you can draw lines directly on the screen with it. There is more about light pens on page 20.

Controlling robots



You can use a computer to control a robot arm which picks things up, or a "turtle" which explores a room or draws pictures.* You plug it into the socket called the user port, or input/output port. If your computer does not have one of these, you might be able to buy one.

Sending messages



You can send programs and information between computers over the telephone lines, using an acoustic coupler, or modem, to connect the computer to the telephone system. You can also receive information from a databank or computer club in this way.

Where to find programs and information

You can buy all kinds of programs on tape in computer shops or by mail order. There are painting and drawing programs, games, business programs and even programs which teach you to write your own programs. Different computers use different versions, or dialects, of BASIC, so make sure the programs you buy are written to run on your computer.

Look in home computer magazines for program listings, and to keep in touch with the latest developments in the computer world. They are full of adverts for programs and equipment you can buy.

A user group is for people who have the same kind of computer and who want to exchange programs, problems and information. Some user groups print newsletters with hints and programming ideas, and they might charge a subscription. Look in computer magazines for details of local user groups.

Running programs

Lots of programs are useful and fun to use just as they are, but you can also adapt and add to them. There is plenty of scope for that in this book. You can find out how to alter a program on the opposite page. Your computer might have an EDIT or COPY key to help you.

The programs in the book are written in a standard BASIC, but your computer might need different commands in some lines. There are conversion charts at the back of the book which list the different commands, and here you can find out how to use the charts.

Converting the programs

Where a program line is marked with a ▲, check in the BASIC conversion chart on page 45 to see if your computer uses a different command to the one in the listing. You will soon get to know which commands you need to change.

If you have a Sinclair Spectrum or ZX81 (Timex 2000 or 1000), you will need to change whole lines or sections in some programs. These programs are marked with a ★, and the lines to substitute are in Sinclair/Timex program alterations on page 46.

Playmates program

This program asks for your name, and then tells you if the computer knows you. Type it in exactly as it is here.*

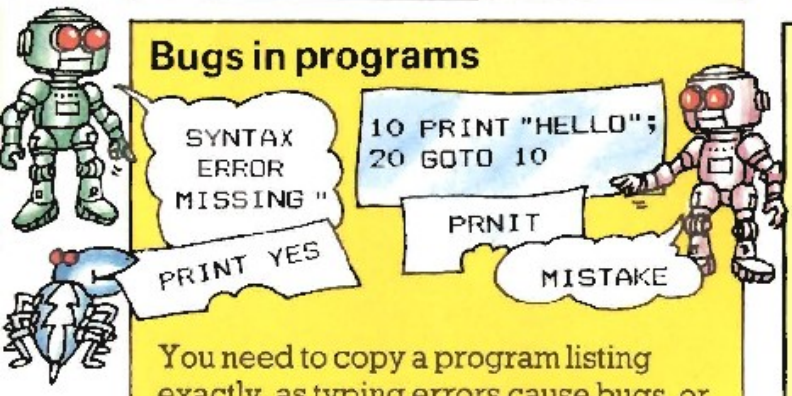
```
★10 DATA "ME","FRED","UNCLE BILL"
 20 DATA "ROBOT","LUCY","FIDO"

 30 DIM N$(6)
 40 FOR I=1 TO 6
 50 READ N$(I)
 60 NEXT I

 70 LET A=0
 80 PRINT "PLEASE TYPE YOUR NAME"
 90 INPUT X$
```

After typing each line, press RETURN (or ENTER or NEWLINE – it varies on different computers). This tells the computer to store the line in its temporary memory (random access memory, or RAM).

```
100 FOR I=1 TO 6
110 IF X$=N$(I) THEN LET A=1
120 NEXT I
130 IF A=0 THEN GOTO 170
140 PRINT "I KNOW YOU ";X$
150 PRINT "I ONLY PLAY WITH PEOPLE
    I KNOW"
160 STOP
170 PRINT "I DON'T PLAY WITH
STRANGERS"
180 STOP
```

Typing LIST will show you the program listing on the screen.

Bugs in programs

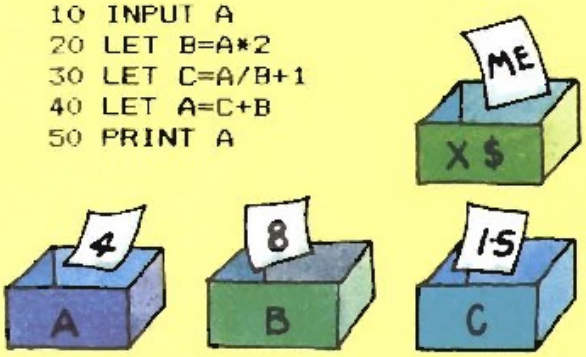


You need to copy a program listing exactly, as typing errors cause bugs, or mistakes, when you run the program. Try the program above, and then alter it to leave out the ; at the end of line 10. Run it again to see the difference.

If you misuse or mis-spell a BASIC word, some computers print the message SYNTAX ERROR or MISTAKE on the screen. Some also tell you where the mistake is, and what sort it is.

Variables

```
10 INPUT A
20 LET B=A*2
30 LET C=A/B+1
40 LET A=C+B
50 PRINT A
```



In this program, A, B and C are variables. They are labels for areas in the computer's memory which can each store a number. Letters are stored as "string variables", with a $ sign, e.g. X$ in the Playmates program. If you want to find out what is happening in a

*The ★ at the beginning of the program is the symbol telling Sinclair (Timex) computer users to refer to the program alterations on page 46.

After typing in all the lines, type RUN, to tell the computer to carry out the instructions in the program.

List of names the computer will recognize.

DIM is short for dimension. It tells the computer to label a space in its memory N$, and make it big enough to hold six items of data. The space N$ is called an array.

This is a FOR-NEXT loop. It makes the computer repeat line 80 six times. Each time, it puts one of the names in lines 10-20 into array N$.

INPUT tells the computer to print a question mark (or other symbol) on the screen and wait for your reply to the question in line 80. Your reply is stored in a memory space, called a variable, labelled X$. (There is more about variables at the bottom of the page.)

This is another loop. Each time the loop is repeated, the computer compares X$ with one of the names stored in array N$. If X$ is the same as one of the names in N$, it sets variable A to 1.

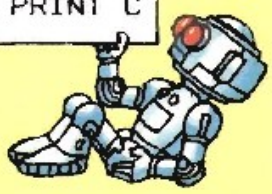
If none of the names is the same as X$, A is still 0, and the computer prints the message in line 170, then stops.

If A = 1, the computer ignores line 130 and prints the message in lines 140-150.

Do not press RETURN until you have typed in a complete program line, even if it runs over to the next line on the screen or in the listing, as in lines 150 and 170.

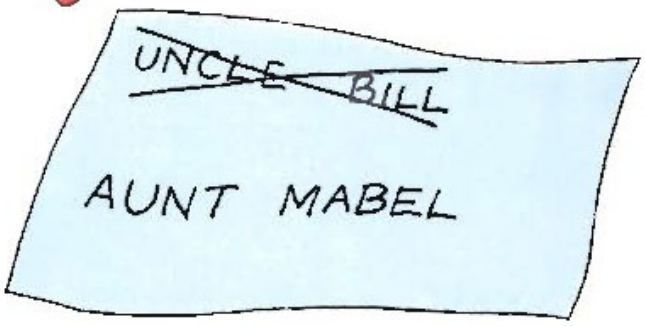
If you add these lines to the program, you can see the values of the variables at each stage. You might want to use this technique in later programs in the book.

```
25 PRINT B
35 PRINT C
```



program, it sometimes helps to tell the computer to print out the values of the variables at different stages in the program. This might even highlight where a program is going wrong, and help you "debug" it, or find its mistakes.

How to alter a program

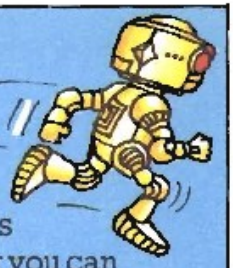


You change a line in a program by retyping the whole line, including the line number. The computer substitutes your new line for the old one with the same number. In the Playmates program, you could retype the DATA lines with six new names, or alter the PRINT lines to make the computer say something else.

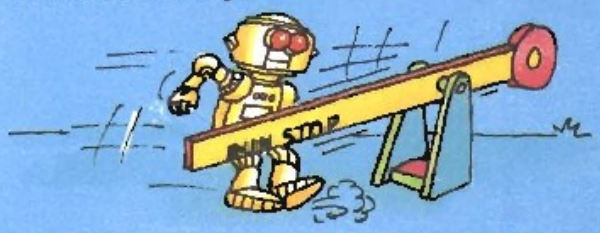
To erase a line, type just the line number and then press RETURN (or ENTER or NEWLINE).

You can add extra lines by typing them in with new line numbers which show the computer where to slot them into the program. Try adding extra DATA lines to the Playmates program. You will also need to change the figure six in lines 30, 40 and 100 to equal your number of names.

Stopping a program



You can run a program as many times as you like. To erase it from the computer's memory, you type NEW, or you can switch the computer off and on again to clear its memory.



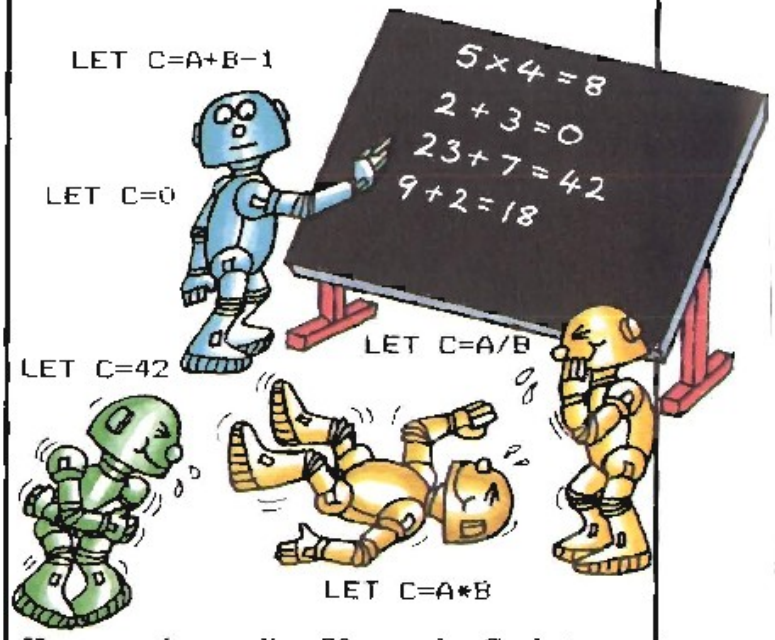
Most computers have a key called ESCAPE, STOP or RUN STOP. This makes the computer stop running a program, but does not erase it from the memory.

Program pitfalls

Here are some programs to run which show a few of the pitfalls you might come across when using other people's programs, and how to avoid them when you write your own. It is often harder to find out what is wrong with a program that runs and does something unexpected than with one that stops and gives you an error message. The program on the right shows how a computer follows its instructions exactly, even if it means doing something silly. It adds two numbers together – but gives the wrong answer every time.

Number dunce

```
10 PRINT "NUMBER PLEASE"
20 INPUT A
30 PRINT "ANOTHER NUMBER PLEASE"
40 INPUT B
50 LET C=A+B+1
60 PRINT A;"+";B;"=";C
```



You can change line 50 to make C whatever you like, to get different wrong answers. Until you explain how it is done, people will think the computer is making terrible mistakes.

Smalltalk

This program is the opposite of Number Dunce – it makes the computer seem quite intelligent. When you run it, the computer appears to be having a conversation with you. However, it only recognizes YES and NO, and the rest of its replies to what you type in are chosen at random, so some conversations are less sensible than others.

```
PLEASE TALK TO ME

?THE WORLD RECORD FOR
TALKING TO A COMPUTER
IS 6 WEEKS 3 DAYS.

IS THAT TRUE?
```

When you run the program, the computer puts a ? or other symbol on the screen and waits for you to type in your reply.

```
10 DATA "I AGREE","YES","NO",
   "SURE"
20 DATA "OK","YEAH!","IF YOU
   SAY SO"
30 DATA "IS THAT TRUE?","SAY
   THAT AGAIN"
40 DATA "HOW INTERESTING","YOU
   RECKON?"
50 DATA "DOES THAT WORRY YOU?",
   "AH....","HUH?"
60 DIM R$(14),R(14)
70 LET W=0
80 FOR I=1 TO 14
90 READ R$(I)
100 LET R(I)=0
110 NEXT I
120 CLS
130 PRINT "SMALL TALK"
140 PRINT
150 PRINT "PLEASE TALK TO ME"
160 PRINT
170 INPUT X$
180 PRINT
190 IF X$="NO" THEN GOTO 340
200 IF X$="YES" THEN GOTO 360
210 LET R=INT(RND(1)*14+1)
220 IF W=14 THEN GOTO 280
230 IF R(R)=1 THEN GOTO 210
240 LET R(R)=1
250 PRINT R$(R)
260 LET W=W+1
270 GOTO 160
280 FOR I=1 TO 14
290 LET R(I)=0
300 NEXT I
310 PRINT "WHY DO YOU SAY ";X$;"?"
320 LET W=0
330 GOTO 160
340 PRINT "WHY NOT ?"
350 GOTO 160
360 PRINT "POSITIVE ?"
370 GOTO 160
```

Debtors and creditors

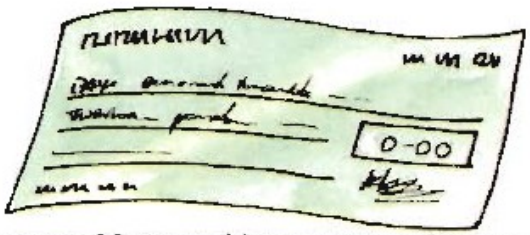
People often complain when computers do stupid things, like sending out bills demanding instant payment of £0.00. This is usually the fault of the program, though. Try running the program below and see what happens if you type in 0 for the amount of money owed. Can you improve the program?*

```
 10 PRINT "DEBTORS AND CREDITORS"
 20 PRINT "PERSON'S NAME PLEASE"
 30 INPUT P$
 40 PRINT "AMOUNT OF MONEY?"
 50 INPUT M
 60 PRINT "IS IT DOLLARS OR POUNDS?"
 70 INPUT M$
 80 PRINT
 90 IF M<0 THEN GOTO 210
100 PRINT "DEAR ";P$
110 PRINT
120 PRINT "YOU OWE ME ";M;" "M$
130 PRINT "AND NO LESS. I WANT"
140 PRINT "IT NOW! PAY UP OR ELSE."
150 PRINT
160 PRINT "YOURS SINCERELY"
170 PRINT
180 PRINT "FREDERICK R. GRUMMET"
190 PRINT
200 GOTO 10
210 LET M=ABS(M)
220 PRINT "DEAREST ";P$
230 PRINT
240 PRINT "I OWE YOU ";M;" "M$
250 PRINT "HO HO HO!"
260 PRINT "YOURS TILL HELL
FREEZES."
270 PRINT
280 PRINT "F.R. GRUMMET"
290 PRINT
300 GOTO 10
```

P$ is a variable to hold name of person who owes you money, or to whom you owe money.

When you run the program, computer waits for you to type in a figure. Type in amount with no £ or $ sign. If you owe money, put a minus sign in front of amount. This tells computer which letter to write.

PRINT on its own will leave a blank line on the screen when you run the program.



Converts M to a positive amount, so letter makes sense.

Bugs to watch out for

Lots of bugs in programs are caused by typing mistakes. Here are some of the most common mistakes to watch out for.

- Punctuation mistakes. Make sure you put quotation marks, semicolons and commas exactly as they are in the program listing.
- Typing I instead of 1, or O instead of Ø.
- Forgetting to press RETURN at the end of each program line.
- Using the same line number twice, so you lose the original line.
- Leaving out a comma between DATA items, or the word DATA at the beginning of a line, or just typing in too many or too few DATA items.

Hints for better programs

You can often improve the instructions in a program by adding extra PRINT lines. For example, the Debtors and creditors program does not tell you to type in the amount in figures. Can you change the program to make it clearer? (Answer on page 47.) The program would not work if you typed the amount in words because variable M in line 50 is a number variable.


```
10 PRINT "PLEASE TYPE IN YOUR AGE"
20 INPUT A
30 PRINT "YOU ARE ";A;" YEARS OLD"
```

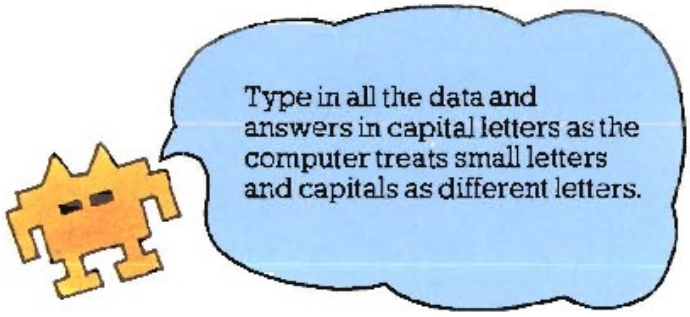
When you run this program, what happens if you type your age in words? Can you make the instructions clearer?

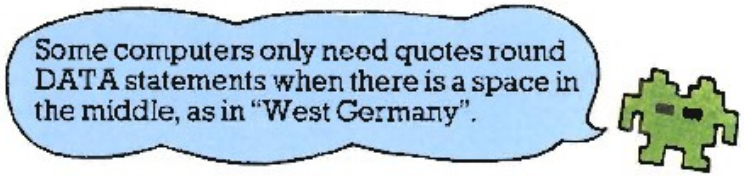
Quizmaster

Here are two quiz programs for you to try out on your friends and family. One is a general knowledge quiz and the other tests your French. You can make up lots of other quizzes based on these programs, and there are some ideas for different subjects on the opposite page.

```
QUIZMASTER
CAPITAL QUIZ

WHAT IS THE CAPITAL OF FRANCE?
?PARIS
CORRECT

WHAT IS THE CAPITAL OF THE USA?
?NEW YORK
WRONG
```

This quiz tests your knowledge of capital cities. The computer prints a question, and puts a ? (or other symbol) on the next line to tell you to type in your answer. It then tells you if you are right or wrong.






```
★10 PRINT
 20 PRINT "QUIZMASTER"
 30 PRINT "CAPITAL QUIZ"
 40 PRINT
 50 LET N=5
 60 LET W=0
 70 LET R=0
 80 PRINT
 90 DATA "FRANCE","PARIS"
100 DATA "THE USA","WASHINGTON"
110 DATA "WEST GERMANY","BONN"
120 DATA "CHINA","PEKING"
130 DATA "SPAIN","MADRID"
140 PRINT
150 PRINT
160 FOR I=1 TO N
170 READ X$,Y$
180 PRINT "WHAT IS THE CAPITAL
OF ";X$;"?"
190 INPUT Z$
200 IF Y$=Z$ THEN LET R=R+1:PRINT "CORRECT"
210 IF Y$<>Z$ THEN LET W=W+1:PRINT "WRONG"
220 PRINT
230 NEXT I
240 PRINT
250 PRINT "YOU WERE ASKED ";N;" QUESTIONS"
260 PRINT "YOU GAVE ";R;" CORRECT ANSWERS"
270 PRINT "YOU GAVE ";W;" WRONG ANSWERS"
```

Variable N stores number of questions to be asked.

Pairs of questions and answers.

Beginning of a loop to make computer repeat lines 160-230 N times (i.e. the number of questions).

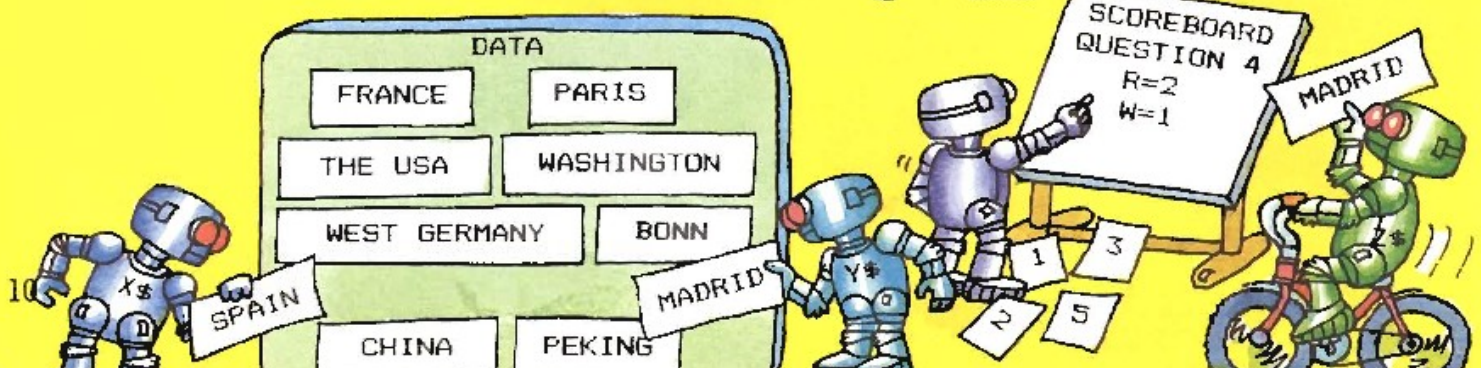
Each time loop is repeated, computer takes one pair of names and stores them in variables X$ and Y$.

Computer stores your answer in variable Z$.

Compares your answer (Z$) with correct answer (Y$). Keeps score and tells you whether you are right or wrong.

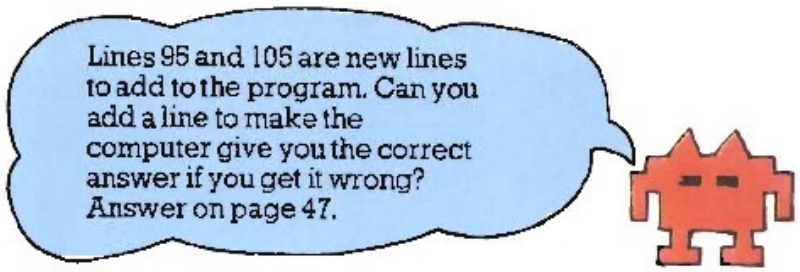
Computer goes back to beginning of loop in line 160, takes next pair of names and asks question.

After repeating loop five times, computer continues with these lines.



French test

You can alter the program to make a different quiz by changing the pairs of words in the DATA lines and asking a different question. For instance, you could change the following lines to make it into a French test.



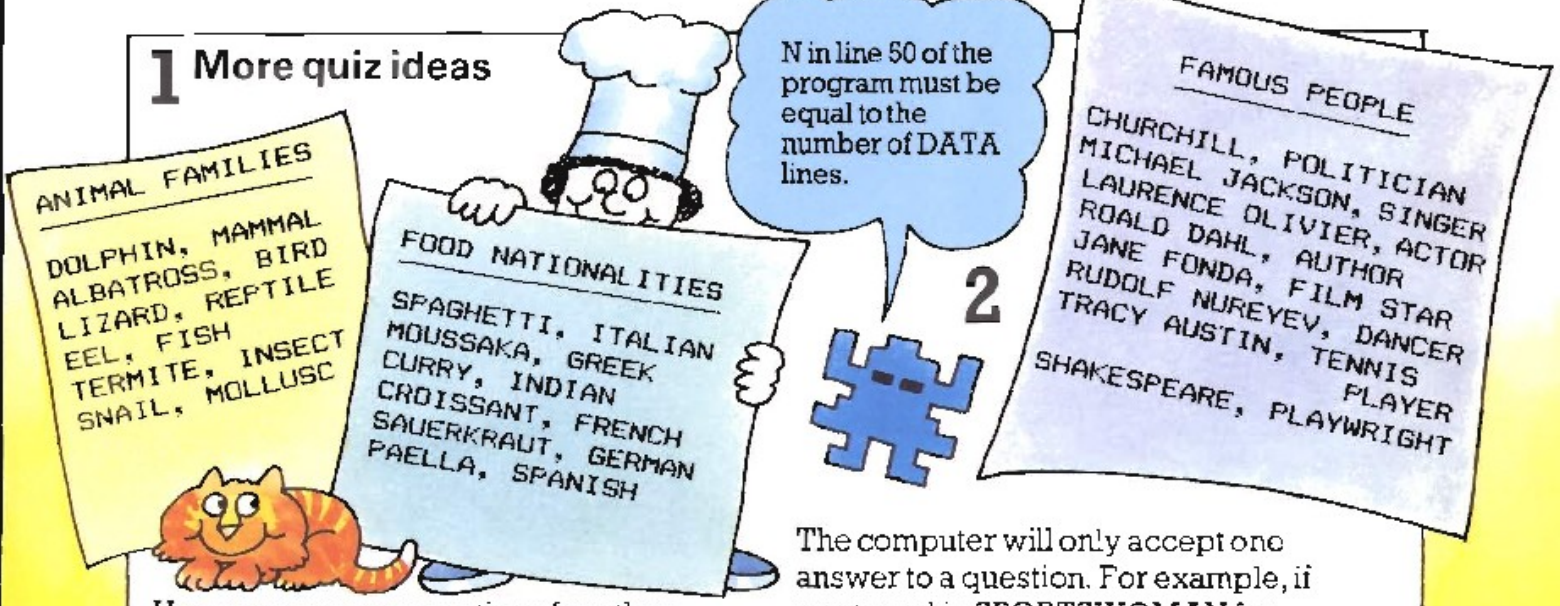

```
 30 PRINT "FRENCH TEST"
 50 LET N=7
 90 DATA "CHIN","LE MENTON"
 95 DATA "SOCK","LA CHAUSETTE"
100 DATA "UMBRELLA","LE PARAPLUIE"
105 DATA "PEPPER","LE POIVRE"
110 DATA "SAILOR","LE MARIN"
120 DATA "LAMP POST","LE REVERBERE"
130 DATA "FOG","LE BROUILLARD"
180 PRINT "WHAT IS THE FRENCH FOR ";X$;"?"
```

There are seven pairs of words in this quiz, so change N to seven.

Alter question and answer pairs.

Change PRINT statement to ask different question.

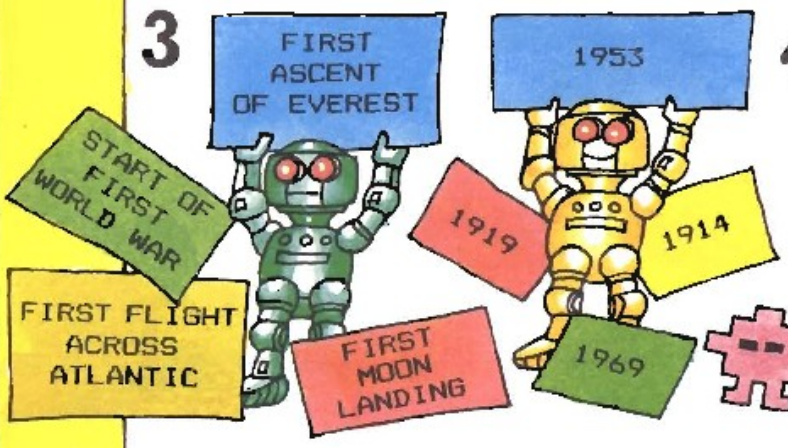
1 More quiz ideas



Here are some suggestions for other quizzes. You can make the quiz as long as you like by adding more DATA lines.

The computer will only accept one answer to a question. For example, if you typed in SPORTSWOMAN for Tracy Austin in the quiz above, the computer would tell you it was wrong.

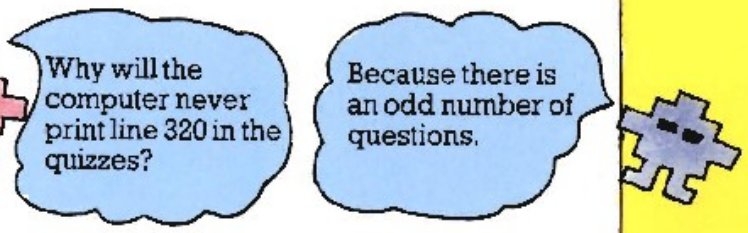
3



You can use numbers in the quiz as well as words, for instance, in a history dates quiz.

4

```
300 IF R=N THEN PRINT "PERFECT
SCORE, BRAINBOX"
310 IF R=0 THEN PRINT "USELESS"
320 IF R=W THEN PRINT "HALF
CORRECT"
330 IF R>W AND R<N THEN PRINT
"QUITE GOOD"
```



Try adding these lines to give people an idea of how well they scored. You can put whatever comments you like inside the quotes.

Wordprocessing and printing

You can buy wordprocessing programs for most home computers. They allow you to use your computer like a typewriter. The words are displayed on the screen, and the program lets you change them and check for spelling mistakes. When you are happy with your text, you can save it on tape or floppy disk, and print it out using a printer.

A printer connected to a computer can make paper copies of any text on the screen. Some can also print graphics. The paper copies are called print-outs, or hard copy.

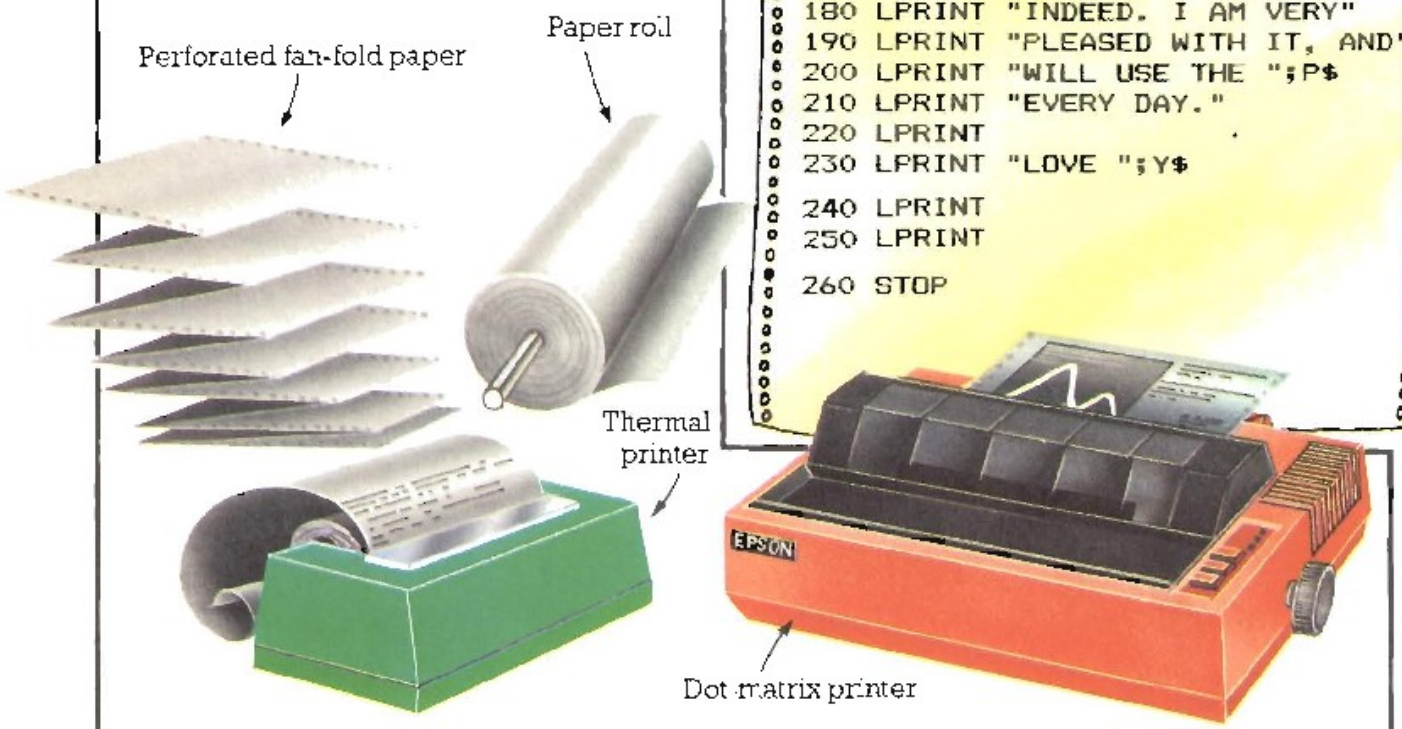
Thank you letter generator

This program could save you from writer's cramp after Christmas or a birthday. It is similar to the wordprocessing programs used in offices to send out standard letters and documents. They are printed out on very good printers, so they look as if they have been individually typed.

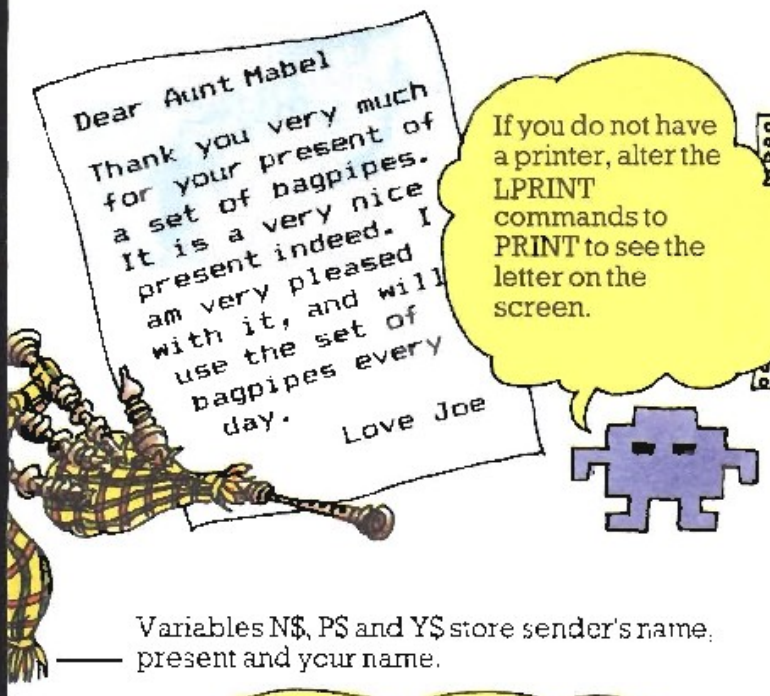
```
10 PRINT "THANK YOU LETTER
GENERATOR"
20 PRINT
30 PRINT "WHO ARE YOU WRITING
TO?"
40 INPUT N$
50 PRINT "WHAT WAS THE PRESENT?"
60 INPUT P$
70 PRINT "WHAT IS YOUR NAME?"
80 INPUT Y$
90 PRINT
▲100 LPRINT
110 LPRINT
120 LPRINT "DEAR ";N$;","
130 LPRINT
140 LPRINT "THANK YOU VERY MUCH"
150 LPRINT "FOR YOUR PRESENT OF"
160 LPRINT "A ";P$;". IT IS A"
170 LPRINT "VERY NICE PRESENT"
180 LPRINT "INDEED. I AM VERY"
190 LPRINT "PLEASED WITH IT, AND"
200 LPRINT "WILL USE THE ";P$
210 LPRINT "EVERY DAY."
220 LPRINT
230 LPRINT "LOVE ";Y$
240 LPRINT
250 LPRINT
260 STOP
```

Printers and print-outs

There are lots of different kinds of printer, and they vary a great deal in price. User groups sometimes have printers which their members can use.



Most printers use paper rolls, or perforated "fan-fold" paper, so you do not have to keep feeding sheets in. The cheapest printers are thermal printers, which use rolls of heat-sensitive paper. They are adequate for program listings, but are not always very clear. Dot-matrix printers give better quality print. Teletypes are slow and noisy, but are reliable, and you can buy them quite cheaply second-hand. Daisy-wheel printers give the best quality print, but are expensive. All printers work more slowly than a computer, so good ones have a "buffer" which stores messages from the computer. The printer prints them out in its own time.



If you do not have a printer, alter the LPRINT commands to PRINT to see the letter on the screen.

Variables N$, P$ and Y$ store sender's name, present and your name.

LPRINT stands for Line Printer.

LPRINT tells computer to print out everything between quote marks.

You can change these lines to write different letters.

Hints

When you are running the program and it asks you what the present was, do not type in "a" or "the" in front of the name of the present, or the letter will look very odd. Also, you cannot put in a plural present, like "paintbrushes". See what happens to the letter if you do either of these things.

Interfacing printers

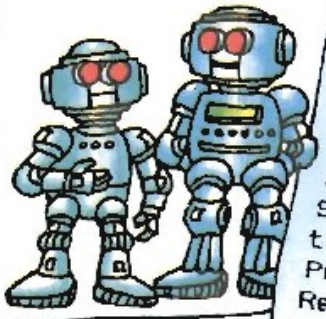
Most computers have a socket, or port, and a built-in interface* for a printer. If not, you can usually buy the necessary interface. There are two standard printer interfaces. One is called a serial interface which uses a simple connecting cable with, usually, three wires. The other is a parallel interface which works faster and uses a flat "ribbon" cable with lots of wires. If you want to buy a printer, make sure you get one that has the same interface as your computer.

Things to try

You can add lines like these to print out your address:

```
▲LPRINT "22 SILLY STREET"
 LPRINT "LITTLEBRAIN"
 LPRINT
```

Can you add lines to the program so that it asks for today's date, and then prints it out on the letter? (You need three extra lines – a PRINT line, an INPUT line and an LPRINT line.)

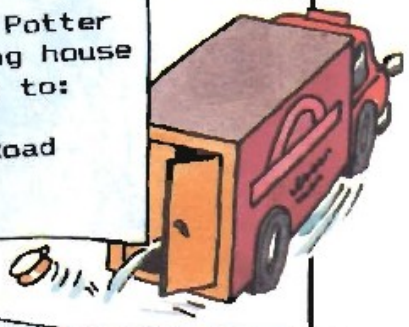


Batley User Group
Dear Member
The next meeting of BUG will be held on Wednesday 4th May at 7.00 at Batley School. The main topic will be Printer Projects. Refreshments provided. See you there!

Change of address

From:
Penny and Peter Potter
We will be moving house on February 21st to:

8 Nightingale Road
Bunford



Party invitation
To: Jane
I'm having a Zombies' and Mummies' party on March 6th at 7.30.
Please come!
From Boris

You can alter the program so you can write different sorts of letters, like those above.

A wordprocessing program was used to help write this book. The author's ideas were typed into the computer, and alterations made on the screen. Then a print-out was made to send to the editor, and a copy kept on disk.

*An interface regulates the signals between a computer and a piece of equipment attached to it.

Finding averages and sorting data

A computer can do calculations much faster than a person can, and can be extremely useful for organizing numbers and other information. The program on this page works out the average, and the biggest and smallest in a list of numbers. The program on the opposite page sorts numbers into numerical order, and could easily be adapted to sort words into alphabetical order.

```
 10 PRINT "AVERAGES PROGRAM"
 20 DIM X(50)

 30 LET N=0
 40 PRINT "YOU CAN TYPE IN UP TO"
 50 PRINT "50 NUMBERS"
 60 PRINT
 70 PRINT "TYPE 999 AT THE END OF"
 80 PRINT "YOUR NUMBER LIST"
 90 PRINT
100 LET N=N+1
110 IF N>50 THEN GOTO 160

120 PRINT "TYPE IN NUMBER ";N
130 INPUT X(N)

140 IF X(N)=999 THEN GOTO 160
150 GOTO 100
160 LET N=N-1
170 PRINT "YOU HAVE ENTERED ";N;" NUMBERS"
180 PRINT

190 LET T=0
200 FOR I=1 TO N
210 LET T=T+X(I)

220 NEXT I
230 PRINT "THE TOTAL IS ";T

240 LET A=T/N
250 PRINT "THE AVERAGE IS ";A

260 LET B=X(1)
270 LET S=X(1)
280 FOR I=2 TO N
290 IF X(I)>B THEN LET B=X(I)
300 IF X(I)<S THEN LET S=X(I)
310 NEXT I
320 PRINT "BIGGEST NUMBER IS ";B
330 PRINT "SMALLEST NUMBER IS ";S

340 PRINT
350 GOTO 30
```

You can enter up to 50 numbers stored in array X.

N keeps count of how many numbers you type in.

999 is a code number to tell computer you have finished typing in numbers. Use a code number which is not in your list of numbers, or computer will stop when you type it in.

Computer will not accept more than 50 numbers.

Computer asks for numbers one at a time.

If you type in 999, computer goes to line 160 and starts calculations.

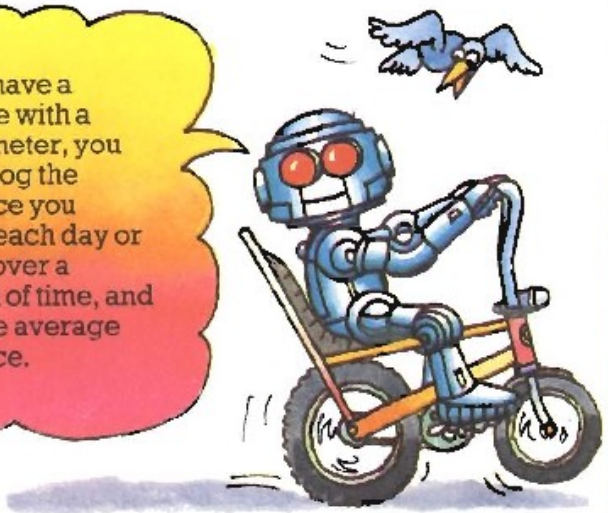
Loop to add numbers together to find total, T.

Computer divides total T by number N to find average, A.

Computer compares all the numbers to find biggest and smallest.

Try finding the average age in your family, or the height and weight of a group of friends, the hours of TV you watch per day, or money you spend a day or week.

If you have a bicycle with a cyclometer, you could log the distance you travel each day or week over a period of time, and find the average distance.



Weather project

If you are interested in meteorology, you could make a chart of the average temperature and rainfall per week or month. You need to keep a thermometer and test-tube marked off in centimetres outside. Take readings every day at the same time, and find the average each

Sorting program

This "bubble sort" program puts up to 20 numbers in order. You can see why it is called a bubble sort at the bottom of the page. It sorts numbers much faster than a person could, but it is slow compared to some other computer sorting programs because it goes through all the numbers to find the biggest, and then through all of them again to find the next biggest, and so on. You can alter the program to sort words or names into alphabetical order by changing variables A and B to string variables A$ and B$, and rewording the PRINT lines. You could use it to make a file of names and phone numbers, or a book collection.

```
10 DIM A(20)
20 PRINT "THIS PROGRAM SORTS UP
TO 20 NUMBERS INTO ORDER"
30 PRINT
40 GOTO 60
50 PRINT "MUST BE BETWEEN 2 AND 20"
60 PRINT "HOW MANY NUMBERS DO YOU WANT
TO SORT?"
70 INPUT N
80 IF N<2 OR N>20 THEN GOTO 50
90 PRINT "ENTER NUMBERS ONE AT A TIME"
100 FOR I=1 TO N
110 PRINT "WHAT IS NUMBER ";I
120 INPUT A(I)
130 NEXT I
140 FOR S=1 TO N-1
150 FOR I=1 TO N-S
160 IF A(I)>A(I+1) THEN GOSUB 250
170 NEXT I
180 NEXT S
190 PRINT "THE SORTED NUMBERS ARE"
200 FOR I=1 TO N
210 PRINT A(I)
220 NEXT I
230 PRINT
240 STOP
250 LET B=A(I)
260 LET A(I)=A(I+1)
270 LET A(I+1)=B
280 RETURN
```

Change number 20 if you want to sort more numbers.

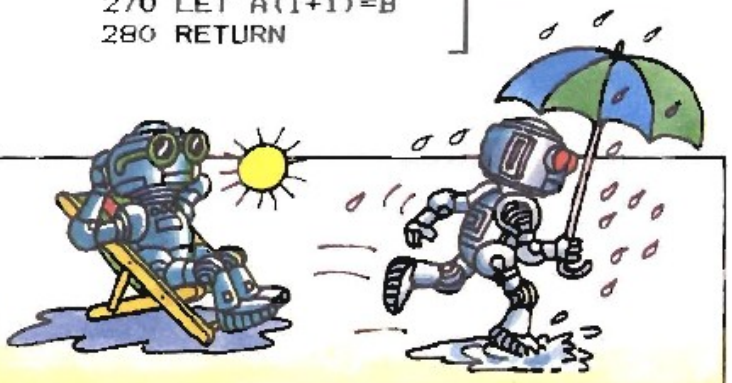
Computer asks how many numbers are in your list and stores your answer in variable N.

Numbers are put into array A in the order you type them in.

Computer compares numbers in list in pairs. If two numbers are in wrong order, it goes to subroutine in line 250 to change them round. Then it compares next pair, and so on, and the biggest number is moved to end of list one place at a time. The whole process is then repeated to find next biggest.

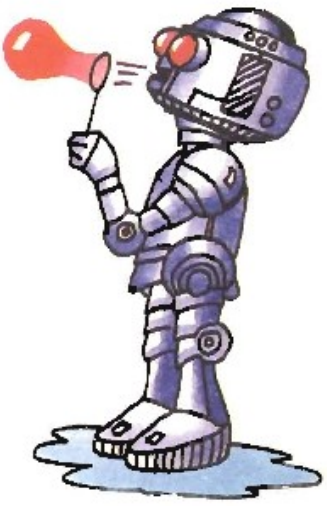
Prints out sorted list.

Subroutine puts two numbers at a time into the right order.



week or month. You could use the sorting program above to make a list of the months in order of temperature and rainfall. If you have a penfriend or relative living a long way away, you could ask them to take similar readings so you can compare them.

This kind of program is called a "bubble sort" because each number "bubbles" through the list until it is in its correct position. The biggest numbers bubble first. Other computer sorting programs use much faster and more complicated sorting methods.



Cryptic codes and sending messages

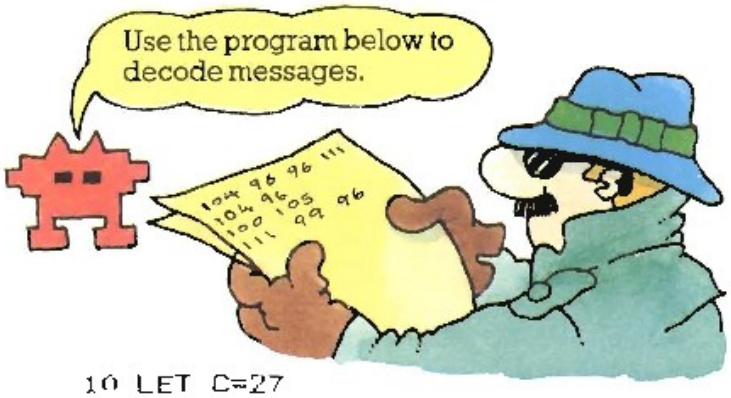
Here are some programs for coding messages with your computer. The programs use the fact that inside the computer, all the characters on the keyboard are converted into numbers. Most computers use the ASCII* code to determine which number represents which character. The BASIC command ASC changes a letter into its corresponding ASCII number, and CHR$ changes an ASCII number back into a character.

You can send coded messages to another computer over the telephone. Find out what equipment you need and what else you can do with it on these pages.

Number codes

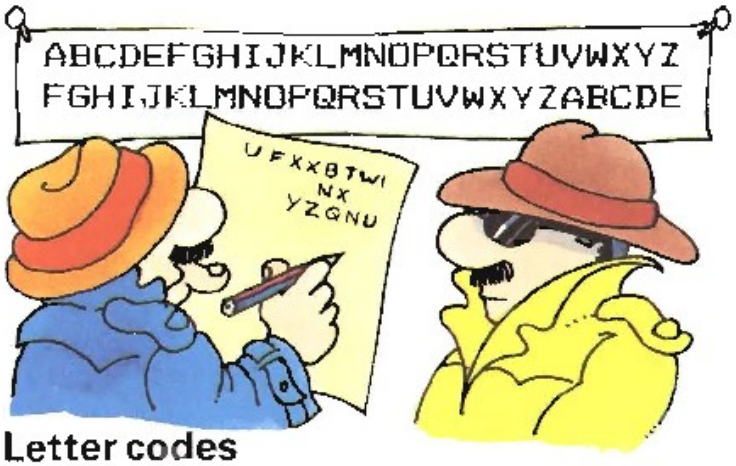
This program converts a message, letter by letter, to ASCII numbers and then adds a "cryptic code number", C, to make the code harder to break.

```
★10 LET C=27
 20 PRINT "TYPE IN FIRST LETTER"
 30 INPUT X$
 40 LET X=ASC(X$)+C
 50 PRINT X
 60 PRINT "TYPE IN NEXT LETTER"
 70 GOTO 30
```




```
10 LET C=27
20 PRINT "TYPE IN THE FIRST NUMBER"
30 INPUT X
40 LET X$=CHR$(X-C)
50 PRINT X$
60 PRINT "TYPE IN THE NEXT NUMBER"
70 GOTO 30
```

A computer was used to decode messages during World War II. It was called Colossus and was the first successful electronic computer.



Letter codes

The spies in the picture are encoding a message by replacing each letter with the fifth letter after it in the alphabet. The chart shows the alphabet with the code letters underneath. This program codes messages in the same way, by finding the ASCII code for each letter, then adding a code number and printing the letter for that ASCII number.

```
★10 LET Z=ASC("Z")
 20 PRINT "CHOOSE A CRYPTIC CODE
 NUMBER"
 30 PRINT "BETWEEN 1 AND 25"
 40 INPUT S
 50 PRINT "TYPE YOUR MESSAGE FOR
 CODING"
 60 INPUT X$
 70 PRINT
 80 FOR I=1 TO LEN(X$)
 90 LET Y$=MID$(X$,I,1)
100 IF Y$<"A" OR Y$>"Z" THEN GOTO 160
110 LET X=ASC(Y$)
120 IF X+S<Z+1 THEN PRINT CHR$(X+S);
130 IF X+S>Z THEN PRINT CHR$(X+S-26);
140 NEXT I
150 STOP
160 PRINT Y$;
170 GOTO 140
```

Computers on the telephone

You can send messages to a friend's computer, provided you both have a modem or acoustic coupler to send the signals along the telephone lines. Modems are often advertised in computer magazines, and they convert computer signals to a series of rapid bleeps. You can send programs, too, as long as the computers understand the same dialect of BASIC.

*American Standard Code for Information Interchange.

Letter decoder

The person receiving your coded message can use this program to decode it without knowing your cryptic code number. It prints out all 26 possible versions of the message, and you should be able to pick out the correct one.

```
★10 DIM Z$(26)
 20 PRINT "TYPE IN THE CODED
 MESSAGE"
 30 INPUT X$
 40 LET L=LEN(X$)
 50 FOR K=1 TO 26
 60 FOR I=1 TO 26
 70 IF I<K THEN LET Z$(I)=
 CHR$(ASC("A")+I-K+26)
 80 IF I>=K THEN LET Z$(I)=
 CHR$(ASC("A")+I-K)
 90 NEXT I
100 FOR J=1 TO L
110 LET A$=MID$(X$,J,1)
120 IF A$=" " THEN PRINT A$;
130 IF A$=" " THEN GOTO 150
140 PRINT Z$(ASC(A$)-ASC("A")+1);
150 NEXT J
160 PRINT
170 NEXT K
```

Try decoding these messages yourself – and then see how quickly the computer can do it using the letter decoder program.

WJZ YKILQPANO NQHA KG
XJSI YMJ LZS YTSNLMY

Here are some extra lines and some replacement lines for the letter decoder. They program the computer to recognize certain words as it goes through the 26 possible versions of the message. (You can put your own three-letter words into lines 180-210.) If it recognizes a word, it prints out the current version of the message, and it should be the correctly decoded one.

```
★15 DIM C$(50)
120 IF A$=" " THEN LET C$(J)=A$
130 IF A$=" " THEN GOTO 150
140 LET C$(J)=Z$(ASC(A$)-ASC("A")+1)
160 FOR J=1 TO L-2
170 LET R$=C$(J)+C$(J+1)+C$(J+2)
180 IF R$="THE" THEN GOTO 270
190 IF R$="AND" THEN GOTO 270
200 IF R$="BUT" THEN GOTO 270
210 IF R$="GUN" THEN GOTO 270
220 NEXT J
230 NEXT K
240 PRINT "CANNOT DECODE"
250 PRINT
260 GOTO 20
270 PRINT "MESSAGE DECODED"
280 FOR J=1 TO L
290 PRINT C$(J);
300 NEXT J
310 PRINT
320 PRINT "READY FOR NEXT MESSAGE"
330 PRINT
340 GOTO 20
```

GENERAL INFORMATION

1. NEWS AND WEATHER
2. SPORTS AND HOBBIES
3. ENTERTAINMENT
4. HOLIDAYS, TRAVEL
5. MARKET PLACE

VIC 20
HOME COMPUTER
CLUB

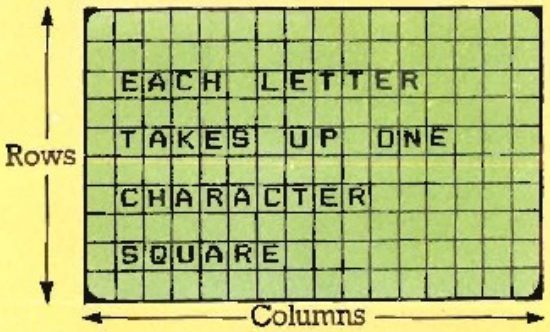
1. APRIL MAGAZINE
2. SOFTWARE
3. MEMBERS' MESSAGE SERVICE

With a modem, you can also link up with a computerized information centre, called a database. You can find out more about these in computer magazines. You dial the database on the telephone and type in an identification number on the computer keyboard. You then have access to all kinds of information, games, computer news, competitions and programs to run on your computer. Some systems allow you to exchange messages with other users of the database or to hire a "page" in the database for, perhaps, a user group bulletin board. You normally have to pay a subscription to the database, and a fee for programs you want to keep.

Designing on the screen

Some computers have graphics characters on some of their keys, which are patterns you can use to build up designs or decorate the screen display. You may even be able to make up your own character shapes to use on the screen. These are called user-defined characters. You can find out more about them below, and also how to position text and other characters anywhere you like on the screen.

Positioning characters on the screen



The screen is divided up into columns and rows, and each letter, number or symbol takes up a space called a character square. You tell the computer in which character square to print by giving it the column and row number of the square. The command varies between computers, but is usually something like PRINT AT or PRINT TAB.

Try writing a program to print out your address in the top right hand corner of the screen.

```
**********
*DESIGNER*
**********
```

```
10 PRINT TAB(15,9);"**********"
20 PRINT TAB(15,10);"*DESIGNER*"
30 PRINT TAB(15,11);"**********"
```

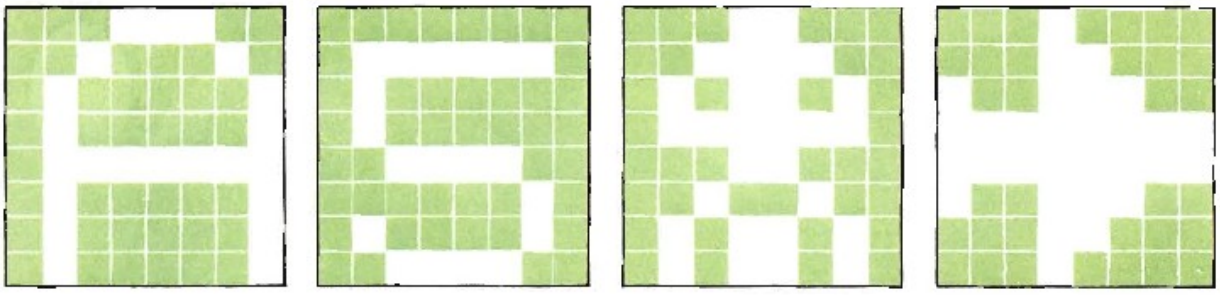
On a computer that uses PRINT TAB, and has a screen 40 columns wide and 20 rows deep, these lines print a border round a heading in the middle of the screen. Try adapting the lines so they work on your computer. (See page 45.)

Graphics characters

The key you press to print graphics characters looks different on different computers. It works like a shift key.

You can use graphics character keys to build up shapes and patterns. For instance, you can make ticks and crosses out of the ■ and ■ keys, and put them in a quiz program to liven up the display. You usually need to press a graphics key to tell the computer to print graphics instead of text characters. Look in your manual to find out how to use graphics characters.

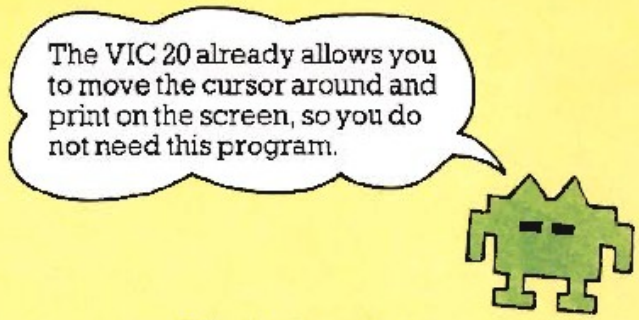
User-defined characters



All text and graphics characters are made by lighting up dots (called pixels) on the screen, usually in an 8 × 8 grid. If you can make user-defined characters on your computer, draw out an 8 × 8 grid (or the right size for your computer's characters) on paper and plan the shape by filling in the dots you want lit up. Then look in your manual to find out how to program the computer to store the shape in its memory and print it out when you press a certain key. Try making the shapes from an arcade game.

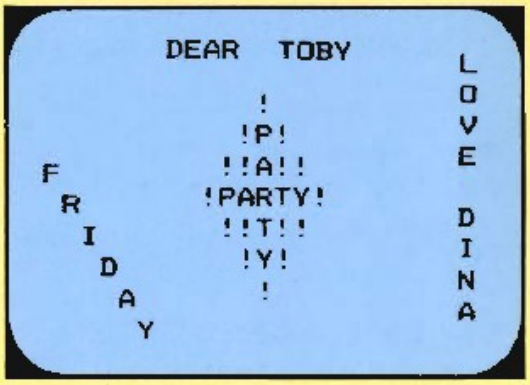
Designer program

This program enables you to position characters wherever you want on the screen. If you have a printer, you could use the program to design invitations or birthday greetings and print them out.

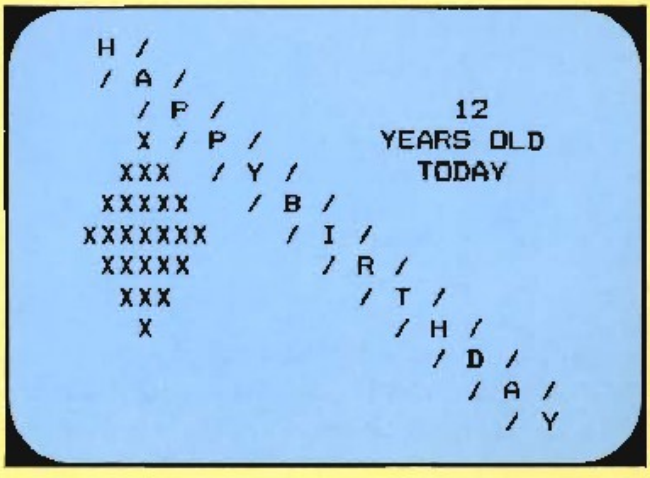



```
★10 PRINT "DESIGNER"
 20 PRINT
 30 PRINT "PRESS THE KEY YOU WANT
 TO USE TO"
 40 PRINT "MOVE THE CURSOR UP"
 50 INPUT U$
 60 PRINT "MOVE THE CURSOR DOWN"
 70 INPUT D$
 80 PRINT "MOVE THE CURSOR RIGHT"
 90 INPUT R$
100 PRINT "MOVE THE CURSOR LEFT"
110 INPUT L$
120 PRINT "WHAT IS YOUR SCREEN WIDTH"
130 INPUT W
140 PRINT "WHAT IS YOUR SCREEN DEPTH"
150 INPUT D

160 LET L=1
170 LET T=1

180 DIM S$(W,D)
190 FOR K=1 TO D

200 FOR I=1 TO W
210 LET S$(I,K)=" "
220 NEXT I
230 NEXT K

▲240 CLS
250 LET C=INT(W/2)
260 LET R=INT(D/2)
270 GOSUB 390

▲280 LET A$=INKEY$(0)
290 IF A$="" THEN GOTO 270

300 IF A$=U$ AND R>T THEN LET R=R-1:GOTO 270
310 IF A$=D$ AND R<D THEN LET R=R+1:GOTO 270
320 IF A$=L$ AND C>L THEN LET C=C-1:GOTO 270
330 IF A$=R$ AND C<W THEN LET C=C+1:GOTO 270
340 IF C=W OR C=L OR R=T OR R=D THEN GOTO 270

350 LET S$(C,R)=A$
360 GOSUB 390
370 LET C=C+1
380 GOTO 280

▲390 PRINT TAB(C,R);"*"
400 FOR Q=1 TO 200
410 NEXT Q
▲420 PRINT TAB(C,R);S$(C,R)
430 FOR Q=1 TO 200
440 NEXT Q
450 RETURN
```

Most keyboards have punctuation keys in the bottom right-hand corner. You could use a block of these as your symbol-moving keys. (Do not use the cursor-control keys.)



When you run the program, the computer asks you to choose your symbol-moving keys and type in the number of columns and rows of character squares on your screen. You can then move the * symbol around the screen to where you want to print a letter.



Computer graphics

You draw pictures and shapes on the screen by telling the computer which groups of dots, or pixels, to light up. The commands to do this vary between computers, and graphics programs written for one may not work on another. It is possible, though, to convert a simple program to work on a different computer. The checklist of graphics commands below might help you convert programs you come across in books or magazines. Another way to draw on the screen is to use special graphics equipment, such as the light pen described on this page.

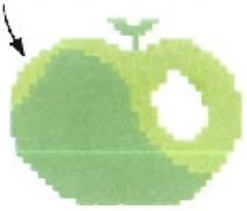
Graphics checklist

Here is a list of the main BASIC graphics commands for different computers, and what they do. If you come across a command in a graphics program that does not work on your computer, check in your manual to see if it uses one of the others listed here.

Plot a point	PLOT, PSET
Draw a line	DRAW, PLOT, LINE
Draw a circle/curve	CIRCLE, SIN, COS
Colour in a shape	PLOT, COLOUR, PAINT FILL
Change background/ foreground colours	GCOL, INK/PAPER, COLOR
Move cursor	PLOT, CURMOV, CURSET, MOVE
Change mode	MODE, HIRES, LORES, PMODE
Draw line of symbols	PATTERN, DRAW, PLOT

Graphics modes and resolution

Small pixels on a high-resolution screen.

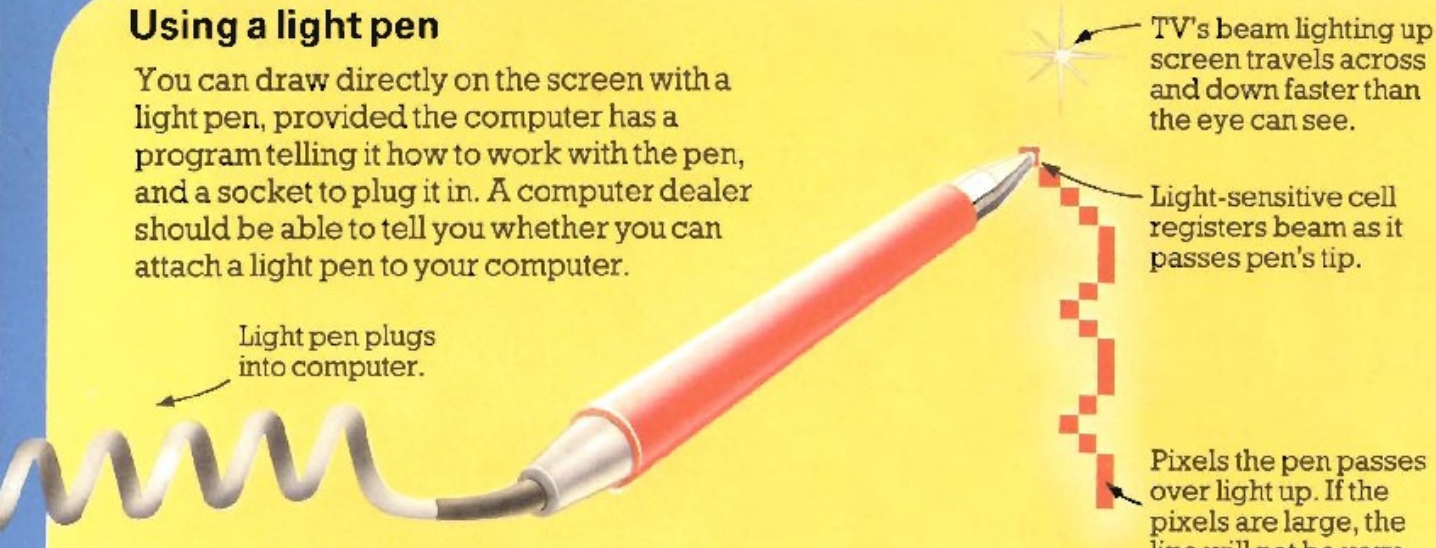


Larger pixels on a low-resolution screen.

Some computers have more than one graphics mode, where the number of pixels you can control on the screen and the number of colours you can use vary. A screen with lots of very small pixels which can make fine lines and details is called a high resolution screen. A low resolution screen has fewer, larger pixels.

Using a light pen

You can draw directly on the screen with a light pen, provided the computer has a program telling it how to work with the pen, and a socket to plug it in. A computer dealer should be able to tell you whether you can attach a light pen to your computer.



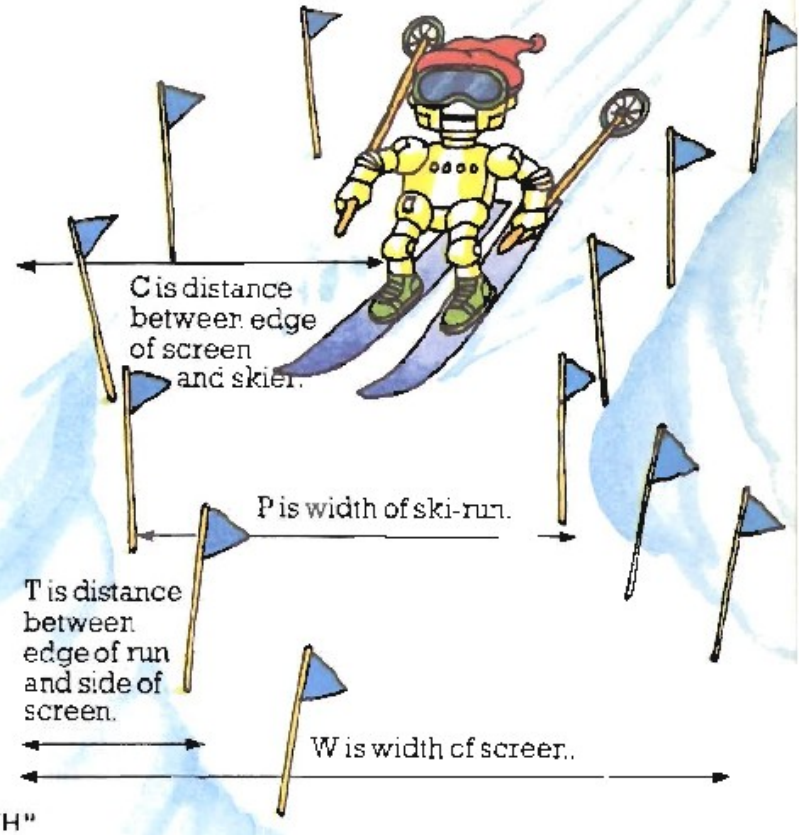
Inside a TV, a tiny beam flashes across and down the screen, lighting it up. The light pen has a light-sensitive cell in its tip which spots the beam as it travels past and sends a signal down the wire to the computer. The computer knows where the TV's beam is at any moment, so when it gets the signal it knows where the light pen is, and can change the colour or brightness of pixels the pen is pointing at.

The light pen is sold with programs to make it do various things. For example, some programs change the colour of the pixels the pen passes over to draw lines. Others tell you the co-ordinates of the pixel the pen is facing, which is useful if you are writing a program to draw a shape on the screen.

Downhill racer game

This game program shows one way in which you can create an impression of movement on the screen, using its upwards "scrolling" motion. When each line on the screen has been printed, the top line disappears and everything moves, or "scrolls", up a line, leaving a blank line at the bottom for more printing.

In the game you have to ski down a slalom ski-run without falling over the precipice.




```
★10 PRINT "WHAT IS YOUR SCREEN WIDTH"
  20 INPUT W
  30 LET P=15
  40 LET R$="."
  50 LET L$=","
▲60 CLS
  70 PRINT
  80 PRINT TAB(INT(W/2-9));
  "<<DOWNHIILL RACER>>"
  90 PRINT
 100 PRINT "YOU ARE SKIING DOWN A SLALOM"
 110 PRINT "RUN. PRESS ";L$;" TO GO"
 120 PRINT "LEFT AND ";R$;" TO GO RIGHT."
 130 PRINT "IF YOU GO OFF THE RUN, YOU"
 140 PRINT "FALL OVER THE PRECIPICE."
 150 PRINT "HIT ANY KEY TO START."
▲160 LET A$=INKEY$(0)
 170 IF A$="" THEN GOTO 160
 180 PRINT
 190 LET T=W/2-P/2
 200 LET C=W/2

 210 PRINT TAB(T);"P";TAB(C);"!!";TAB(T+P);"P"

▲220 LET R=RND(1)
 230 IF R<0.5 AND T>2 THEN LET T=T-1
 240 IF R>0.5 AND T+P<W-2 THEN LET T=T+1
▲250 LET A$=INKEY$(0)
 260 IF A$=L$ THEN LET C=C-1
 270 IF A$=R$ THEN LET C=C+1

 280 IF C<T THEN GOTO 310
 290 IF C>P+T THEN GOTO 310
 300 GOTO 210

 310 FOR T=1 TO 40
 320 PRINT "**!! OVER THE PRECIPICE **!!"
 330 NEXT T

 340 PRINT "HIT ANY KEY FOR ANOTHER GO"
 350 GOTO 160
```

Alter P (width of run) to suit your screen size and skill at game.

R$ stores symbol used to move right, and L$ symbol to move left. Put in different symbols if you want to use different keys.

Prints title in centre of line.

Computer waits for you to hit a key to start.

T is distance between left-hand side of screen and edge of run.

Sets C (skier's position) to centre of run to start.

Prints Ps for sides of run and !! for skier.

Random number decides whether ski-run curves left or right.

Computer moves skier to left or right according to key pressed.

Computer tests to see if skier has fallen over precipice.

Writing games programs

If you want to try writing your own games programs, you might find some of the routines in this one useful, for instance, the lines which print moving shapes, or the score-keeping at the end.

In the game, you are an absent-minded scientist who has been left behind after an exploratory visit to a far planet. Your fleet of spaceships has taken off, and you have time for ten attempts to attract their attention with a laser signal before they set off for home.

```
▲10 CLS
 20 PRINT
 30 PRINT "WHAT IS YOUR SCREEN WIDTH"
 40 INPUT W
 50 PRINT
 60 PRINT "WHAT IS YOUR SCREEN DEPTH"
 70 INPUT D
 80 LET S=0
 90 LET N=0
100 LET Q=0
110 LET RB=0
120 LET CB=0
▲130 CLS
140 LET R=1
150 LET C=0
160 LET C$=STR$(S)
170 GOSUB 610
180 LET C$="-"
190 LET R=D-1
200 FOR C=1 TO W-1
210 GOSUB 610
220 NEXT C
230 LET C$="!"
240 LET R=D-3
250 LET C=W/2
260 GOSUB 610
270 LET C$="+++"
280 LET C=W/2-1
290 LET R=D-2
300 GOSUB 610
310 LET CG=INT(W/2+LEN(C$)-3)
320 LET RG=D-4
330 LET R=1
340 LET C=10
350 LET C$="PRESS ANY KEY TO SIGNAL"
360 GOSUB 610
370 FOR J=1 TO 1000
380 NEXT J
```

You can store your screen width and depth permanently in the program using LET, if you do not want to type them in each time you play the game.

Variable S will keep your score.

N counts your attempts to signal to spaceships.

Variable Q will be 1 if your signal reaches spaceship, and 0 if not.

RB and CB store row and column numbers of spaceship's position.

R and C stand for row and column.

STR$ converts a number variable into a string variable, so your score, S, can be printed out by C$ in line 610.

Draws ground using dashes.

When you run the program, spaceships appear one at a time on the screen, and travel straight across. You press any key to signal. Your laser signal travels upwards, and you must time your signalling so that the beam hits the centre of the spaceship.

Draws laser signal.

CG stores number of column in which laser beam travels up.

RG is row in which laser beam starts.

This is a delay loop. It makes computer count to 1000, giving you time to read the instructions. Some computers are faster than others, so you may need to change the number 1000.

Delay loops are used to slow programs down. A higher figure in the loop makes a longer delay. (You could put one into Downhill Racer on the previous page if it runs too fast.)

```
▲390 LET RB=INT(RND(1)*(D-8)+3)
400 LET G=0
410 LET I=0
420 LET I=1
430 LET R=RB
440 LET C=I
450 LET C$="-O-"
460 GOSUB 610
470 LET C=I-1
480 LET C$=" "
490 GOSUB 610
▲500 LET A$=INKEY$(0)
510 IF A$<>"" THEN LET G=1
520 IF G=1 AND RG>1 THEN LET RG=RG-1
530 LET C=CG
540 LET R=RG
550 LET C$="."
560 GOSUB 610
570 GOSUB 630
580 LET I=I+1
590 IF I>W-5 THEN GOTO 740
600 GOTO 430
▲610 PRINT TAB(C,R);C$
620 RETURN
630 IF RG=RB AND CG=I+1 THEN GOTO 650
640 RETURN
650 LET C=CG-1
660 FOR K=1 TO 20
670 LET C$="-*-"
680 GOSUB 610
690 LET C$="-O-"
700 GOSUB 610
710 NEXT K
720 LET Q=1
730 RETURN
740 IF Q=0 THEN GOTO 760
750 LET S=S+(D-RB)*10
760 LET N=N+1
770 IF N<10 THEN GOTO 100
780 LET C=1
790 LET R=10
800 IF S>300 THEN GOTO 840
810 LET C$="THEY HAVE GONE HOME"
820 GOSUB 610
830 STOP
840 LET C$="THEY ARE PICKING YOU UP"
850 GOSUB 610
860 LET C$="-O-"
870 FOR C=1 TO W-4 STEP 4
880 FOR R=1 TO D-5 STEP 2
890 GOSUB 610
900 NEXT R
910 NEXT C
920 STOP
```

Row along which spaceship travels is selected at random.

G changes to 1 in line 510 if you send a signal.

Lines 420-490 print the spaceship (-O-) moving across the screen. Lines 420-460 print the spaceship in the next column along, and lines 470-490 print a blank in the previous position, so you do not get a line of spaceships printed across the screen.

Waits for you to signal, and sets G to 1 if you do.

Prints laser beam travelling upwards as a column of dots.

Whenever computer prints anything on screen, it goes to this subroutine.

Tests to see if signal has reached spaceship.

Subroutine to print signal reflecting on spaceship.

Sets Q to 1 if your signal reaches spaceship.

Your score for each signal contact depends on height of spaceship. Score is added to total S.

N keeps count of your signals.

If you score over 300, prints out victory message. If you score less than 300 you get stranded. You can make this number bigger to make the game more difficult.

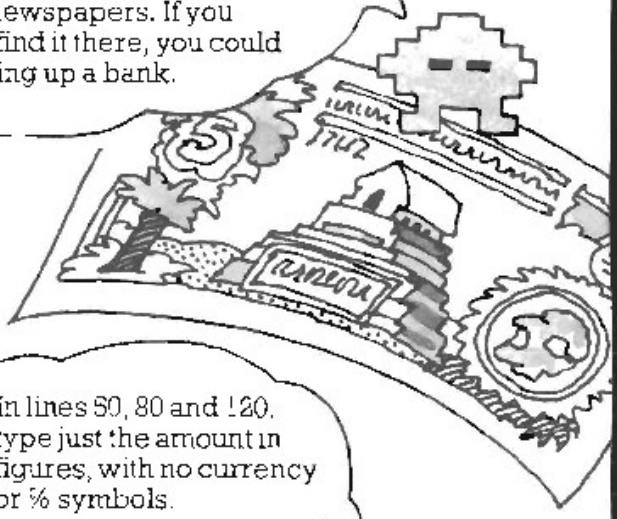
Prints out landing spaceships.

You could invent user-defined characters for the laser signal and spaceships.

Inflation calculator

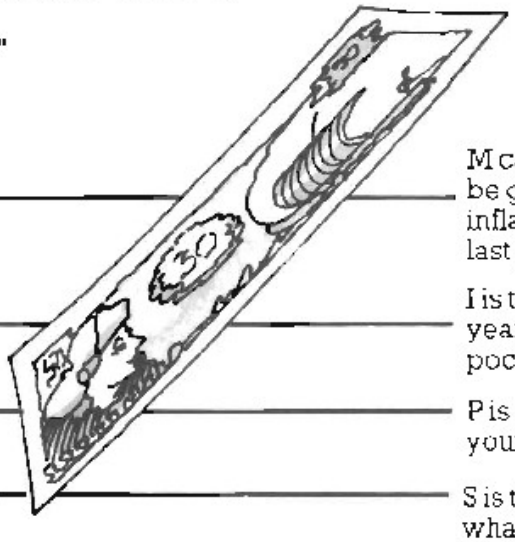
Inflation causes prices to rise so you need more money to buy the same things as you did, say, a year ago. This program works out whether your level of pay or pocket money is keeping up with inflation. It also shows how much a pay rise is worth after inflation has been taken into account.

The current rate of inflation is often mentioned on the news and in newspapers. If you cannot find it there, you could try ringing up a bank.



In lines 60, 80 and 120, type just the amount in figures, with no currency or % symbols.

```
 10 PRINT "INFLATION CALCULATOR"
 20 PRINT
 30 PRINT "HOW MUCH PAY OR POCKET MONEY"
 40 PRINT "DID YOU GET LAST YEAR"
 50 INPUT L
 60 PRINT
 70 PRINT "HOW MUCH DO YOU GET THIS YEAR?"
 80 INPUT Y
 90 PRINT
100 PRINT "WHAT IS THE CURRENT RATE OF
INFLATION"
110 PRINT "(NUMBER ONLY)"
120 INPUT R

130 PRINT
140 LET M=L+(L*R)/100
150 LET I=Y-L
160 LET P=(I/L)*100
170 LET S=M-Y
180 PRINT
190 PRINT "YOUR INCREASE IS ";I
200 PRINT "INCREASE AS A PERCENTAGE: ";P;"%"
210 PRINT "TO KEEP UP WITH INFLATION YOU"
220 PRINT "SHOULD RECEIVE ";M
230 PRINT

240 IF S=0 THEN GOTO 320
250 IF S>0 THEN GOTO 290
260 LET S=ABS(S)
270 PRINT "YOU ARE GETTING ";S;" TOO MUCH"
280 STOP

290 PRINT "YOUR PAY IS ";S;" BEHIND"
300 PRINT "INFLATION"
310 GOTO 280

320 PRINT "YOUR INCREASE IS EXACTLY"
330 PRINT "IN LINE WITH INFLATION"
340 GOTO 280
```



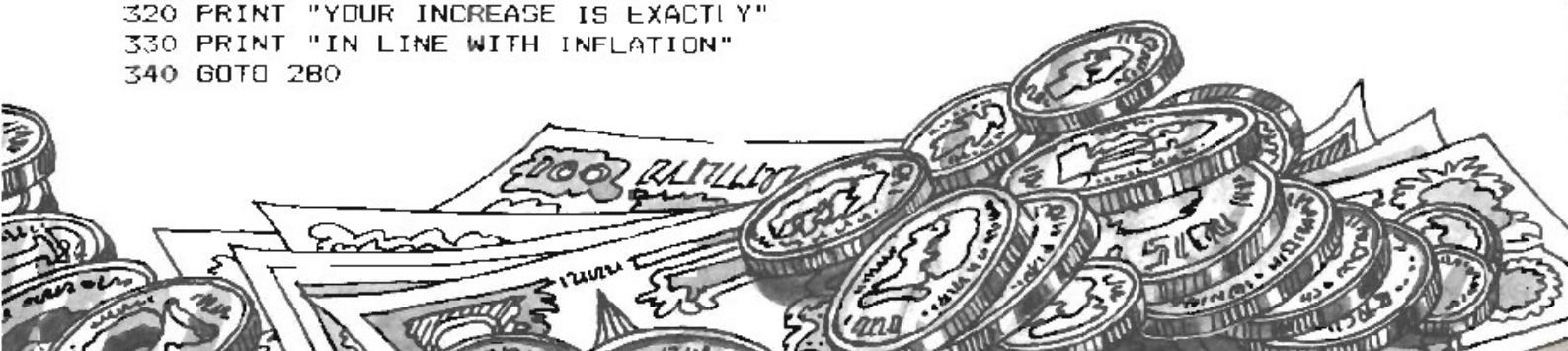
M calculates what you *should* be getting to keep up with inflation by adding R% to your last year's pay.

I is the difference between this year's and last year's pay or pocket money.

P is the percentage increase in your pay.

S is the difference between what your pay rise should be to keep up with inflation and what it actually is. If your rise is above the rate of inflation, S is a minus number.

Converts S to a positive number if necessary.



Graphic inflation

This program uses simple graphics to show the effect of inflation. It draws a chocolate bar, asks you for the rate of inflation and then shows you the amount of chocolate you can buy for the same money after inflation.

```
▲10 CLS
 20 PRINT
 30 PRINT "HERE IS A CHOCOLATE BAR"

 40 LET C$="L"
 50 FOR R=10 TO 14
 60 FOR C=1 TO 20
 70 GOSUB 290
 80 NEXT C
 90 NEXT R

100 LET C=0
110 LET R=2
120 LET C$="WHAT IS THE INFLATION RATE"
130 GOSUB 290

140 PRINT "(NUMBER ONLY)"
150 INPUT X
160 PRINT

170 PRINT "NOW YOU CAN ONLY BUY THIS MUCH"
180 PRINT
190 LET X=100-100/(100+X)*100

200 FOR C=20 TO 1 STEP -1

210 FOR R=10 TO 14
220 IF X>0 THEN LET C$="."
230 IF X<=0 THEN LET C$="L"
240 GOSUB 290
250 LET X=X-1

260 NEXT R
270 NEXT C
280 STOP

▲290 PRINT TAB(C,R);C$
300 RETURN
```

The chocolate bar is printed out as a block of "L" characters. Each L represents 1% of the bar.

R stands for row and C for column. Subroutine in line 300 prints Ls at position C, R for each value of C and R in the loops. For example, first L will be in row 10, column 1 on the screen.

If you are using a VIC 20, add this line:
175 PRINT CHR$(147)

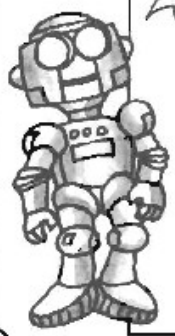
Works out percentage of chocolate bar you will *not* be able to buy after inflation.

For each percent (X) of the bar you will not be able to buy, computer changes an L to a dot.

Subroutine to print symbols on the screen.

Coke bottle program

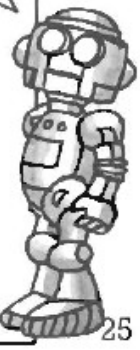
Here are some hints for drawing the coke bottle. If you get stuck, the correct program lines are on page 47.



You could adapt the program to draw a coke bottle that holds less coke after inflation. Draw the bottle as two rectangles, one for the neck and one for the body. You need two loops in the program to draw the rectangles. You can use different symbols, for example, % signs for the body and dots for the neck. Make the body out of 50 (10 × 5) % symbols, so that each one represents 2% of the bottle's contents. After inflation, you need to redraw the body, replacing % symbols with dots to show how the level of coke has gone down.

```
. . .     . . .
. . .     . . .
. . .     . . .
. . .     . . .
. . .     . . .
%%%%%     . . . . .
%%%%%     . . . . .
%%%%%     . . . . .
%%%%%     %%%%%
%%%%%     %%%%%
%%%%%     %%%%%
%%%%%     %%%%%
%%%%%     %%%%%
%%%%%     %%%%%
%%%%%     %%%%%
```

Try these co-ordinates for drawing the neck and body of the coke bottle. You may need to adjust them to suit the dimensions of your screen.
To draw neck:
FOR R = 5 TO 9
FOR C = 18 TO 20
To draw body:
FOR R = 10 TO 19
FOR C = 17 TO 21



Horoscope generator

This program prints out a horoscope when someone types in their birthday. It shows how the computer uses arrays to store lists of information in a particular order, for instance, the birth signs, month numbers and horoscopes.

```
★10 PRINT "HOROSCOPE GENERATOR"
 20 DATA 0,31,59,90,120,151,181,212,
    243,273,304,334
 30 DIM M(12)
 40 FOR I=1 TO 12:READ M(I):NEXT I
 50 DATA "AQUARIUS",20,"PISCES",50,"ARIES",80
 60 DATA "TAURUS",111,"GEMINI",141,"CANCER",172
 70 DATA "LEO",203,"VIRGO",234,"LIBRA",265
 80 DATA "SCORPIO",296,"SAGITTARIUS",326
 90 DATA "CAPRICORN",355
100 DATA "YOU WILL MEET A DARK, MYSTERIOUS
    STRANGER"
110 DATA "YOU WILL BE RICH AND HAPPY WRITING
    PROGRAMS"
120 DATA "YOU WILL LEARN THE MEANING OF LIFE"
130 DATA "YOU WILL BE VERY HAPPY LIVING IN
    HOLLYWOOD"
140 DATA "YOU WILL FALL IN LOVE WITH A GREEN
    COMPUTER"
150 DATA "YOU WOULD BE HAPPIER A LONG WAY AWAY"
160 DATA "LEARN TO LOVE FROGS AND TOADS"
170 DATA "DO NOT RUN FOR BUSES IN THE RAIN"
180 DATA "YOUR CHILDREN WILL NOT BELIEVE IN HORDSCOPES"
190 DATA "ALWAYS TELL THE TRUTH ON THURSDAYS"
200 DATA "YOUR FACE WILL BE YOUR FORTUNE -
    SOMEHOW"
210 DATA "TAKE NO NOTICE OF ANY HOROSCOPE"
220 DIM S$(12),S(12)
230 FOR I=1 TO 12
240 READ S$(I),S(I)
250 NEXT I
260 DIM D$(12)
270 FOR I=1 TO 12
280 READ D$(I)
290 NEXT I
300 PRINT:PRINT:PRINT
310 PRINT "PLEASE TYPE THE NUMBER OF THE MONTH"
320 PRINT "IN WHICH YOU WERE BORN"
330 PRINT "(JAN=1, FEB=2 ETC)"
340 INPUT M
350 IF M<1 OR M>12 THEN PRINT "NO SUCH
    MONTH":GOTO 300
360 PRINT
370 PRINT "PLEASE TYPE IN THE DAY OF THE MONTH"
380 INPUT D
390 IF D<1 OR D>31 THEN PRINT "NO SUCH DAY
    IN ANY MONTH":GOTO 360
400 LET X=M(M)+D
410 LET A=0
420 FOR I=1 TO 11
430 IF (X>=S(I)) AND (X<S(I+1)) THEN LET A=I
440 NEXT I
450 IF A=0 THEN LET A=12
▲460 CLS
470 PRINT:PRINT:PRINT
480 PRINT "YOUR STAR SIGN IS: ";S$(A)
490 PRINT
500 PRINT D$(A)
510 PRINT:PRINT
520 GOTO 300
```

DATA line with number of day in year at which each month starts e.g. March starts on day 90.

Birth signs and number of day in year on which each starts.

Horoscope data for each birth sign.

You can make up new horoscopes or make them longer. Read the horoscopes in newspapers or magazines to get some ideas.

Sets up array for birth signs (S$) and their day numbers (S). DIM tells computer how many pieces of information will be stored in arrays. READ transfers each DATA item into arrays and numbers it.

Sets up array for horoscopes.

Can you add some lines telling the computer what to do if the number of the month or day is typed in words instead of a number?

Computer checks you have typed a number between 1 and 12 for month, and between 1 and 31 for day.

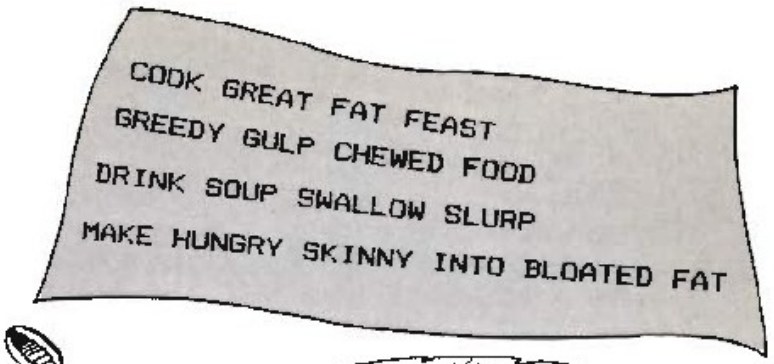
Computer works out day number for your birthday by looking up day number for your month in array M, and adding number you typed in for D.

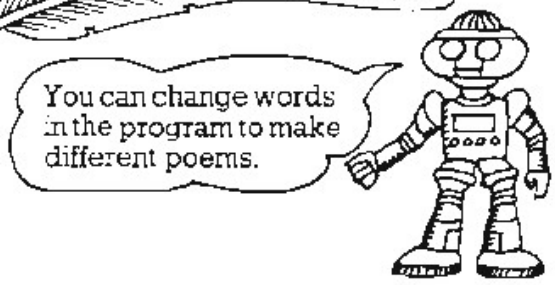
Checks your day number (X) against day number for each birth sign (S) to find your sign.

Prints out birth sign and corresponding horoscope.

Computer poet

You can program the computer to pick words from a list stored in DATA statements, and string them together in lines to look like poetry. The computer chooses words entirely at random, so usually it only turns out one or two good lines amongst a lot of rubbish. Still, you can have fun changing the words in the DATA statements to try and make better poems, or poems on different subjects.






```
★10 PRINT "POEM"
  20 DATA "FOOD","GOOD","MEAT","FEAST",
  "CHEWED","SOUP"
  30 DATA "HUNGRY","GREEDY","BLOATED","FAT",
  "GREAT","THIRSTY"
  40 DATA "EAT","DRINK","THINK","SKINNY",
  "LOOK","CHEW"
  50 DATA "FULL","GULP","HAS","FROM","SLURP",
  "SWALLOW"
  60 DATA "IS","WAS","DOES","MAKE","COOK",
  "INTO"
  70 DIM W$(30)
  80 FOR I=1 TO 30
  90 READ W$(I)
 100 NEXT I
 110 FOR T=1 TO 3
 120 FOR I=1 TO 4
▲130 FOR K=1 TO INT(RND(1)*4+2)
▲140 LET R=INT(RND(1)*30+1)
 150 PRINT W$(R);" ";
 160 NEXT K
 170 PRINT
 180 NEXT I
 190 PRINT
 200 PRINT
 210 NEXT T
 220 STOP
```

30 words for computer to use in poem.

Tells computer there will be 30 words in array W$.

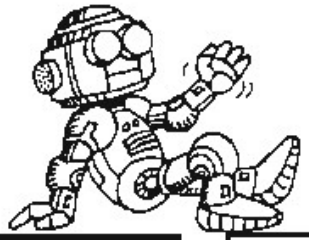
Stores DATA words in array W$.

Loop repeats three times, once for each verse.

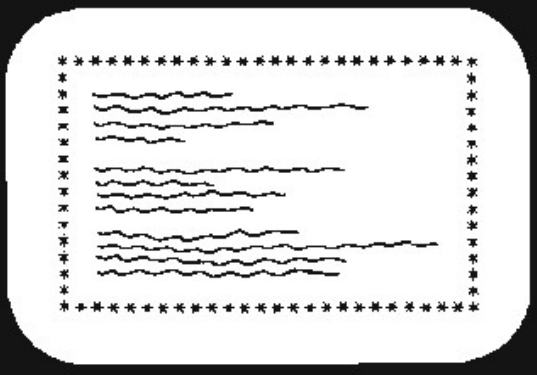
Loop repeats four times, once for each line in a verse.

Computer picks random number, K, between 2 and 6 to choose number of words in a line.

Computer picks random number, R, between 1 and 30, and prints out word in this position in W$.

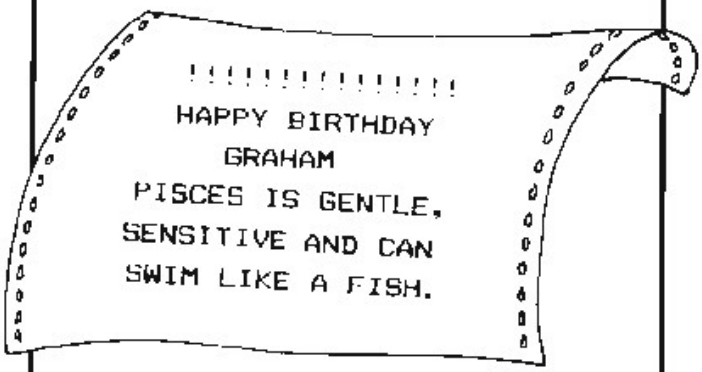


Decorating the screen



You could design more attractive and colourful layouts for both programs on these two pages using the graphics features of your computer. There are hints on designing screen displays on page 18.

Using the programs



If you have a printer you can print out copies of poems and horoscopes – perhaps even to sell at a fund-raising event. You could also give decorated, personal horoscopes instead of birthday cards.

Using tapes and disks

It is a good idea to save programs on cassette tape. Then you can load them into the computer and alter them without having to retype them. Saving programs on tape can be tricky, though, and there are some hints on the opposite page, along with some information about disk drives. The program on this page is a useful one to save on tape or disk. It gives you an organized list of information that is easy to refer to or update. The program listing gives a catalogue of films and the years in which you might have seen them, but you can put different information into the program.

Filmsearch program

You can find the year you saw a particular film by typing in its name and a code number, 1, which tells the computer to search for the year. If you type ALL, and a year, it will list all the films you saw in that year.

```
★10 PRINT "FILMSEARCH"
 20 DATA "SUPERMAN",1979
 21 DATA "THE BLACK HOLE",1979
 22 DATA "STAR WARS",1980
 23 DATA "MOONRAKER",1980
 24 DATA "CHARIOTS OF FIRE",1981
 25 DATA "THE EMPIRE STRIKES
 BACK",1981
 26 DATA "TRON",1982
 27 DATA "BLADE RUNNER",1982
 28 DATA "ET",1983
 29 DATA "OCTOPUSSY",1983
 30 DATA "RETURN OF THE JEDI",1983
100 DATA "END"

110 PRINT:PRINT
120 PRINT "TO FIND THE YEAR WHEN YOU SAW FILM,"
130 PRINT "TYPE NAME OF FILM FOLLOWED BY"
140 PRINT "SEARCH CODE NUMBER 1 (EG. TRON,1)"

150 PRINT
160 PRINT "FOR A LIST OF ALL FILMS YOU SAW"
170 PRINT "IN A YEAR, TYPE ALL, FOLLOWED BY"
180 PRINT "YEAR IN WHICH YOU ARE INTERESTED"
190 PRINT "(EG. ALL,1982)"

200 PRINT
210 PRINT "FOR A COMPLETE LIST OF ALL FILMS"
220 PRINT "AND YEARS, TYPE ALL,1"
230 PRINT
240 INPUT F$,D
250 PRINT:PRINT
260 PRINT "FILM                DATE"
270 PRINT "____                ____"
280 PRINT
290 READ X$
300 IF X$="END" THEN STOP
310 READ Y
320 IF F$="ALL" AND D=1 THEN GOTO 360
330 IF F$=X$ AND D=1 THEN GOTO 360
340 IF F$="ALL" AND D=Y THEN GOTO 360
350 GOTO 290
360 PRINT X$;TAB(25);Y
370 GOTO 290
```

Put the names of films and years when you saw them in these lines. Add more DATA lines for new films.

END is a signal which is used in line 300 to tell computer when to stop running the program. To ask another question, run the program again.

You could adapt this program to make a catalogue of your record collection, with album titles and names of artists instead of dates.

Headings for columns. Leave 20 spaces between words.

Computer checks to see what you want to find out.

Prints out films and years. TAB prints names and dates in line with headings in line 260.

If your computer's BASIC allows the command RESTORE, change line 300 to:
IF X$="END" THEN RESTORE: GOTO 110
This resets the data pointer which tells the computer where to start reading the data, to the first DATA line, i.e. line 20. You can then ask the computer another question without having to run the program again.

Loading and saving on cassette

If you have trouble saving and loading programs on cassette, go through this checklist for possible causes:

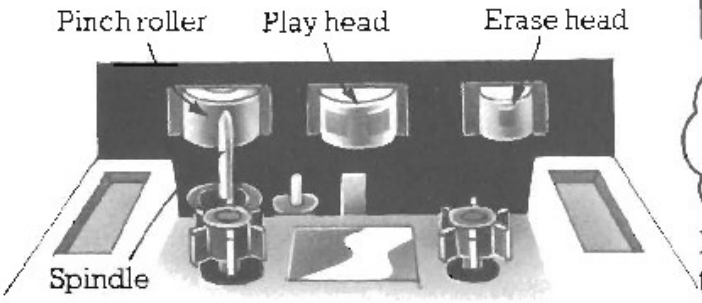
★ Wrong tone and volume setting

Set the tone on high and the volume at three quarters. If this does not work, adjust the volume slightly. The signals may be too quiet for the computer to pick up, or so loud they are distorted. When you have found the correct settings, mark them with a blob of paint.

★ Bald tape

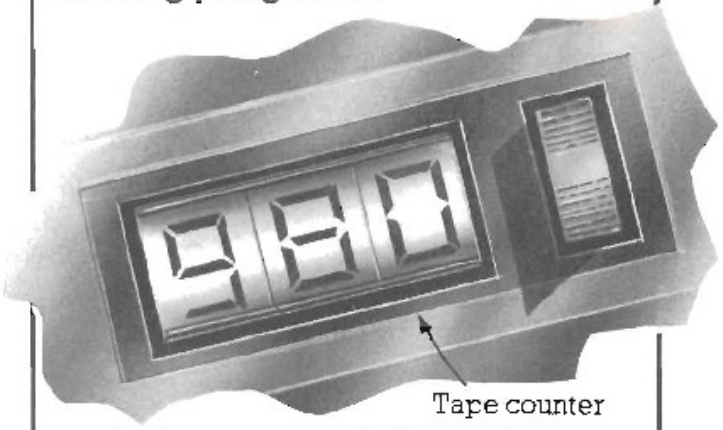
The tape may be worn or damaged. Try a different one.

★ Dirty heads, pinch roller and spindle



The magnetic oxide which coats cassette tape can flake off, especially from poor quality or old tapes. Clean the erase and play heads with a cassette cleaning tape, or a cotton bud dipped in methylated spirits. Also, clean the pinch roller and spindle by pressing PLAY and holding a cotton bud against them as they rotate.

Finding programs

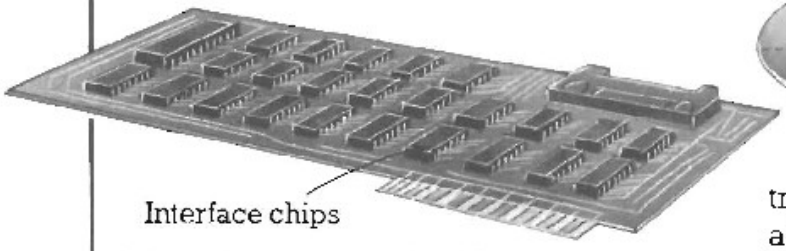


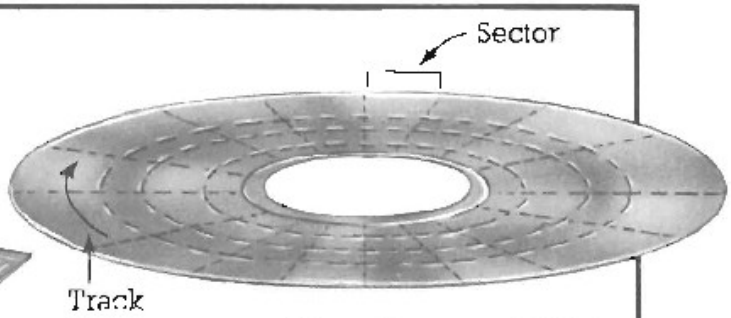
To find programs quickly, use the tape counter on the cassette recorder and keep a note of the number where the program begins on the tape. Use short tapes with only a few programs on each side. For a precaution, save a couple of copies of each program so that if the tape wears out or gets damaged, you do not lose the program.

Some computers list block headers and lengths on the screen while saving and loading. Others show different patterns or lines and colours for different blocks.

Most computers record programs on to tape in "blocks", which are sections rather like the chapters in a book. The computer gives each block a label, or "header". This is usually in hexadecimal, a number code which uses the digits 0 to 9 and letters A to E. If you keep getting BLOCK? or HEADER? error messages while loading or saving, and have tried the suggestions in the checklist opposite, get advice from a dealer.

Disks and disk drives





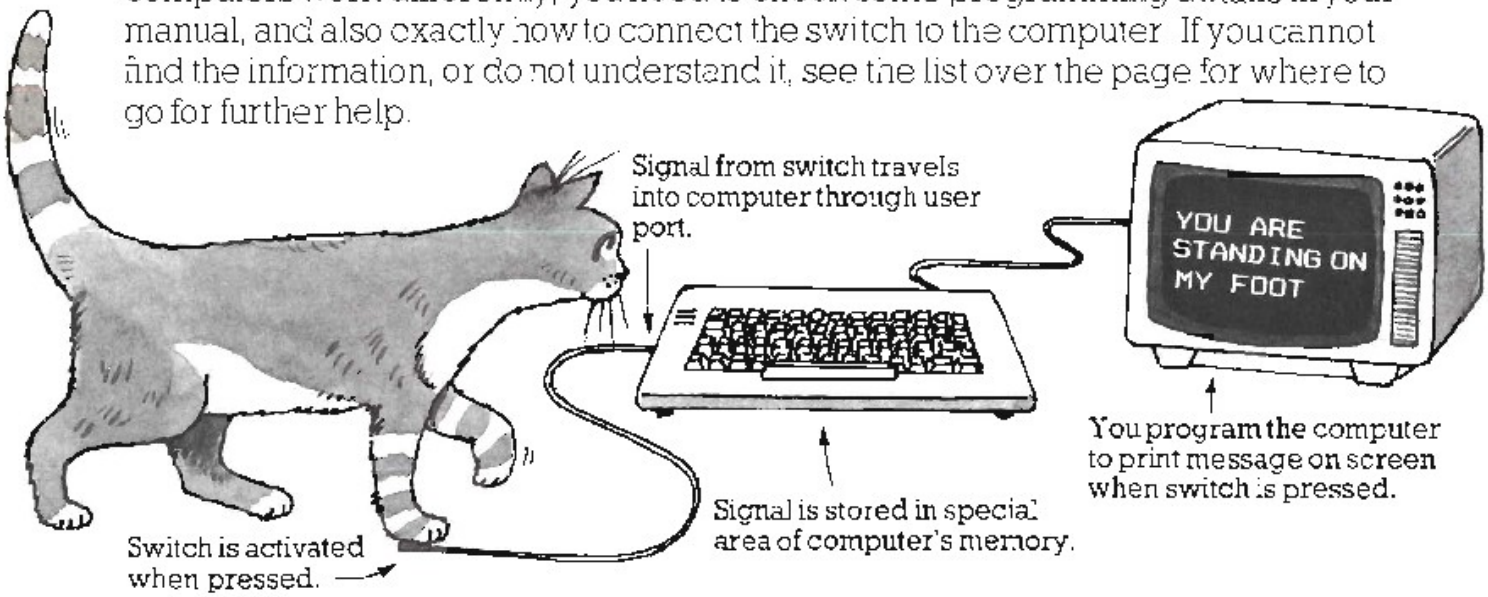
Before it can use a disk, the computer needs to be told how and where to store information on it. These instructions are called a Disk Operating System (DOS) and usually come on a floppy disk with the disk drive. A disk is divided into areas called tracks and sectors. Some of the tracks are reserved for a directory. This is a list of the programs on the disk, with their track and sector locations.

To use a disk drive, the computer needs a disk interface. This is usually a few chips which slot into the computer's printed circuit board. If your computer does not have a disk interface, you can have it fitted by a dealer.

Simple circuits to build

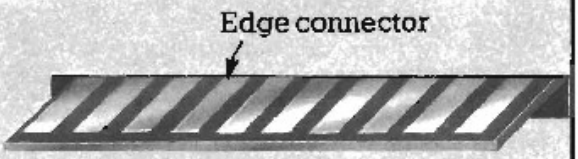
These next few pages show you some circuits to build and attach to your computer, with some ideas for how to use them. You connect them to a computer via the user port, or input/output port. If your computer does not already have a user port, ask a dealer if you can buy one to add on, or you may see one advertised in a computer magazine*. The first project shows you how to attach a switch to the computer, and how to program the computer to do different things according to the signals it receives from the switch.

The instructions for the circuits show how to build them, but as different computers work differently, you need to check some programming details in your manual, and also exactly how to connect the switch to the computer. If you cannot find the information, or do not understand it, see the list over the page for where to go for further help.

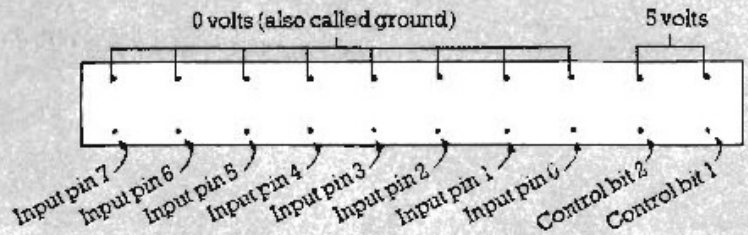


The user port

Your user port may have a different number of pins or tracks.



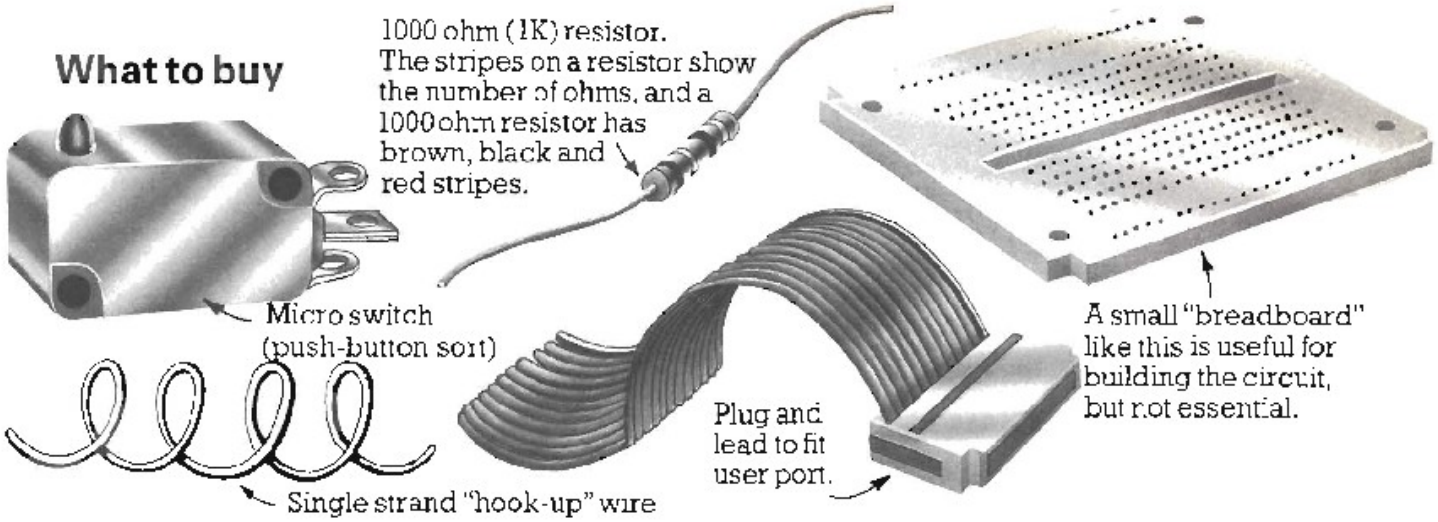
The user port, or input/output port, might be a socket with a number of pins in it, or an edge connector with metal tracks. Eight of the pins or tracks carry input signals into the computer. These signals are stored in the computer's memory as a binary number with eight digits, or bits. (All the information inside a computer is stored in binary numbers.) Some computers have another eight pins to carry signals out of the computer. Others use the same eight pins for input and output signals, and you have to give the computer a code telling it whether to receive or send signals. This is called setting the data direction register (DDR).



Groups of pins in the user port are assigned to different jobs, for instance, carrying signals or supplying different voltages. There should be a diagram in your manual telling you what each pin does. The input pins are usually numbered 0 to 7. Some user ports have "control bit pins" which are used when interfacing certain things. For this project, you will need to use a 0 volt and a 5 volt pin, and one input pin.

*The VIC 20, Pet and the BBC have user ports. For other machines, you need to buy a user port (or parallel input/output port).

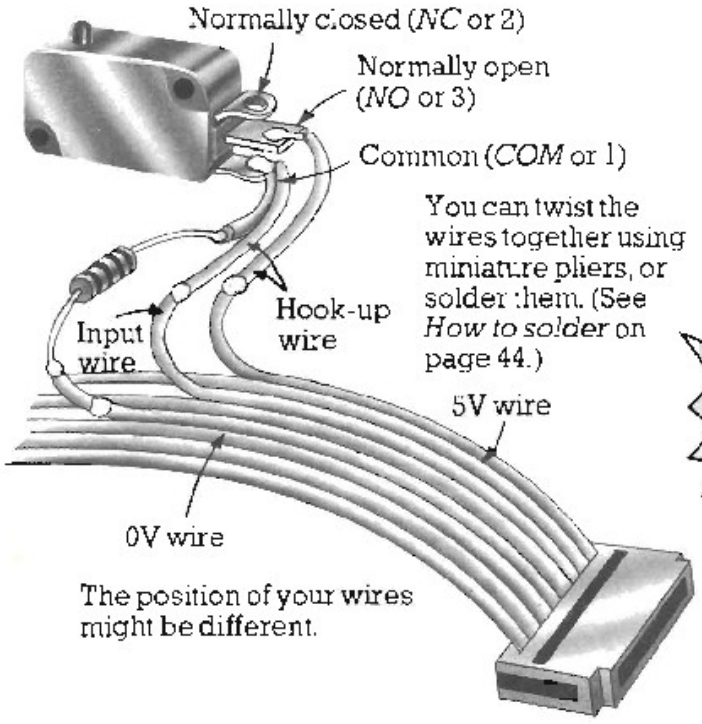
What to buy



You can buy most of this equipment cheaply at an electronics components shop, or by mail order.* You may need to go to a specialist computer shop for the plug and lead to fit the user port. The lead will probably be flat "ribbon" cable, consisting of lots of wires stuck together. When you buy it, ask which wires to connect to the 5 volt and 0 volt pins in the user port, and which wires to connect to the input pins.

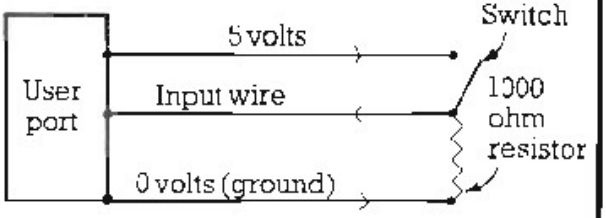
Connecting the switch

Your switch should have three places to connect wires, labelled 1, 2 and 3, or *NO* (normally open), *NC* (normally closed) and *COM* (common). Look for these labels when you buy the switch, and if they are different, ask what they mean.



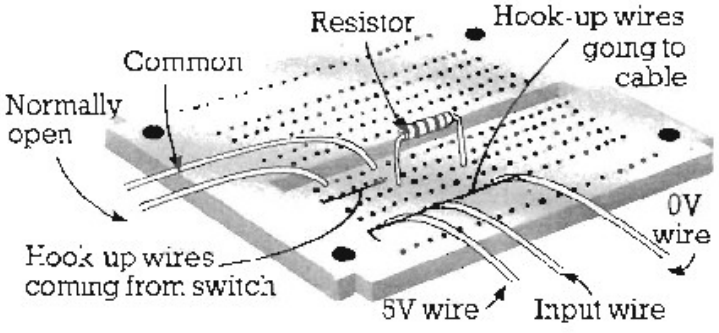
Using wire-strippers, remove about 1.5cm of the plastic from the wires to connect to the 5V, 0V and input pin in the user port. With hook-up wire, attach the 5V wire to *NO*, and the input wire to *COM*. Connect the resistor to *COM* also, and attach the other end of the resistor to the 0V wire. Do not connect anything to *NC*.

How the switch works



The switch is connected to a 0 volt, a 5 volt and an input pin in the user port. When the switch is closed, the user port receives 5 volts along the input wire, and the computer stores a 1 in its memory. When the switch is open, there is no voltage along the input wire, and the computer stores a 0.

Follow these instructions carefully. You can damage your computer if you wire up the switch wrongly.



If you prefer, you can poke the wires into a breadboard as shown. This has lines of holes in it which are connected underneath by metal tracks, and you do not need to solder the wires to it.

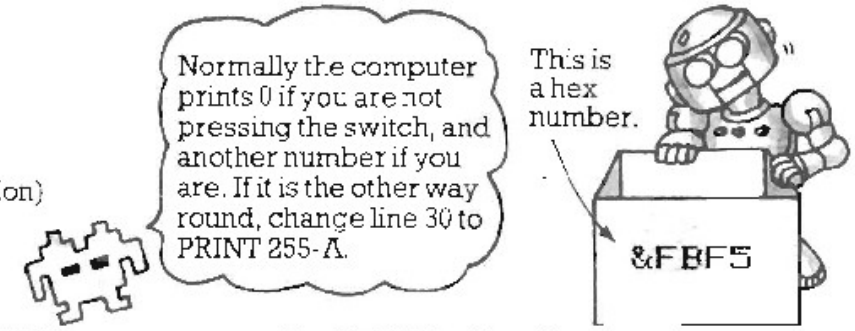
*There will be some more to buy for the projects on the next few pages.

Programs for the switch circuit

When you have wired up the switch, plug it into the user port. Then you can test it with the program below which shows you what happens in the computer's memory when you press the switch. Signals received by the user port are stored in one place, or location, in the memory. Each location has an address. You need to look up the address of the user port's memory location in your manual. If you cannot get the switch to work, check all the connections. If you are really stuck, look at the HELP list on the opposite page.

Test program*

```
10  (Set data direction
    register if necessary.)
20  LET A=PEEK (Address of user
                port's memory location)
30  PRINT A

40  GOTO 20
```



This program uses the command PEEK to tell the computer to look in the user port's memory location. It copies what is there into variable A. Check in your manual whether you need to set the data direction register to tell the computer to expect input signals, and look up the address of the memory location. If your computer uses a different command to PEEK, look up how to find out what is stored in the user port's location. The address of the user port's memory location may be given in your manual as a decimal number, or in hexadecimal (hex for short). Copy it exactly as it is shown. Hex numbers are often preceded by the symbol "&".

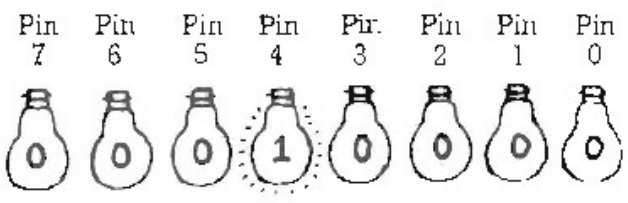
Binary counting

The user port stores the signals it receives as binary numbers, like all the information inside a computer. Binary numbers are made up of 1s and 0s, and reading from the right, these digits show how many 1s, 2s, 4s, 8s and so on there are in a number. To convert binary to decimal, add the numbers at the tops of the columns with 1s in.

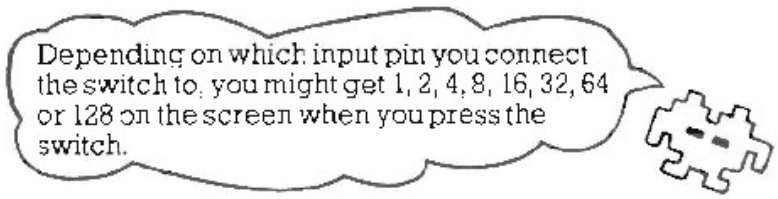
128	64	32	16	8	4	2	1	
0	0	0	0	0	1	0	0	4
0	0	1	0	0	0	1	1	35
1	0	0	0	1	0	0	0	136
1	1	1	1	1	1	1	1	255
0	1	1	1	0	1	0	1	117
1	1	0	1	1	0	1	0	218

Decimal equivalents to these binary numbers.

The user port and the memory



The user port's memory location stores a different binary number depending on the signals it gets from the user port. The number is made up of eight binary digits (bits) and each bit corresponds to the signal received by one of the eight input pins. If a bit receives a voltage, it changes to a 1. If it receives 0 volts, it changes to 0.



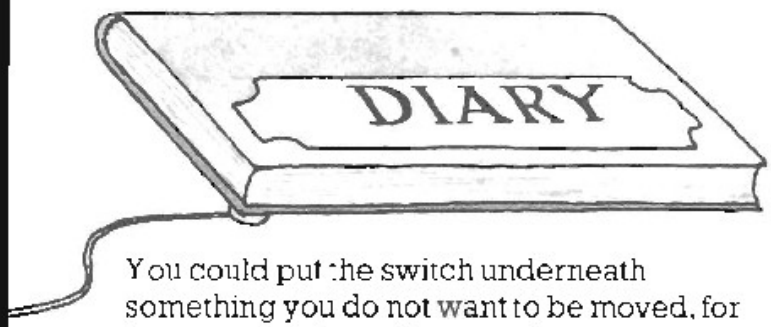
When you run the test program, all the bits are 0 to start with, so you get a 0 on the screen. When you press the switch one of the bits changes to 1, and the computer prints the decimal equivalent to the binary number it now has stored in the memory location. In the picture, the switch is connected to pin 4, so the computer stores 00010000 when the switch is pressed. This is 16 in decimal so the computer prints 16 on the screen.

*If you are using a BBC or a VIC 20, see page 47.

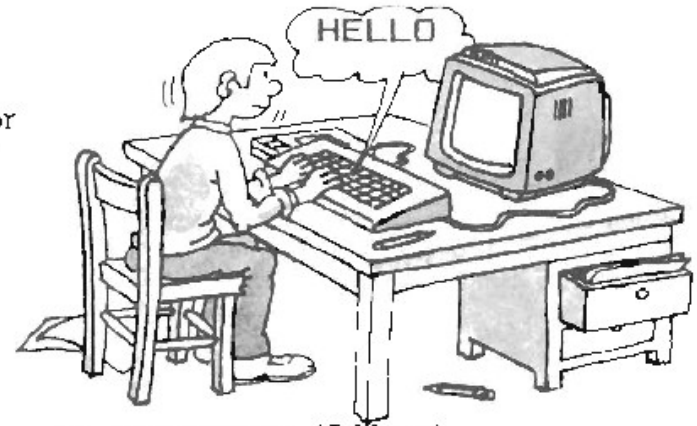
Ideas for things to do

Below and over the page are some ideas for using switches. You can probably think of others.

```
10 LET A=PEEK (Address)
20 IF A<>0 THEN GOTO 10      i.e. if switch
                             is closed
30 IF A=0 THEN (Program in alarm noise)
```



You could put the switch underneath something you do not want to be moved, for instance, a private diary, and program the computer to make an alarm noise if someone picks up the diary.

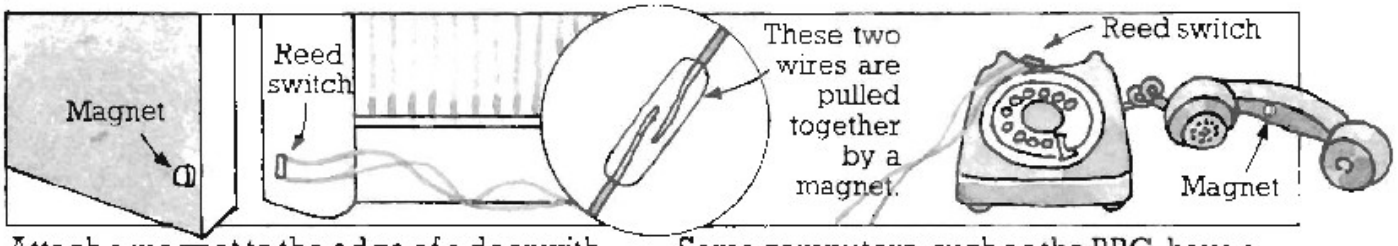



```
10 LET A=PEEK (Address)
20 IF A=0 THEN GOTO 10       i.e. if switch
30 IF A<>0 THEN              is open
PRINT "HELLO"
```

Tape the switch to a chair and put a thin cushion over it. Program the computer to print "HELLO" when someone sits down and activates the switch. If you have a speech synthesizer you could make the computer say something by adding a speech command at line 30 instead of PRINT.

Intruder alarm

You can build an intruder alarm using a different kind of switch, called a reed switch. This is activated when a magnet is passed over it. You connect it to the computer in the same way as a micro switch. Connect the input wire and 0V wire with its resistor to one end of the reed switch, and the 5V wire to the other end.



Attach a magnet to the edge of a door with sticky tape, and fix the reed switch next to it on the door frame. Program the computer to sound an alarm when the door is opened, using the program at the top left on this page.

Some computers, such as the BBC, have a TIME command. You can use it together with a reed switch to time how long the switch is open or closed. Try using it to time telephone calls, for instance.

HELP!

If you cannot get the projects to work, or cannot find the necessary information in your manual, here are some ideas for where to go for further help:

- Contact and join a local user group – they are advertised in computer magazines, or ask at your local library.
- Ask someone experienced in computing, such as a teacher, to give you some help.
- Contact your computer's manufacturers – they should be able to answer your questions and might have extra information they could send you.
- Write to a home computer magazine – they may publish your letter or put you in touch with a user group.
- Ask at your local computer dealers.

If you still have problems after trying these, write to Judy Tatchell at Usborne Publishing (the address is at the back of the book) stating your difficulty and the make of your computer.

Build a "bitswitch" keypad

On the next few pages there are two projects which show how the user port works. The bitswitch keypad has eight switches each connected to a different bit in the user port's memory location. You can control whether the bits are 1s or 0s by pressing the switches, and the computer stores a different binary number depending on which switches are pressed. You can program the computer to print the decimal equivalents of these binary numbers on the screen, and you can also find out how to make it print letters.

Mounting the switches

You need to build eight identical circuits, one for each switch. The picture below shows the first two circuits for you to copy, and you can see what the finished set of switches looks like at the top of the next page. There are instructions for how to solder on page 44. When you have built the circuits, connect the input wire of the first switch to input pin 0, the second switch to input pin 1, and so on.

What to buy

8 small, flat keyboard switches. Ask for normally open switches with two legs to fit Veroboard.

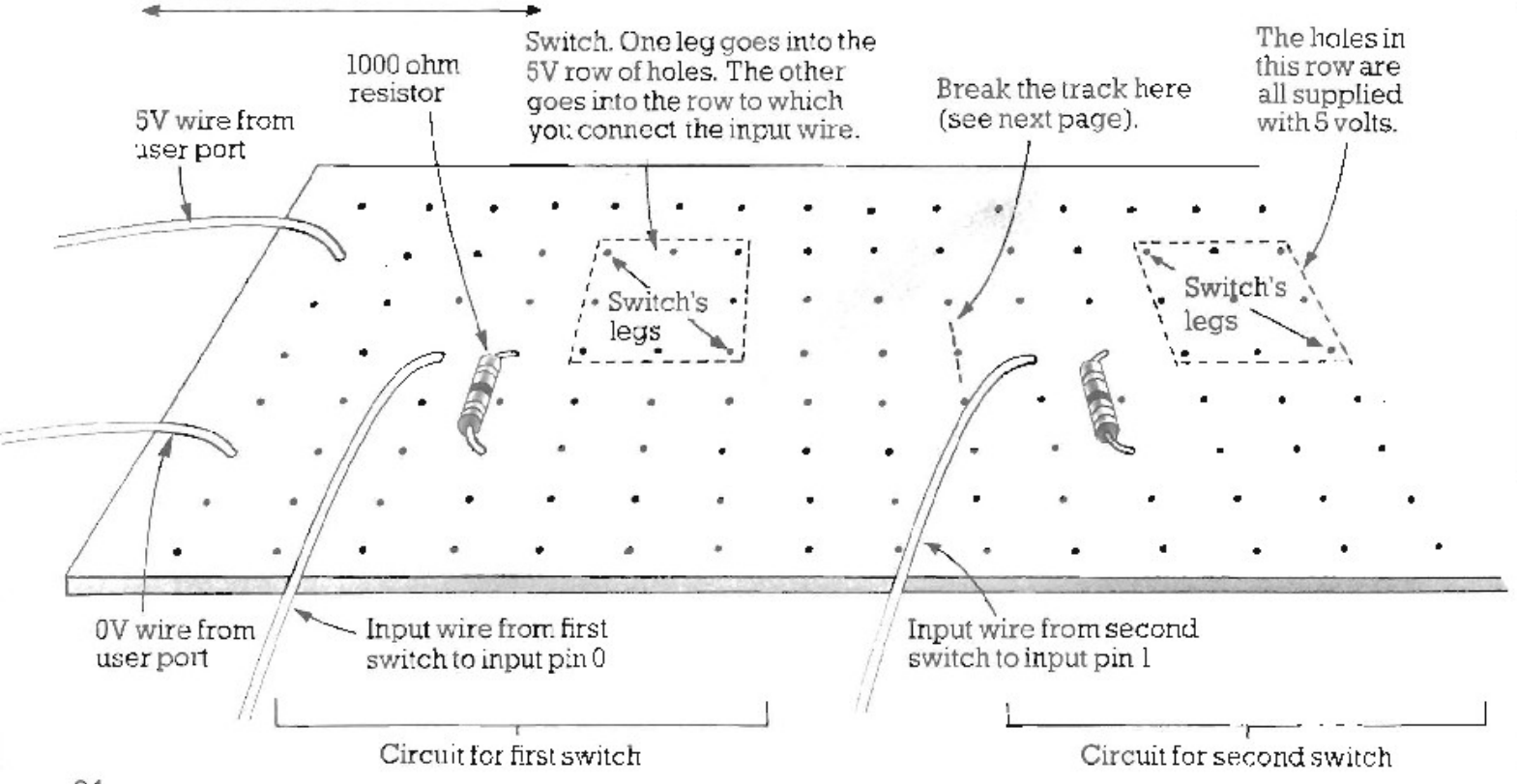
8 × 1000 ohm (1K) resistors (brown, black and red stripes)

Single strand hook-up wire

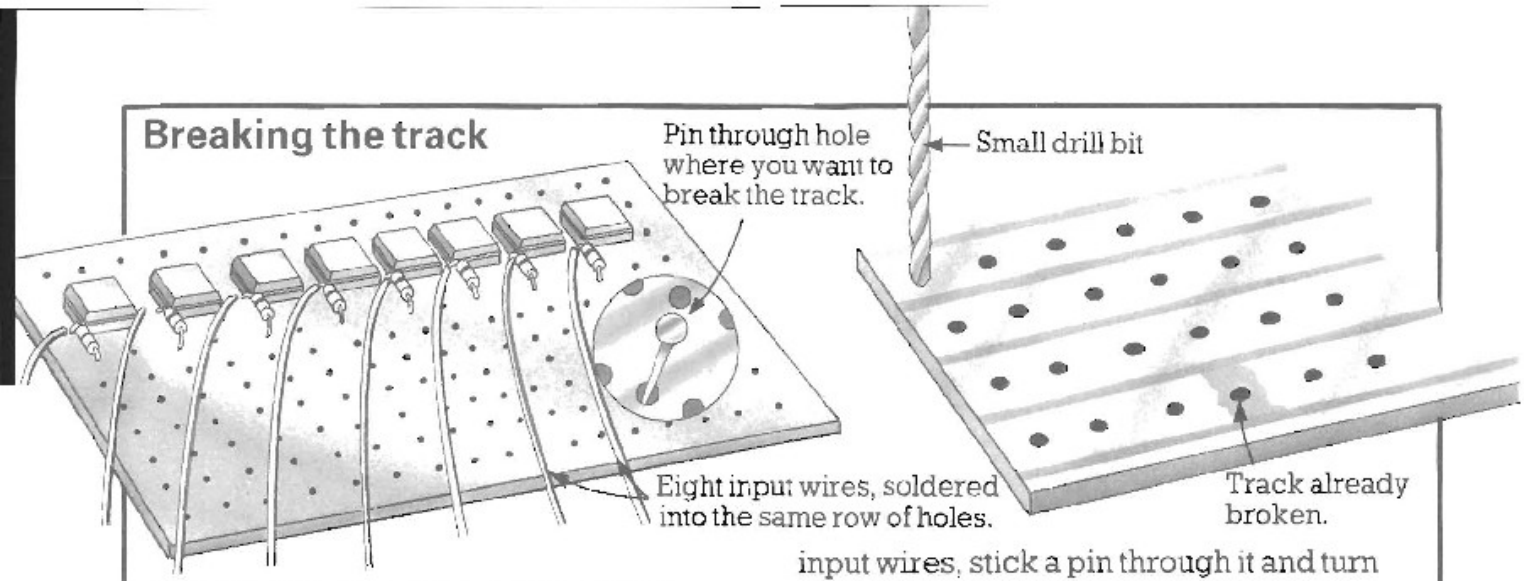
Plug and lead for user port

Piece of Veroboard about 24cm long, with 5 holes to a centimetre.

Veroboard has copper tracks on the underside linking the holes in rows. To build the circuit, you push wires and legs of resistors and switches through the holes, and solder them underneath. The current flows along the copper tracks between them.



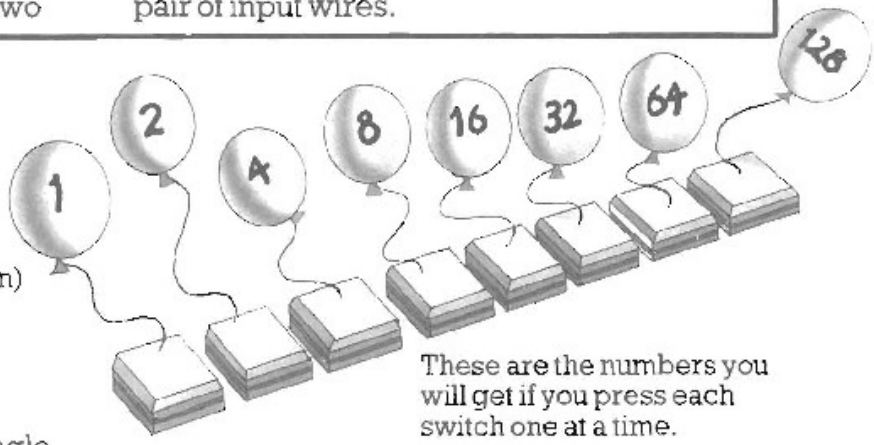
Breaking the track



The finished circuits should look like this. Next, you need to break the copper track on the underside of the Veroboard between the input wires, to stop the current flowing between them. To identify a hole in the track between two input wires, stick a pin through it and turn the Veroboard over. Scrape away the copper track around that hole with a penknife, or twist a small drill bit over the hole to remove the copper. You need to do this seven times, once between each pair of input wires.

Programs for the switches

```
10 (Set DDR if necessary)
20 LET A=PEEK (Address of user
                port's memory location)
30 PRINT A
40 GOTO 20
```



These are the numbers you will get if you press each switch one at a time.

This is the same program as for the single switch in the previous project, but here you can get lots of different numbers on the screen by pressing different switches. The computer stores the number in binary, but converts it to decimal before printing it on the screen.

The number shown on the screen when you press a switch depends on which bit in the user port the switch is connected to. You can get any number between 0 and 255 by pressing different combinations of switches.

Character keypad

```
10 (Set DDR if necessary)
20 LET A=PEEK (Address)
30 IF A=0 THEN GOTO 20
40 IF A<>0 THEN GOSUB 100
50 LET X$=CHR$(C)
60 PRINT X$
70 LET A=PEEK (Address)
80 IF A<>0 THEN GOTO 70
90 GOTO 20
100 FOR I=1 TO 10
110 NEXT I
120 LET C=PEEK (Address)
130 RETURN
```

There might be an ASCII table in your manual telling you which numbers represent which letters.

This program turns the set of switches into a very simple version of a keypad called Microwriter. This has buttons on it, and by pressing combinations of these you can print letters, numbers and symbols on the screen.

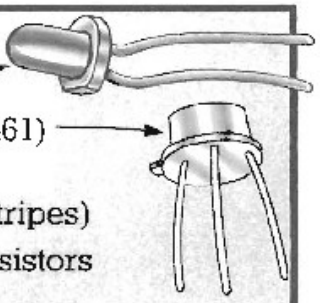
The program uses CHR$ to convert the numbers stored when you press the switches into ASCII characters* to print on the screen. There is a short delay loop in the subroutine in case you do not press all the switches that make up a number at exactly the same time, and you can alter the number in the delay loop if necessary. The computer also tests to see if you have taken your fingers off the switches before going back to wait for the next number, so you do not get a row of the same characters (line 80).

*There is an explanation of ASCII on page 46

Binary light display project

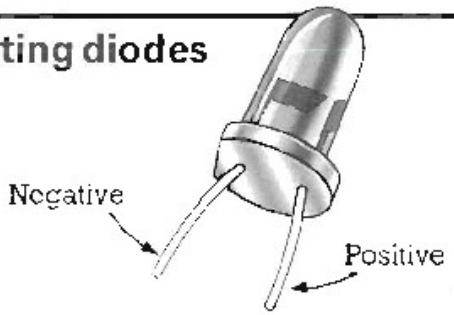
You can build a row of light-emitting diodes (LEDs), which light up in different patterns when you type numbers on the keyboard. The numbers are stored in binary in the user port's memory location. Each LED is connected to a bit in that location, and turns on or off depending on whether the bit is 1 or 0. If your user port uses the same pins for input and output, set the data direction register to output. If it has separate pins, connect the LEDs to the output pins.

What to buy



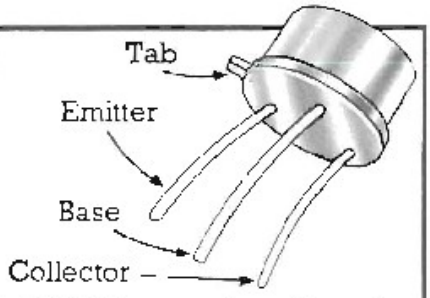
- 8 LEDs
- 8 transistors (type BC461)
- 8 × 82 ohm resistors (grey, red and black stripes)
- 8 × 2200 ohm (2.2K) resistors (three red stripes)
- Single strand hook-up wire
- Plug and lead for user port
- Veroboard (at least 13 cm long and 4 cm wide)

Light emitting diodes



A current only flows through an LED in one direction, from the positive to the negative leg, so you need to solder it on to the board the right way round. Most LEDs have a flat side indicating which is the negative leg, and this leg ends in a larger piece of metal which you can see inside the LED.

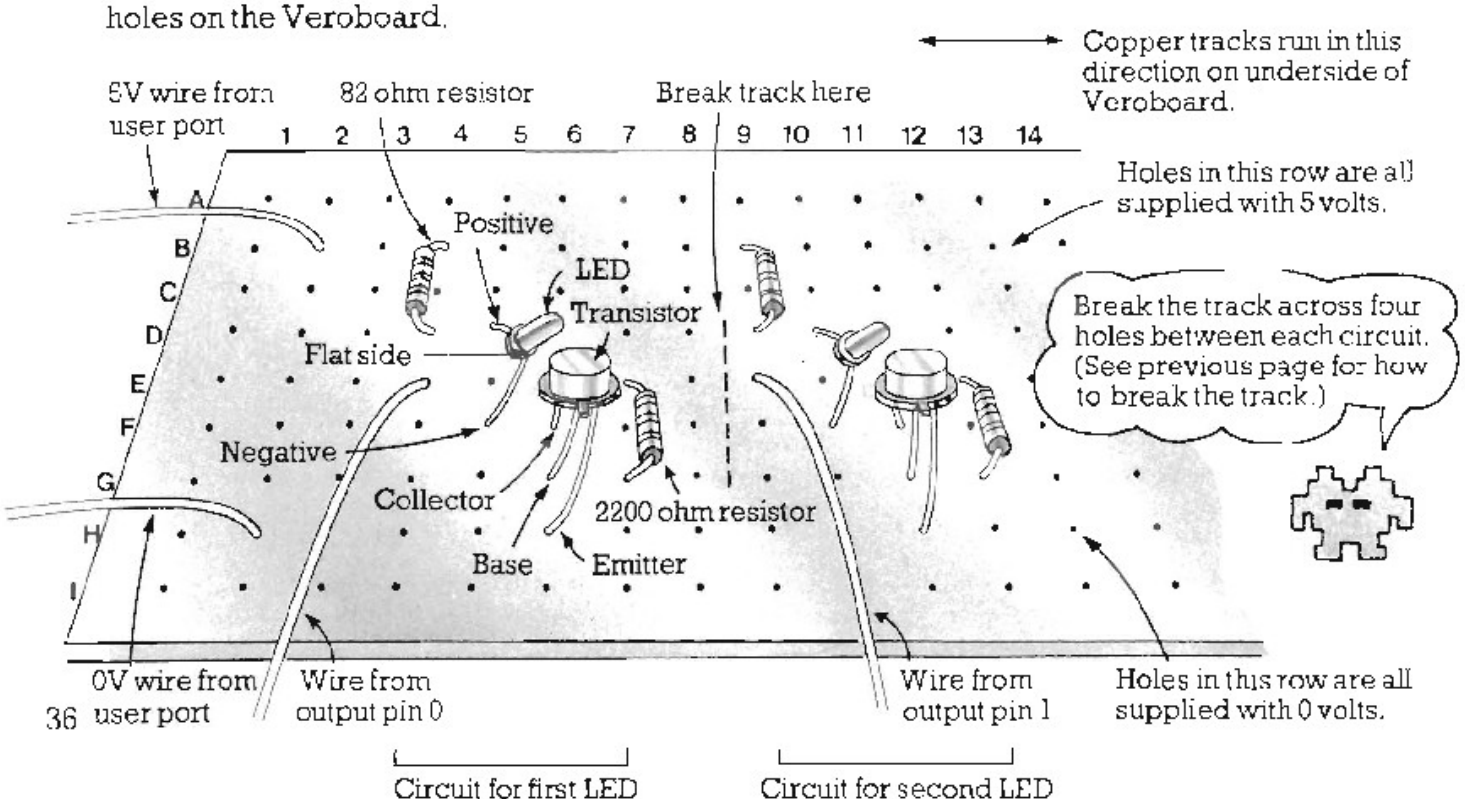
Transistors



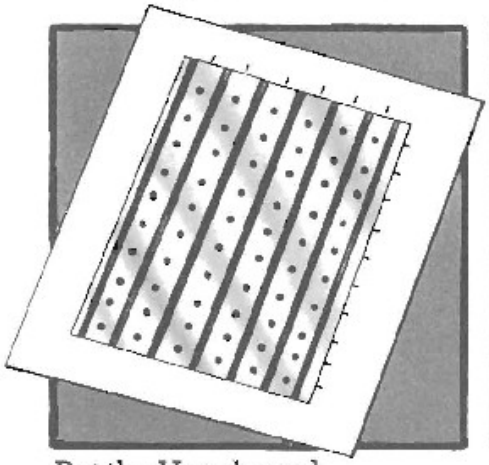
Transistors have three legs, and you have to be careful to solder each leg into the correct hole, or the transistor might be damaged. A BC461 transistor has a little tab next to the emitter leg. The base is the leg in the middle. The third leg is the collector. Check this with your supplier if you are not sure.

Mounting the LEDs

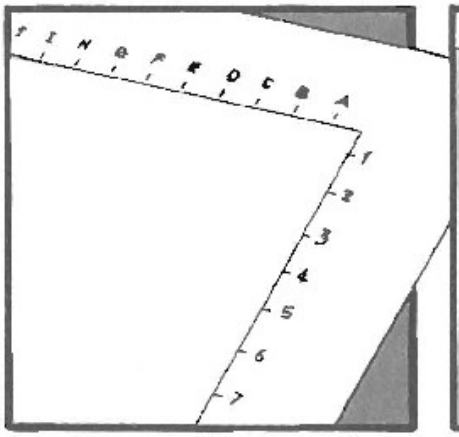
You need to build eight LED circuits along the board, exactly like the two shown below. There are instructions and a key on the opposite page showing you how to find the right holes on the Veroboard.



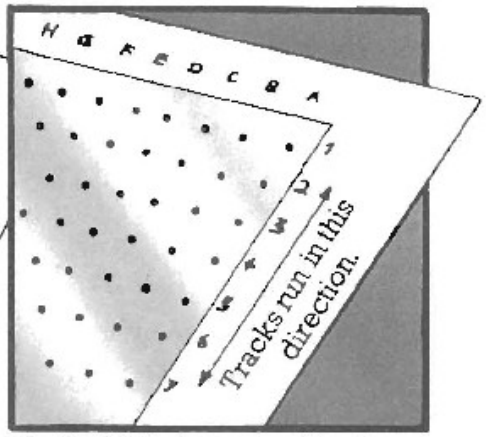
Finding the holes on the Veroboard



Put the Veroboard copperside up on a plain piece of paper, and mark the lines of holes and tracks.



Label the tracks across the top with letters and put numbers down the side, exactly as shown.



To find a hole named in the key below, lay the board on the paper copperside down, and read off the hole's letter and number.

Key

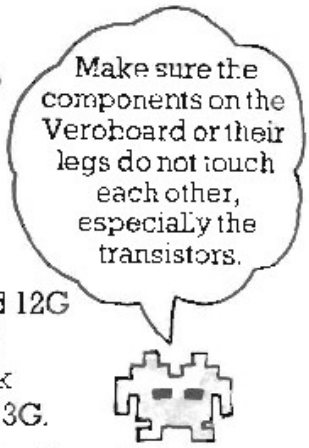
5V wire from user port – 2B
0V wire from user port – 2H

Circuit for first LED:

82 ohm resistor – 4B and 4D
LED: positive leg – 5D
negative leg – 5F
Transistor: collector – 6F
base – 6G
emitter – 6H
2200 ohm resistor – 7E and 7G
Wire from output pin 0 – 4E
You need to break the track at holes 8D, 8E, 8F, 8G.

Circuit for second LED:

82 ohm resistor – 9B and 9D
LED: positive leg – 10D
negative leg – 10F
Transistor: collector – 11F
base – 11G
emitter – 11H
2200 ohm resistor – 12E and 12G
Wire from output pin 1 – 9E
You need to break the track at holes 13D, 13E, 13F and 13G.



The components for this circuit are in the same tracks as the first circuit, but five holes along. Build six more LED circuits with the same layout as these, each one five holes along from the last.

Programs to try

```
10 (Set DDR to output signals
   if necessary.)
20 PRINT "TYPE IN A NUMBER"
30 PRINT "BETWEEN 0 AND 255"
40 INPUT X
50 POKE (Address of user port's
   memory location) , X
60 GOTO 20
```

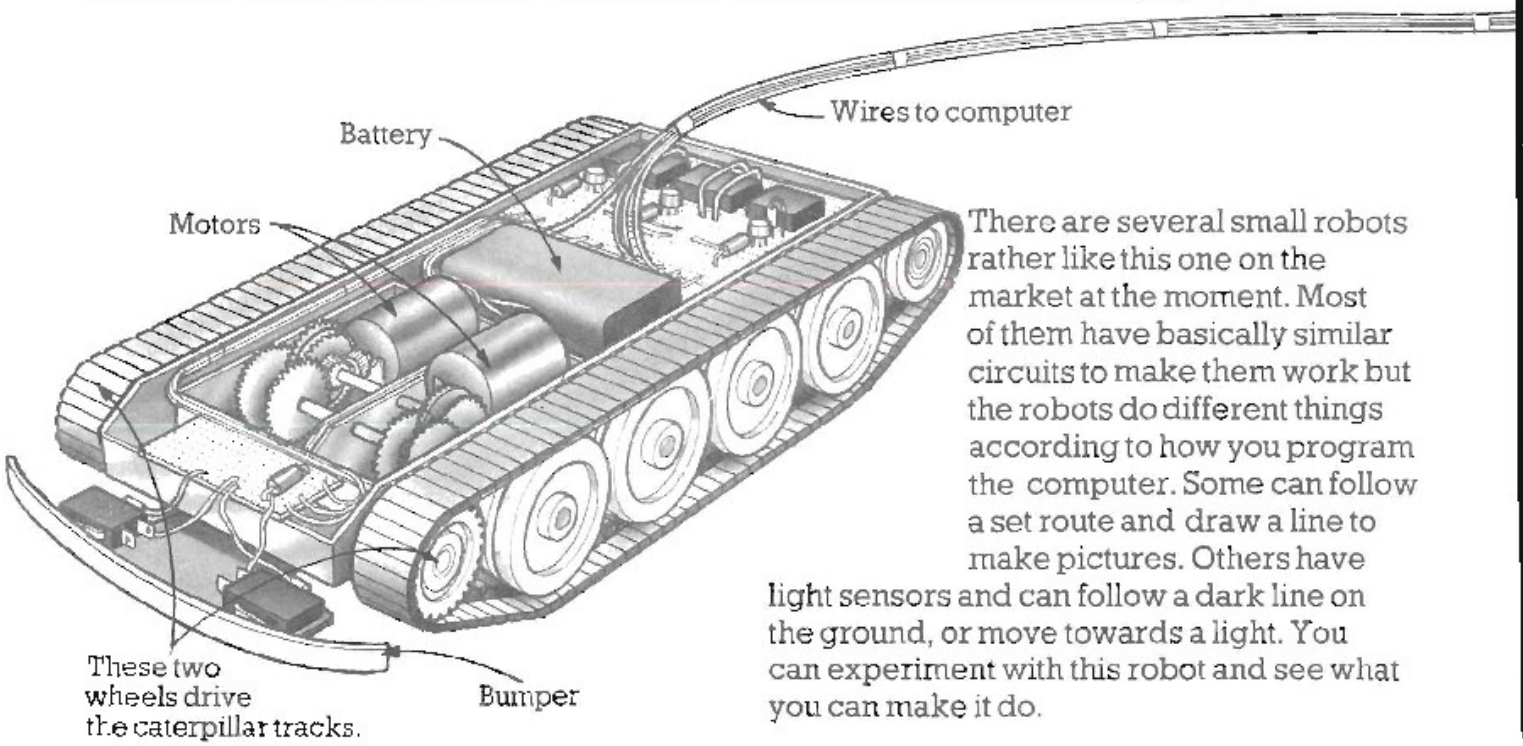
POKE puts X, the number you type in, into the user port's memory location. The number is stored in binary form, and the LEDs light up in the pattern of 1s in the binary number.

```
10 (Set DDR to output signals
   if necessary.)
20 FOR I=0 TO 255
30 POKE (Address) , I
40 FOR J=1 TO 30
50 NEXT J
60 NEXT I
```

This program makes the LEDs light up in the pattern of 0s and 1s for each binary number from 0 to 255. Alter the figure in the delay loop in line 40 if the program runs too fast or too slow.

A robot to build

On the next few pages you can find out how to build a computer-controlled robot vehicle. You plug it into the computer via the user port, or input/output port. The robot has a bumper on its front and when you start it up, it can negotiate an obstacle course on the floor. The program for this is on page 43.



There are several small robots rather like this one on the market at the moment. Most of them have basically similar circuits to make them work but the robots do different things according to how you program the computer. Some can follow a set route and draw a line to make pictures. Others have light sensors and can follow a dark line on the ground, or move towards a light. You can experiment with this robot and see what you can make it do.

This robot is a battery-powered model vehicle, adapted for computer control. There are other suggestions for how to build a body for the robot below. The working parts are two motors and two gearboxes which drive a pair of wheels. You also need to build an electronic circuit to control the robot. This is quite complicated but there are detailed instructions for you to follow. You can buy the parts for the circuit at an electronics components shop.

Ideas for the robot's body

★ Here are some suggestions for where to buy parts for the robot's body. Before you buy anything, read through all the instructions on the next few pages so you have a clear idea of what to do and what you need. You can take this book with you to the shops, too, so you can show the supplier what you are looking for if there is any doubt.

★ You can use a pair of motors, gearboxes and wheels from a construction kit, like Fischertechnic. They are easy to assemble and are designed to work together. Assemble the parts on a baseplate from the construction kit.

★ A battery-powered, motor-driven vehicle from a toy shop or model shop can provide a body for the robot. When you buy it, check that it has two motors which can drive the wheels independently forwards and backwards. Do not buy a radio-controlled model.

★ A good model shop should be able to supply you with a pair of motors and gearboxes and a pair of wheels, which you can screw to a piece of 10mm thick plywood. The voltage range of the motors must be between 3 volts and 12 volts. You also need a small balancing wheel to go at the back.

What you need for the robot's control circuit

Piece of Veroboard about 8cm × 8cm (with holes 0.1 inches apart).

2 double-pole changeover subminiature relays, coil voltage 6V d.c., coil resistance 250 ohms (or anything over 50 ohms will do).

1 single-pole subminiature relay, coil voltage 6V d.c., coil resistance 250 ohms (or anything over 50 ohms).

3 transistors type BC107 or BC108

3 × 2.2K resistors

3 diodes type IN4001, IN4002 or IN4003. (Do not use Zener diodes.)

Hook-up wire

Sticky tape e.g. masking tape

Battery to match your motors, e.g. a 9V battery for a 9V motor.

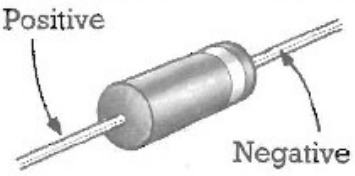
Plug and lead for user port.

Soldering the components on to the Veroboard

Label the holes and tracks on the Veroboard as shown on page 37. Make sure the ends of the tracks are labelled with letters and the holes are labelled down the side with numbers. There is a key over the page telling you in which holes to put the components. It does not matter which way round you solder resistors into the Veroboard but it is very important that you put the legs of diodes, transistors and relays in the right way round.* You can find out how to identify each leg on a transistor on page 36.

Diodes

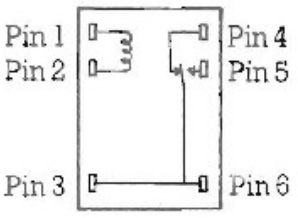
A diode only allows the current to flow through it in one direction. The end with the stripe marks the negative leg and this goes towards the 0 volts, or negative, side of the circuit.



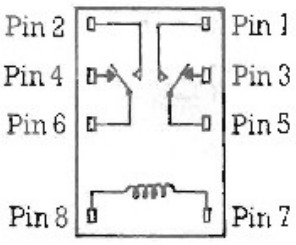
Relays

A relay is a tiny electronic switch. A double-pole relay has two switches inside and a single-pole has one. The double-pole relays in the robot's circuit switch the motors on and make them go either forwards or backwards. Each motor drives a wheel. The single-pole relay turns the power supply on and off.

Circuit diagram for single-pole relay



Circuit diagram for double- pole relay



When you buy your relays, ask for circuit diagrams showing how they work inside and compare them to these. If a diagram is labelled "pin view", you hold the relay with its legs facing you to look at it. Otherwise, you look at the relay from the top. If the circuit looks the same but the pins are labelled differently, substitute the numbers used here.

Place your relays over these diagrams.

Single-pole

Double-pole

The relays you use must have their pins in the same position and the same distance apart, as the ones shown here or they will not fit into the right holes on the Veroboard. To make sure you buy suitable relays, take this book with you to the shop and place the relays over the diagrams above.

*There are instructions for how to solder on page 44.

Assembling the robot

These pages show how to make the robot's control circuit and how to connect it to the computer. Follow the instructions very carefully, as a single error will stop the circuit working.

1. Solder the legs of the components into these holes on the Veroboard.

Double-pole relay:
- Pin 1 – Z5
- Pin 2 – Z8
- Pin 3 – W5
- Pin 4 – W8
- Pin 5 – U5
- Pin 6 – U8
- Pin 7 – S5
- Pin 8 – S8

Double-pole relay:
- Pin 1 – P5
- Pin 2 – P8
- Pin 3 – M5
- Pin 4 – M8
- Pin 5 – K5
- Pin 6 – K8
- Pin 7 – I5
- Pin 8 – I8

Single-pole relay:
- Pin 1 – A8
- Pin 2 – B8
- Pin 3 – F8
- Pin 4 – A5
- Pin 5 – B5
- Pin 6 F5

Diode:
- Positive – B10
- Negative – A10

Diode:
- Positive P10
- Negative – P3

Diode:
- Negative – Z3
- Positive – Z10

*Transistor:
- Collector – B13
- Base – C13
- Emitter – D13

Transistor:
- Collector – P12
- Base – O12
- Emitter – N12

Transistor:
- Collector – Z12
- Base – Y12
- Emitter – X12

Resistor: C15 and C19

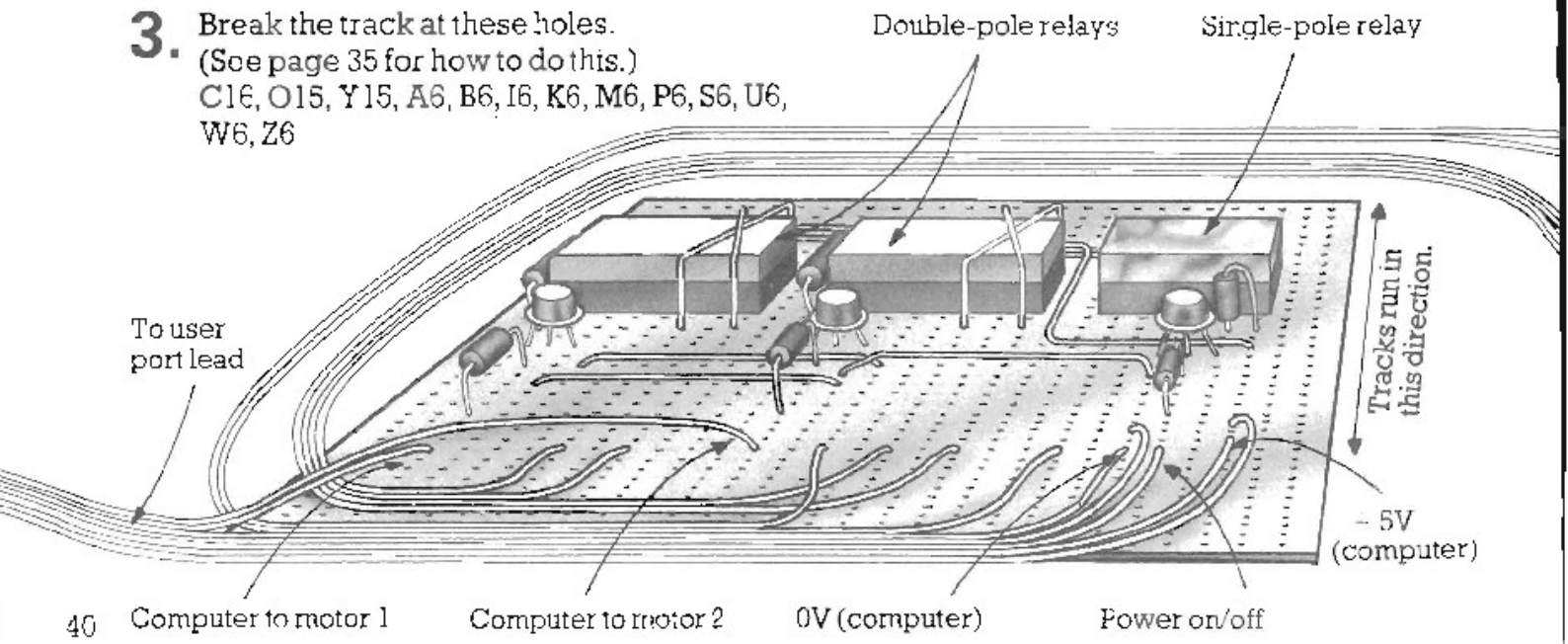
Resistor: O14 and O19

Resistor: Y14 and Y19

2. Cut 11 pieces of wire each about 10cm long and strip about 1cm of the plastic coating from each end using wire strippers. Connect the following pairs of holes with the pieces of wire, soldering each end to the Veroboard underneath.

D15 and N15
N16 and X16
M14 and W14
K3 and I10
S3 and U10
I3 and K10
U3 and S10
A12 and P1
Z2 and P2
W2 and M2
M3 and B3

3. Break the track at these holes. (See page 35 for how to do this.)
C16, O15, Y15, A6, B6, I6, K6, M6, P6, S6, U6, W6, Z6



*See page 36 for how to identify each leg.

Connecting up the robot's circuit

1. Cut seven pieces of wire about 3m long. Strip the plastic from each end. Solder one end of each wire into the following holes, labelling each wire with a sticky label or piece of masking tape.

Wire 1 – A23	+5V (computer)
Wire 2 – C23	Power on/off
Wire 3 – D23	0V (computer)
Wire 4 – F23	+ volts (battery)
Wire 5 – Y23	Computer to motor 1
Wire 6 – M23	– volts (battery)
Wire 7 – O23	Computer to motor 2

2. Cut four pieces of wire about 20cm long. Strip each end and solder one end of each into the following holes, labelling the wires as you go.

Wire 8 – S23	Motor 1A
Wire 9 – U23	Motor 1B
Wire 10 – K23	Motor 2A
Wire 11 – I23	Motor 2B

3. Identify the wires in the user port's lead which connect to output pins 0, 1 and 2 in the user port. (You may need to ask the supplier when you buy the lead.) Strip 1cm of plastic from the ends of these wires and solder them to the labelled wires on the robot's circuit as follows:

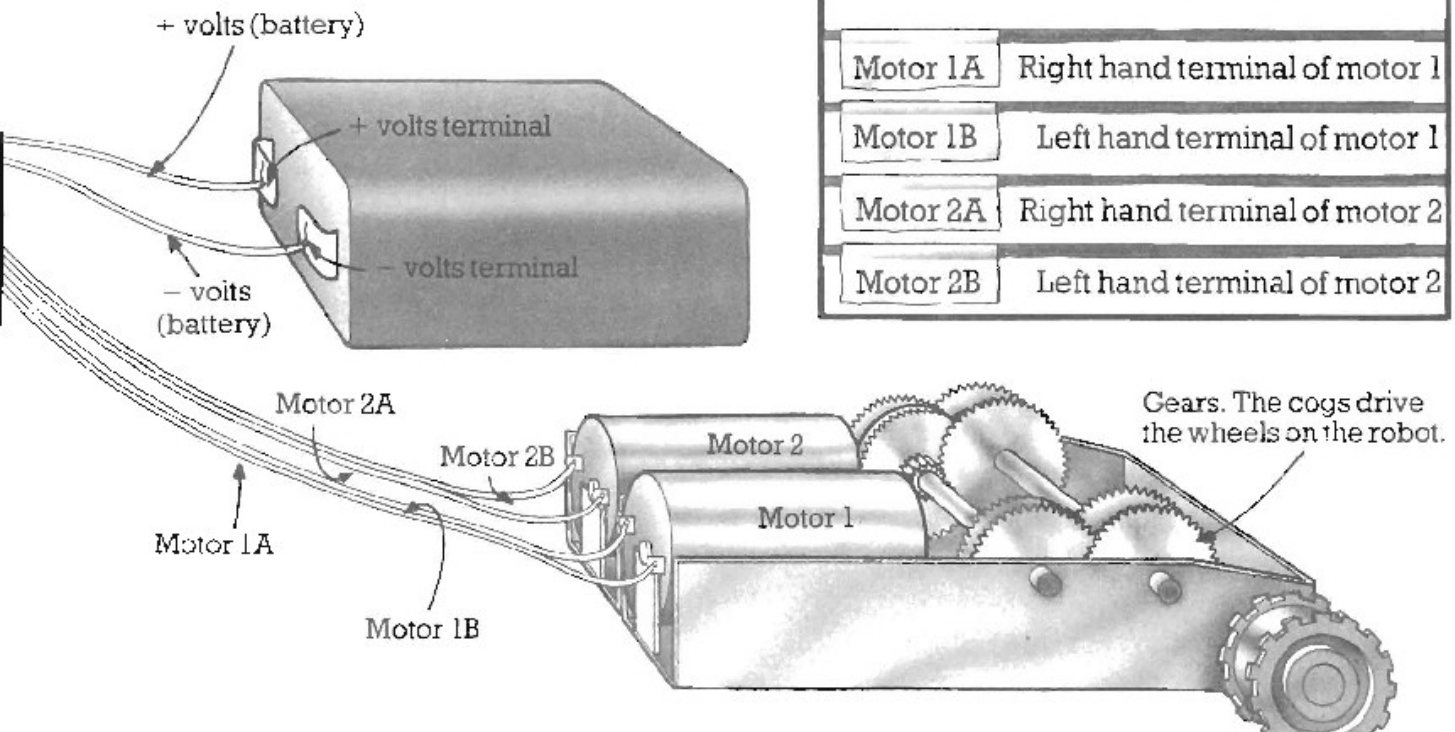
Label on wire	User port lead
Computer to motor 1	Wire to output pin 0
Computer to motor 2	Wire to output pin 1
Power on/off	Wire to output pin 2
+5V (computer)	Wire to 5V pin
0V (computer)	Wire to 0V pin

4. Tape the following two wires to the ends of the battery:

Label on wire	Battery
+ volts (battery)	+ volts terminal
– volts (battery)	– volts terminal

The last four wires connect to the motors. Motor 1 is the motor on the right as you look down at the robot from above and Motor 2 is the motor on the left.

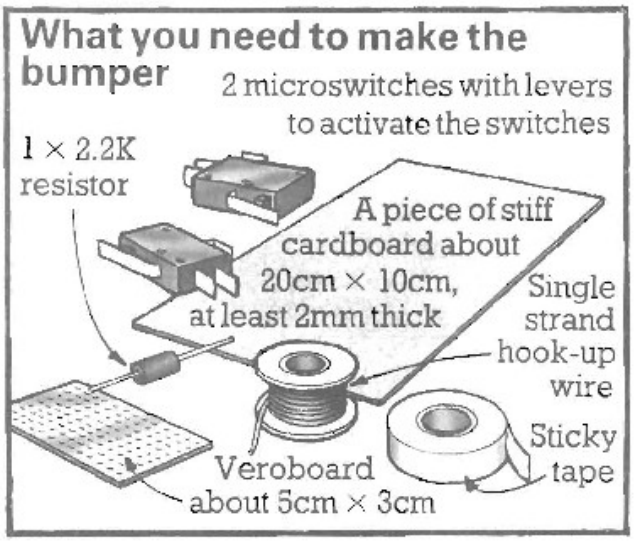
Motor 1A	Right hand terminal of motor 1
Motor 1B	Left hand terminal of motor 1
Motor 2A	Right hand terminal of motor 2
Motor 2B	Left hand terminal of motor 2



How to make the robot's bumper

The robot has a bumper attached to its front, with a switch at either side. The switches are activated when the robot bumps into something and the computer's program tells the robot to reverse and change direction.

What you need to make the bumper



1. Cut pieces of wire to the following lengths and strip 1cm of the plastic from either end. Solder one end of each wire into these holes on the Veroboard:

2 pieces of wire about 25cm long – 2J
– 2C
2 pieces of wire about 5cm long – 7C
– 11C
1 piece of wire about 3m long – 2E
2 pieces of wire about 5cm long – 5E
– 9E
Solder the resistor into holes 6E and 6J

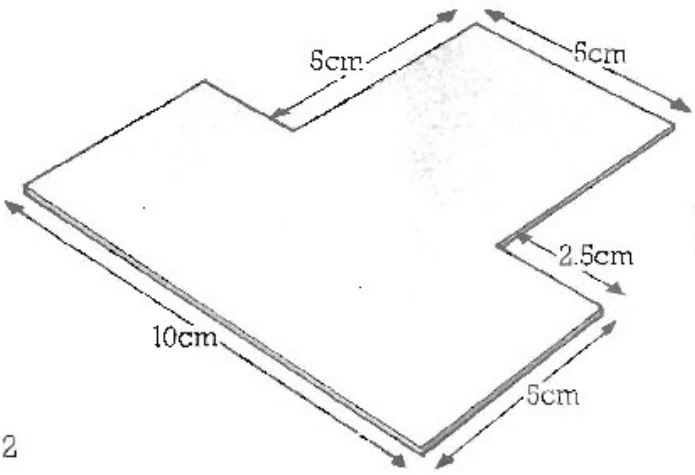
2. Solder the other end of the wire in 7C to the common (*com* or 1) connection on switch 1 and the other end of the wire in 11C to the *com* connection on switch 2. Solder the other end of the wire in 5E to normally open (*NO* or 3) on switch 1 and the other end of the wire in 9E to *NO* on switch 2.

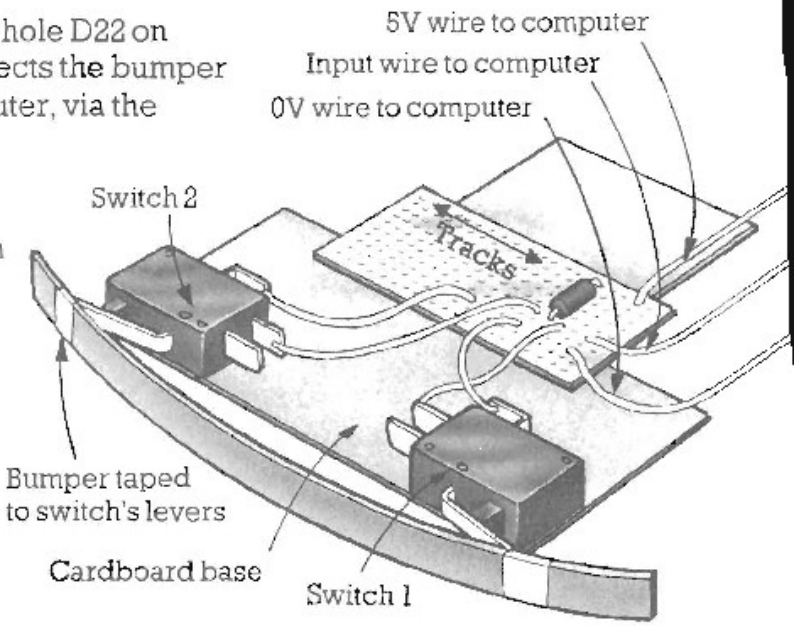
3. Solder the end of the input wire in hole 2E to the wire in the user port lead which connects to input pin 3.

4. Solder the end of the 5V wire (in hole 2C) into hole A21 on the robot's control circuit board. This carries a 5V power supply to the bumper circuit from the computer, via the robot's control circuit.

5. Solder the end of the 0V wire in hole 2J into hole D22 on the robot's control circuit board. This connects the bumper circuit to the 0V power supply in the computer, via the robot's control circuit.

6. Cut a piece of cardboard this size for the base. Stick the circuit and both switches on to the base with tape. You can adjust the distance between the switches according to the width of your robot's body.





7. Cut a strip of cardboard about 18cm × 1.5cm for the bumper (the length will depend on the size of the robot). Tape it to the levers on the switches. Attach the bumper unit to the robot by taping it underneath or on the top at the front – wherever you can fix it firmly.

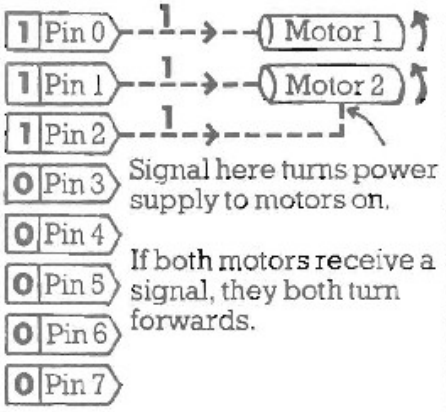
A program for the robot

This program makes the robot move forwards until it hits something, and then reverse and swivel right or left before setting off again. Try building an obstacle course from piles of books and other objects and see if the robot can get round it. Before you can run the program, you need to insert some numbers which will make the program work.

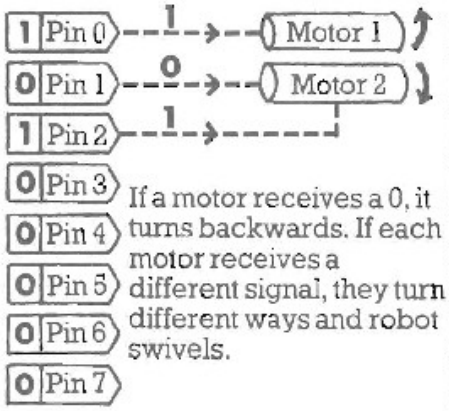
Finding the correct numbers

Signals from the pins in the user port activate the motors. You send the signals by using the command POKE, followed by different numbers which make the robot go in different directions. The letters F, B, R, L and S in the program stand for the numbers which make the robot go forwards, backwards, right, left and stop. You need to experiment to find the right numbers for your computer and motors.

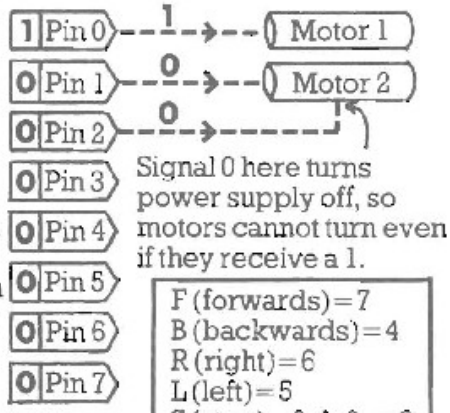
F = 7 (binary 00000111)



L = 5 (binary 00000101)



S = 1 (binary 00000001)



F (forwards) = 7
B (backwards) = 4
R (right) = 6
L (left) = 5
S (stop) = 0, 1, 2 or 3

Here are three examples showing how different numbers make the robot move different ways. Inside the computer the numbers are converted to binary numbers and where there is a 1 in the number, a signal is sent out of the user port from the pin in the same position. Try the numbers shown on the right with your computer by using the command POKE (Address), number.* If they make the robot go in different directions from those listed, swap the numbers around. If you have connected the motors to pins other than 0, 1 and 2 you will need to try different numbers in the range 0 to 255.

```
 10 (Set DDR if necessary)
 20 LET Y=(Address)
 30 POKE Y,F
 40 LET Z=PEEK Y AND 8
 50 IF Z<>0 THEN GOTO 70
 60 GOTO 40
 70 POKE Y,B
 80 FOR I=1 TO 10
 90 GOSUB 190
100 NEXT I
110 POKE Y,S
120 LET P=R
130 IF RND(1)>0.5 THEN LET P=L
140 POKE Y,P
150 FOR I=1 TO INT(RND(1)*20+10)
160 GOSUB 190
170 NEXT I
180 GOTO 30
190 FOR T=1 TO 100
200 NEXT T
210 RETURN
```

Robot moves forwards. (Substitute number for F.)

AND 8 makes computer look at signal on pin 3 in user port, which is connected to wire from bumper circuit.

Waits until switch is pressed (i.e. robot bumps into something) before going to line 70.

Robot goes into reverse. (Substitute number for B.)

Robot reverses while computer goes through delay loops, then stops.

Robot swivels right or left depending on whether random number is greater or less than 0.5. (Substitute numbers for R and L.)

Robot turns through random number of degrees.

Goes back to line 30 to start moving forwards in new direction.

*Set DDR if necessary (see your manual). Set pins 0, 1 and 2 to output, and pin 3 to input. The other pins are not connected to the robot so they do not matter.

How to solder

What you need:

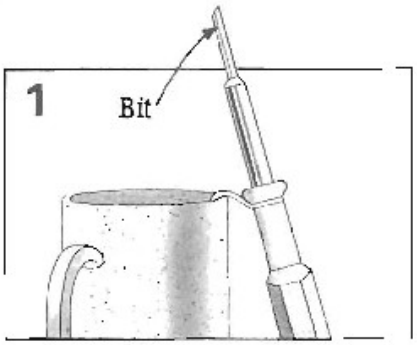
A small soldering iron

some "cored" solder

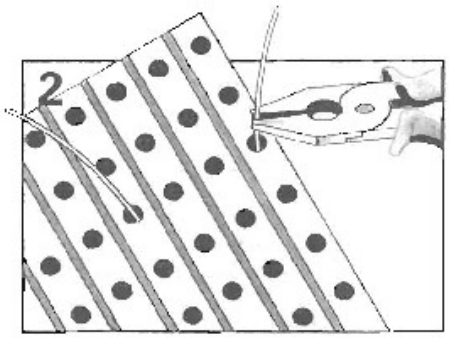
a damp sponge

miniature pliers

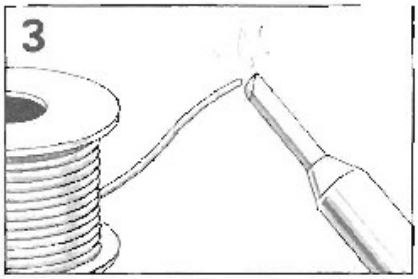
wire cutters



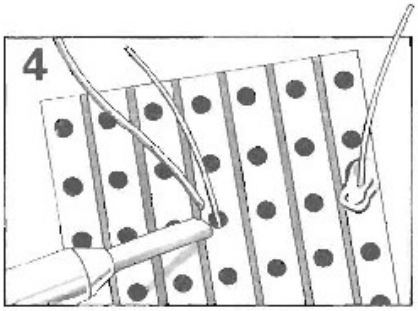
Plug the soldering iron in. While it is heating up, support it so that the bit is not touching anything.



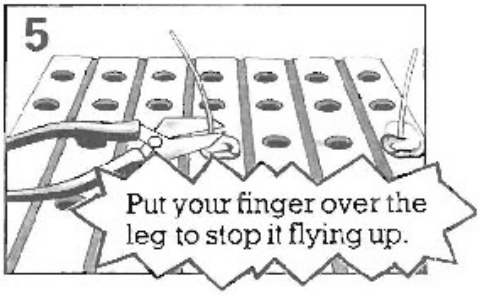
To solder a component on to Veroboard, find the right holes and push the legs through. Bend them out slightly with pliers.



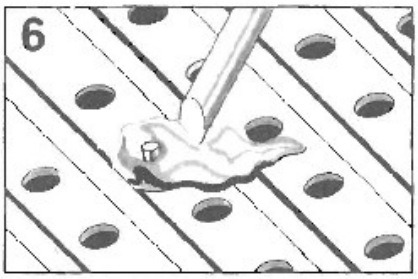
Touch the end of the solder with the hot bit, so that a drop of solder melts and clings to the bit.



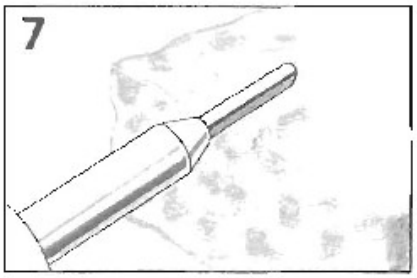
Then touch the leg of the component with the bit and the tip of the solder for a second until a drop of solder joins it to the track.



Let the joint cool for a few seconds. Then, tilt the board away from you and trim the legs close to the soldered joint with wire cutters.



Remove any solder that has run into the groove between the tracks by running the hot bit along the track a few times.



After each joint, clean the bit on the damp sponge and *remember to unplug the soldering iron when you have finished.*

Desoldering

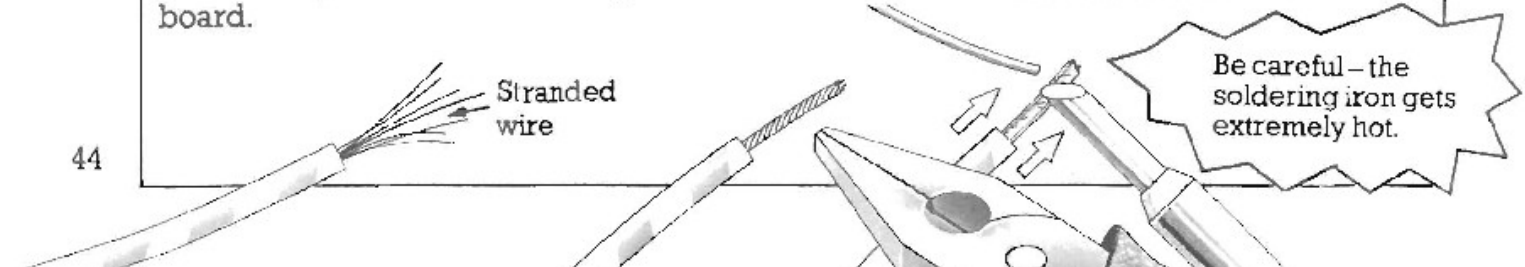
To remove, or desolder, a component, wedge the tip of a pencil between the component's legs on the top side of the Veroboard. Ask someone to tilt the Veroboard and hold the pencil, easing the component out as you melt the joints on the tracks with the soldering iron.

How to tin wire

If you are using stranded wire, it helps to coat (tin) the ends with a layer of solder, making it easier to push them through holes on the board.

Strip about a centimetre of the plastic coating off the wires using wire strippers. Twist the strands of wire together.

Put something heavy on the wire to hold it still. Stroke the twisted strands of wire with the solder and the bit a few times to coat them.



BASIC conversion chart

This chart shows the commands used by different computers for non-standard BASIC words. Check the commands here where you see a ▲ against a program line, and change the command for your computer if necessary.

Commands used in program listings	VIC 20	BBC	ZX81	SPECTRUM	ORIC	TRS-80	DRAGON
CLS	PRINT CHR$(147)	CLS	CLS	CLS	CLS	CLS	CLS
RND(1)	RND(1)	RND(1)	RND	RND	RND(1)	RND(0)	RND(0)
LPRINT	OPEN 4,4 (Turns printer on) CMD 4 PRINT #4,"....." CLOSE 4 — (Turns printer off.)	Press CTRL key and B to turn printer on and CTRL and C to turn it off.	LPRINT	LPRINT	LPRINT	LPRINT	PRINT #−2
ASC	ASC	ASC	CODE Sinclair (Timex) computers use a different set of numbers to ASCII numbers.	CODE	ASC	ASC	ASC
INKEY$(0)	GET A$	INKEY$(0)	INKEY$	INKEY$	KEY$	INKEY$	INKEY$
PRINT TAB (X, Y);	*A$="(Press cursor up/ down key 23 times.)" B$="(Press cursor left/ right key 22 times.)" PRINT "(Press HOME key.)"+LEFT$(A$,Y)+ LEFT$(B$,X);	PRINT TAB(X, Y);	PRINT AT Y,X;	PRINT AT Y,X;	PLOT X,Y,	PRINT @ 64*Y+X;	PRINT @ 32*Y+X;

*VIC 20 has no PRINT TAB(X,Y); command, so use these three lines instead.

Sinclair/Timex program alterations

These alterations apply to both the Spectrum and the ZX81 (Timex 2000 and 1000), unless otherwise stated.

Page 6
Playmates program

```
 30 DIM N$(6,10)
 95 LET L=LEN(X$)
110 IF X$=N$(I)(1 TO L) THEN LET
A=1
```

Line 30 tells the computer that there are six items to store in array N$, and that the longest item consists of ten letters, so the computer sets aside enough space in N$ to store the list. Line 95 is a new line to add to the program.

If you have a ZX81 (Timex 1000), you also need to make the following changes, because it does not use READ and DATA.

Leave out lines 10-20

```
50 INPUT N$(I)
```

When you have typed in the program, run it, and type in each name you want the computer to recognize followed by NEWLINE. To use the program, type GOTO 70 instead of RUN.

Page 8
Smalltalk

```
60 DIM R$(14,20)
65 DIM R(14)
```

The program uses DATA and READ, so for the ZX81 (Timex 1000) make the following changes as well as those above, and use the program in the same way as the Playmates program.

Leave out lines 10-50

```
90 INPUT R$(I)
```

Run the program and type each reply, followed by NEWLINE. Then type GOTO 120 to try out the program.

Page 16
Number codes

```
40 LET X=CODE(X$)+C
```

Letter codes

```
 10 LET Z=CODE("Z")
 90 LET Y$=X$(1 TO 1)
110 LET X=CODE(Y$)
```

Page 17
Letter decoder

```
70 IF I<K THEN LET Z$(I)=
CHR$(CODE("A")+I-K+26)
80 IF I>=K THEN LET Z$(I)=
CHR$(CODE("A")+I-K)
110 LET A$=X$(J TO J)
140 PRINT Z$(CODE(A$)-CODE("A")+1);
```

Second version of letter decoder

```
140 LET C$(J)=Z$(CODE(A$)-
CODE("A")+1)
```

Page 19
Designer

For the ZX81 (Timex 1000), incorporate these lines into the program:

```
300 IF A$<>U$ OR R<=T THEN GOTO 310
305 LET R=R-1
307 GOTO 270
310 IF A$<>D$ OR R>=D THEN GOTO 320
315 LET R=R+1
317 GOTO 270
320 IF A$<>L$ OR C<=L THEN GOTO 330
325 LET C=C-1
327 GOTO 270
330 IF A$<>R$ OR C>=W THEN GOTO 340
335 LET C=C+1
337 GOTO 270
390 PRINT AT R,C;"*"
400 FOR Q=1 TO 5
420 PRINT AT R,C;S$(C,R)
430 FOR Q=1 TO 5
```

Page 10
Quizmaster

Leave out the DATA lines 90-130

```
170 LET X$=Q$(I)
175 LET Y$=A$(I)
200 IF Y$( TO LEN(Z$))<>Z$ THEN
GOTO 210
202 PRINT "CORRECT"
205 LET R=R+1
207 GOTO 220
210 PRINT "WRONG"
```

There is no need to alter this program for the Spectrum (Timex 2000) but you need to make the above changes for the ZX81 (Timex 1000).

```
215 LET W=W+1
280 STOP
300 LET N=5
310 DIM Q$(N,20)
320 DIM A$(N,20)
330 PRINT "PLEASE TYPE IN"
340 FOR I=1 TO N
350 PRINT "QUESTION ";I
360 INPUT Q$(I)
370 PRINT "ANSWER ";I
380 INPUT A$(I)
390 NEXT I
```

When you have typed in the program, type RUN 300. The computer will ask you to type in the question and answer pairs. When you have done this, type GOTO 10 to use the program.

Page 21
Downhill racer game

```
 40 LET R$="B"
 50 LET L$="V"
160 LET A$=INKEY$
210 PRINT AT 21,T;"P";AT 21,C;
"11";AT 21,T+P;"P"
215 SCROLL
315 SCROLL
335 CLS
```

Page 26 – 27

You can convert the following two programs for the ZX81 (Timex 1000) using INPUT as in the Playmates and Smalltalk programs above. If you have a Spectrum (Timex 2000), make these changes:

Horoscope generator

```
220 DIM S$(12,11)
225 DIM S(12)
260 DIM D$(12,44)
```

Computer poet

```
70 DIM W$(31,7)
```

Page 28
Filmsearch

For the ZX81 (Timex 1000), leave out DATA lines 20-100 and line 300. Add or replace the following lines:

```
285 FOR I=1 TO N
290 LET X$=Q$(I)
310 LET Y=Y(I)
330 IF F$=X$( TO LEN(F$)) AND
D=1 THEN GOTO 360
350 NEXT I
355 STOP
370 GOTO 350
500 LET N=11
510 DIM Q$(N,25)
520 DIM Y(N)
530 PRINT "PLEASE TYPE IN "
540 FOR I=1 TO N
550 PRINT "FILM ";I
560 INPUT Q$(I)
570 PRINT "YEAR "
580 INPUT Y(I)
590 NEXT I
```

Then type RUN 500, and type in the names of films and the dates. (You can type in eleven films – you need to change N in line 500 if you want to put in a different number.) To use the program, type GOTO 10.

Page 32
Test program

Here are the program lines to use if you have a BBC or a VIC 20:

```
BBC:     10 ?&FE62=0
         20 LET A=?&FE60
VIC 20:10 POKE 37138,0
         20 LET A=PEEK(37136)
```

Answers

Page 9
Debtors and creditors

You need to put in lines telling the computer not to write a letter if you type in 0 for the amount e.g.

```
55 IF M=0 THEN PRINT "NO DEBTS"
56 IF M=0 THEN GOTO 20
```

Put in lines like these to make the program instructions clearer:

```
45 PRINT "(TYPE IN AMOUNT IN
FIGURES"
46 PRINT "AND CURRENCY IN WORDS."
```

Page 11
French test

This line prints out the correct answer if you get it wrong:

```
212 IF Z$<>Y$ THEN PRINT "THE
ANSWER IS ";Y$
```

Page 25
Coke bottle program

This program will draw a coke bottle before and after inflation:

```
 10 CLS
 20 PRINT
 30 PRINT "HERE IS A COKE BOTTLE"
 40 LET C$="."
 50 FOR R=5 TO 9
 60 FOR C=18 TO 20
 70 GOSUB 350
 80 NEXT C
 90 NEXT R
100 LET C$="%"
110 FOR C=17 TO 21
120 FOR R=10 TO 19
130 GOSUB 350
140 NEXT R
150 NEXT C
160 LET C=0
170 LET R=2
180 LET C$="WHAT IS THE INFLATION
    RATE"
190 GOSUB 350
200 PRINT "(NUMBER ONLY)"
210 INPUT X
220 PRINT "NOW I CAN ONLY BUY THIS
    MUCH"
230 PRINT
240 LET X=100-100/(100+X)*100
250 LET X=X/2
260 FOR R=10 TO 19
270 FOR C=17 TO 21
280 IF X>0 THEN LET C$="."
290 IF X<=0 THEN LET C$="%"
300 GOSUB 350
310 LET X=X-1
320 NEXT C
330 NEXT R
340 STOP
350 PRINT TAB(C,R);C$
360 RETURN
```

Index

First published in 1983 by Usborne Publishing Ltd., 20 Garrick Street, London, WC2E 9BJ, England.

Printed in Spain by Printer Industria Gráfica, S.A. Depósito legal B-27780/83

Usborne Computer Books

Usborne Computer Books are colourful, straightforward and easy-to-understand guides to the world of home computing for beginners of all ages.

Usborne Guide to Computers A colourful introduction to the world of computers. *"Without question the best general introduction to computing I have ever seen."* Personal Computer World

Understanding the Micro A beginner's guide to microcomputers, how to use them and how they work. *"This introduction to the subject seems to get everything right."* Guardian

Computer Programming A simple introduction to BASIC for absolute beginners. *"... lucid and entertaining ..."* Guardian

Computer and Video Games All about electronic games and how they work, with expert's tips on how to win. *"The ideal book to convert the arcade games freak to real computing."* Computing Today

Computer Spacegames, Computer Battlegames Listings to run on the ZX81, Spectrum, BBC, TRS-80, Apple, VIC 20 and PET. *"Highly recommended to anyone of any age."* Computing Today

Practical Things to do with a Microcomputer Lots of programs to run and a robot to build which will work with most micros.

Computer Jargon An illustrated guide to all the jargon.

Computer Graphics Superbly illustrated introduction to computer graphics with programs and a graphics conversion chart for most micros.

Write Your Own Adventure Programs Step-by-step guide to writing adventure games programs, with lots of expert's tips.

Machine Code for Beginners A really simple introduction to machine code for the Z80 and 6502.

Better BASIC A beginner's guide to writing programs in BASIC.

Inside the Chip A simple and colourful account of how the chip works and what it can do.

Published in Canada by Hayes Publishing Ltd, 3312 Mainway, Burlington, Ontario, Canada, L7M 1A7

Published in the USA by EDC PUBLISHING, 8141 E. 44th Street, Tulsa, Oklahoma 74145, USA.

ISBN 0 86020 731 5