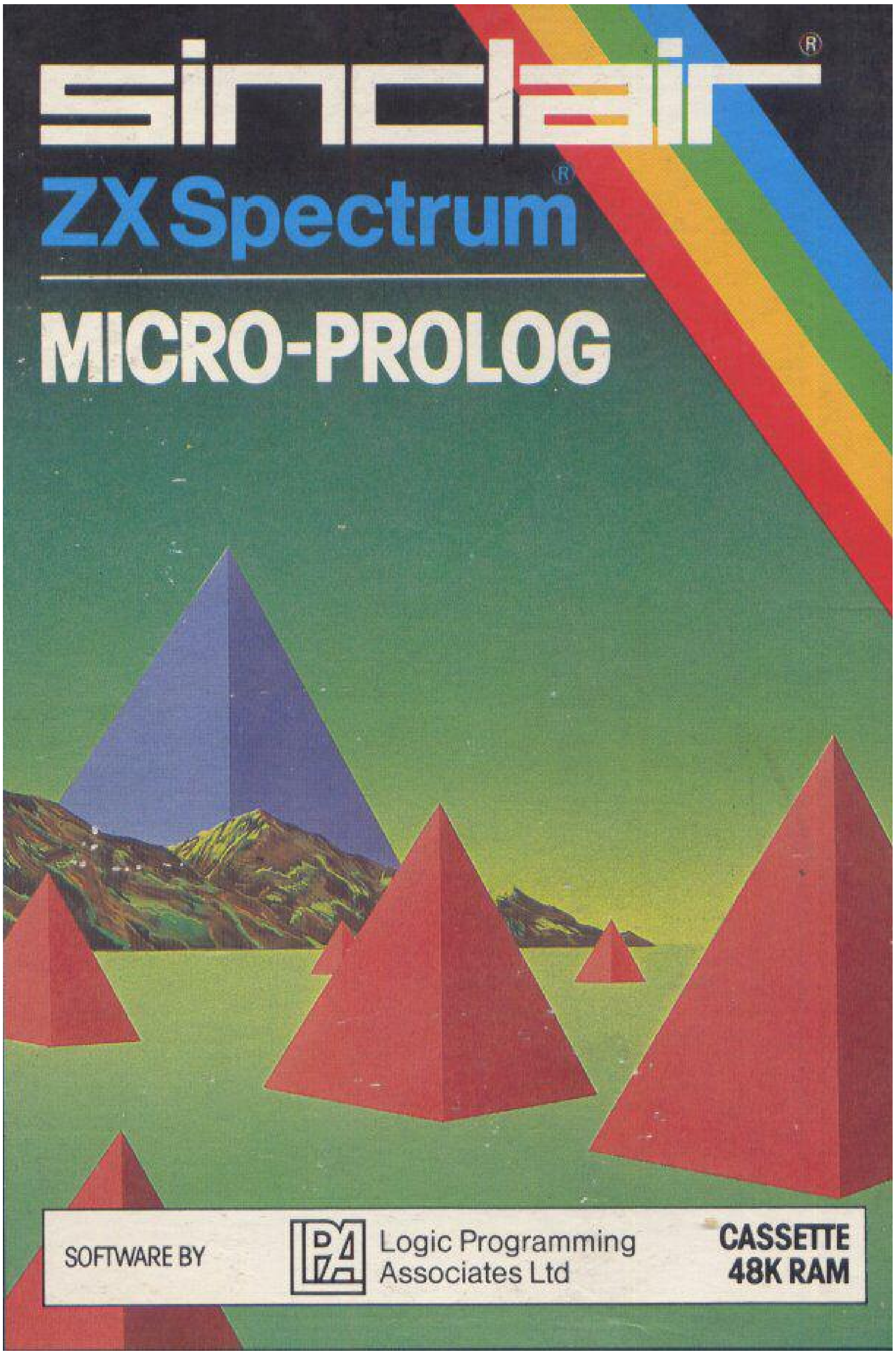

sinclair®
ZX Spectrum®
MICRO-PROLOG
SOFTWARE BY
LPA Logic Programming Associates Ltd
CASSETTE
48K RAM

Introduction to Micro-PROLOG For the Sinclair ZX Spectrum

Jonathan Briggs
Department of Computing
Imperial College
London SW7

First published in 1983 by
Sinclair Research Limited
25 Willis Road Cambridge CB1 2AQ England.

ISBN 085016 008 2

Printed in UK

Introduction

This booklet is designed to get you started with a new computer language for your ZX Spectrum. By the time you have worked through the examples in it you will have learned one way of using the language micro-PROLOG. You will then be ready to start reading the ***micro-PROLOG Primer***.

This booklet also introduces the main differences between this and other versions of micro-PROLOG and so it is worth reading even if you have used the language on another computer.

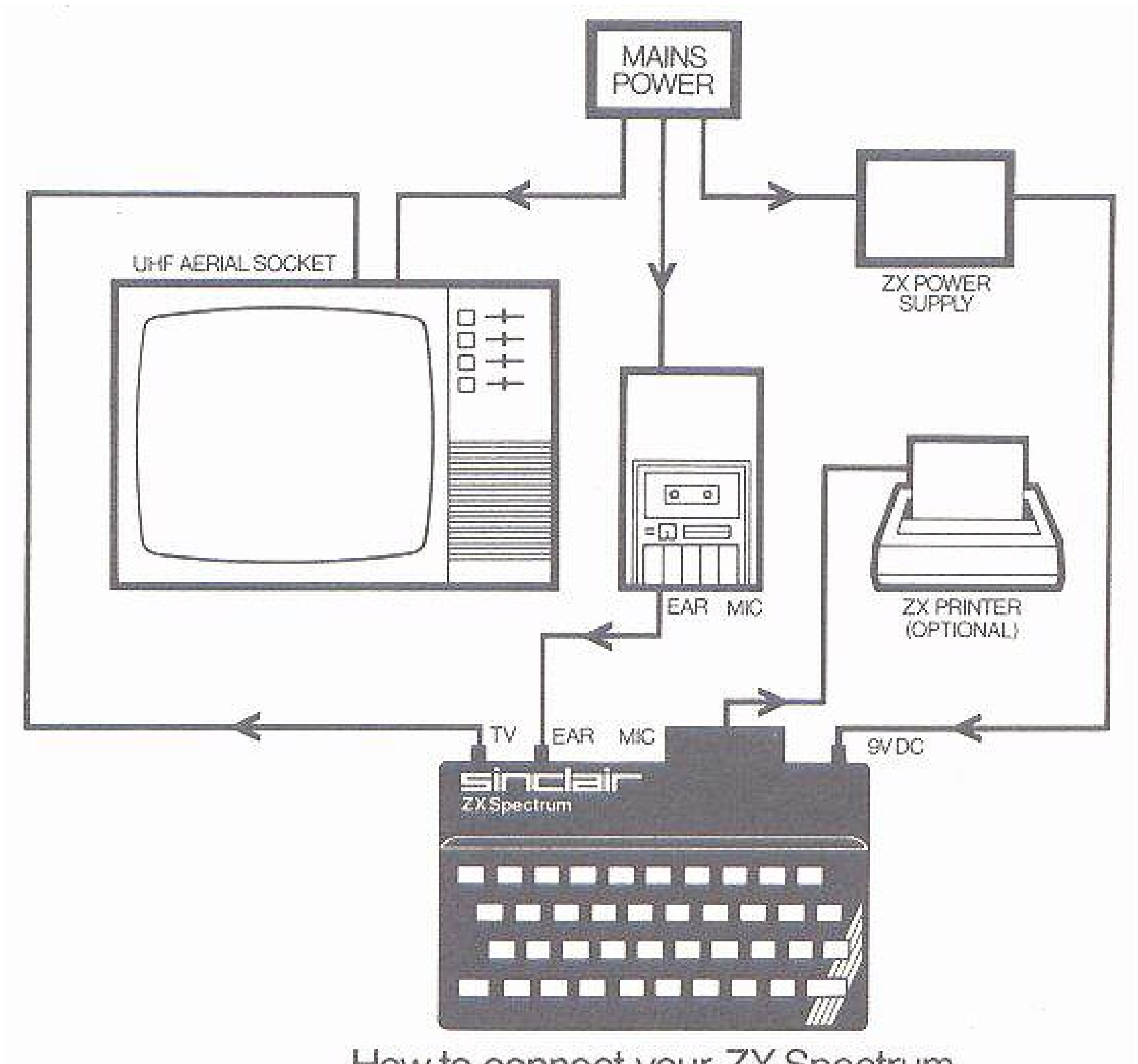
If you own a ZX Spectrum, the chances are that you have some knowledge of the computer language BASIC. micro-PROLOG is a completely different language; however expert or otherwise your BASIC programming, you will really be starting from scratch.

The micro-PROLOG package consists of a cassette tape with micro-PROLOG recorded on it, this introductory booklet and a copy of the ***micro-PROLOG Primer***. A number of micro-PROLOG programs are also recorded on the tape including the SIMPLE front-end program that is used and described in this booklet. These programs are listed in the Appendix.

What you need to start

To use this cassette you will need a 48K ZX Spectrum, a cassette recorder/player, a UHF television (a colour TV will allow you to experiment with colour graphics, but a black and white TV will work just as well) and the cassette, lead, power supply and aerial lead that accompanied your ZX Spectrum when you brought it.

Connect the components of your computer system as follows:

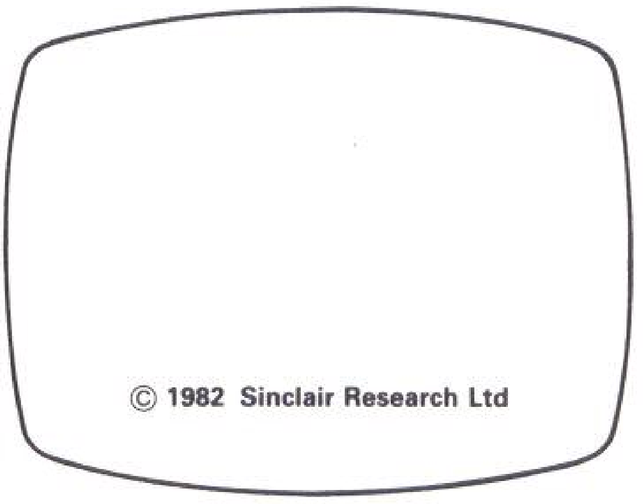


How to connect your ZX Spectrum

Loading micro-PROLOG

When you switch on your computer it understands BASIC. In order for it to understand micro-PROLOG instead, the new language must be loaded into the machine's memory.

If you have connected everything properly, the screen should show this:



You must now type the command to load the language:

LOAD "PROLOG"

A system of keywords is used when the computer is using BASIC. LOAD is a BASIC keyword. You get it by pressing the **J** key. Only by using this keyword can a program be LOADed from cassette.

" is on the **P** key in red. To obtain this symbol you must hold down the red **SYMBOL SHIFT** key and then press **P**.

P R 0 L 0 G is typed out using the ordinary typewriter action of the keyboard.

Further information about using your keyboard in BASIC is to be found in the two books that accompanied your Spectrum when you bought it.

Using the keyboard when micro-PROLOG has been loaded is slightly different. It is described in the next section.

(There is another way of loading PROLOG. If you type **LOAD** "" your Spectrum will load the first program on the tape. Make sure that the tape is fully rewound before loading in this way.)

One you have typed:

LOAD "PROLOG"

*Place the micro-PROLOG cassette into the cassette player
*Make sure it is on side A and fully rewound
*Set the volume control to approximately half way
*Set the tone control (if fitted) to half way
*Press the **ENTER** key on your Spectrum
*Press the **PLAY** key on your cassette player.

If the controls on your cassette player are properly set a pattern of (coloured) horizontal stripes will appear on the border of the television screen.

*Firstly red and light blue horizontal stripes about 1cm wide moving slowly upwards-this is the lead in.

*Secondly much thinner yellow and dark blue stripes-this is the start of the data.

micro-PROLOG is loaded in several sections: this sequence of lead in followed by data is repeated several times.

If everything is set correctly, loading micro-PROLOG should take about one minute. When the computer has finished loading the flashing stripes will disappear and a copyright message will be displayed as follows:

```
Spectrum micro-PROLOG T1.0
© 1983 LPA Ltd.
xxxx
&.L
```

You are now ready to load the SIMPLE front-end program.

If micro-PROLOG did not load

1 Check that the cassette leads are correctly connected (EAR to EAR-same colour plugs)
2 If no striped pattern appeared on the screen then the computer is not picking up information from the tape. Try increasing the volume slightly, rewind the tape and try loading again.
3 If R Tape Loading Error appears, the chances are that the volume or tone levels are too high and the signal is becoming distorted. Try reducing them and loading them again.
4 If you still have problems then try loading another cassette-the HORIZONS cassette supplied with your ZX Spectrum for example-to check that your Spectrum and cassette player are compatible. Some cassette players, particularly old ones, don't play back at a constant speed. If you are using a battery-powered cassette unit, check that the batteries are not worn out.

Once micro-PROLOG has been loaded, it is a good idea to make a note of the volume and tone settings, so that you can repeat the loading process easily.

The SIMPLE system

micro-PROLOG is an extremely flexible language. It is possible, using it, to write many different sorts of program. One program supplied on the micro-PROLOG cassette actually changes the way we use the language. This program is called SIMPLE; it makes micro-PROLOG simpler to understand. We will learn to use the language using SIMPLE.

SIMPLE is recorded on the cassette immediately after micro-PROLOG. It must be loaded.

You should see the symbols **&.** on the left hand side of the screen under the copyright message. Next to this is the flashing L cursor. A cursor is used to indicate whereabouts on the screen the characters you type in will appear. On the Spectrum, the cursor also indicates what sort of character will be produced by a particular key.

You will notice if you press **CAPS LOCK (CAPS SHIFT** together with 2) the L cursor changes to a C which means that everything you type following this will be in capital letters. Pressing **CAPS LOCK** again will return you to the normal lower case keyboard and the flashing **L** cursor.

The combination of **&.L** is call a prompt and means that micro-PROLOG is expecting you to type something. Type:

LOAD SIMPLE

L 0 A D and **S I M P L E** are typed using the ordinary typewriter action of the keyboard. Hold down the **CAPS SHIFT** key to obtain capital letters.

If you make a mistake and want to correct it you can use the **DELETE** key (**CAPS SHIFT** together with **0**). This will delete the last character to the left of the flashing cursor.

The press **ENTER**, and start your cassette player. After a few seconds the patterns of striped lines should reappear on the border of the screen. SIMPLE is now being loaded. It is made up of a large number of small blocks of data and takes about one minute to load.

As each block is loaded, micro-PROLOG checks to see that it has loaded correctly and types out:

```
SIMPLE 01   BLOCK OK
```

This message is made up of the filename, followed by the block number being read. If, for any reason, the computer is unable to understand a block, instead of the BLOCK OK message the computer will display:

```
SIMPLE 07   READ ERROR
```

Should this happen, rewind the tape a little and the computer will try and load that block again. It any other sort of error occurs, the most sensible course of action is to start loading SIMPLE again. Once you have read the Primer you will be able to understand and cure most errors. Until then, to restart micro-PROLOG type:

NEW. (Make sure you remember the full stop)

This will clear away any parts of SIMPLE that may have been loaded. Rewind the tape and try loading SIMPLE again.

When micro-PROLOG has finished loading, you will see the micro-PROLOG prompt **&.** Switch off your recorder; SIMPLE has been loaded and the computer is now ready to accept your next input.

The keyboard in micro-PROLOG

The ZX Spectrum continues the ZX81 tradition of putting a complete BASIC keyword on each key. You can 'type' **INPUT**, **RETURN**, **GOSUB** etc with one key stroke.

micro-PROLOG does not have a system of fixed keywords; the flexibility of the language allows users to define their own keywords if they want to. When you loaded SIMPLE a set of keywords was defined. We will learn to use these words in the next section.

Before that, it is important that you should be familiar with the keyboard. As you work through this section you can try typing anything you like. The computer may try to understand it and fail - giving you an error - or you can rub it out using the **DELETE** key. If, at any time, you have typed screenfuls of meaningless characters, you can return to the **&.** prompt by pressing **BREAK** (the **SYMBOL SHIFT** and **BREAK** keys together). This will prevent the computer from bothering to look at what you have typed. The message that the computer types out when you type **BREAK** simply tells you what the computer was doing when you interrupted it.

Upper and lower case letters

For most purposes, ordinary upper and lower case letters will be sufficient. These are typed using the standard typewriter action of the keyboard. **CAPS SHIFT** pressed, followed by the letter required, is a capital. Try typing something using capitals.

Often, when typing in a program or text, you may wish a section of it all to be in capital letters. As on a typewriter, the ZX Spectrum keyboard provides a CAPS LOCK key. If you press **CAPS LOCK** (**CAPS SHIFT** together with **2**) every letter that you type after this until you press **CAPS LOCK** again will be in capitals. You will notice that the Spectrum indicates that it is in **CAPS LOCK** 'mode' by changing the flashing **L** cursor to a flashing **C**.

Punctuation and arithmetic symbols

The other main symbols that you will need are the punctuation and arithmetic symbols; particularly (and). Most of these symbols appear on the keys in red. To obtain them, press the red **SYMBOL SHIFT** key together with the symbol required. You will notice that for keys on which a BASIC keyword is printed in red, a question mark ? is printed to indicate that you have typed something that micro-PROLOG does not understand.

There are a few symbols printed below keys in red. One of these is particularly important to micro-PROLOG - | called 'bar' - and several of the others might be useful. These are typed in 'extended mode', by pressing **CAPS SHIFT** and **SYMBOL SHIFT** together. You will notice that the flashing **L** cursor - expecting a letter, number or symbol -changes to an **E** to indicate that you are now in extended mode. To obtain the red (below key) symbols you must now press **SYMBOL SHIFT** together with the key you require.

delete

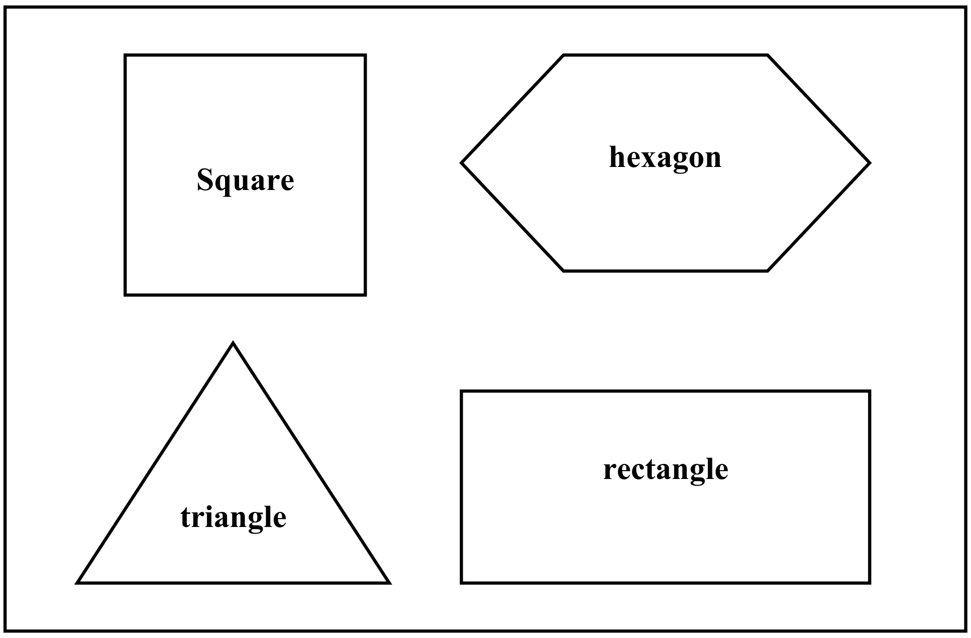
To rub out characters on the line the **DELETE** key is used. This is on the same key as **0. CAPS SHIFT** must be held down at the same time.

The left and right cursor keys (⇦ on key **5** and ⇨ on key **8**) allow you to move the cursor within the current line. The **CAPS SHIFT** must be depressed at the same time. By positioning the cursor in the correct place, incorrect characters can be deleted and text inserted.

Using SIMPLE-learning to describe things

Most computer languages consist of sequences of instructions to the computer. PROLOG is different, as it works with descriptions of everyday concepts and ideas.

To understand this idea of description, let's start by describing a picture consisting of four simple geometric shapes.



If we were describing this picture to a friend we would have to explain the different positions for each of the shapes in the picture. We might say that the square, for example, is above the triangle.

We can tell the computer this information using SIMPLE micro-PROLOG if we present it in a form that the computer will understand:

```
square    above    triangle
```

We separate the fact into three parts. We have two objects, square and triangle, and a relationship that connects them: above.

If we are prepared to translate our descriptions into this sort of form, micro-PROLOG will allow us to continue to use ordinary English words. Let's tell the computer this fact; this piece of description about-our picture.

First check that the computer shows the **&.** prompt. This indicates that micro-PROLOG is ready to do something. If it is not there, type **BREAK (SYMBOL SHIFT** and **BREAK**). This is one way of telling the computer to stop what it's doing and prepare to do something else.

NOTE-The keywords we will learn in this section are part of the SIMPLE front-end program. To use them we must have SIMPLE loaded. If you wish to use micro-PROLOG without SIMPLE you should consult the ***micro-PROLOG Primer***.

add

We will use keyword **add** to indicate that we want the computer to learn something new.

Let's add our first sentence:

add (square above triangle)

We need the keyword **add**, then a left bracket, then our sentence and finally a right bracket. Notice that you can put in several spaces between words if this makes it clearer. Press **ENTER**.

The computer stores this fact. To see all the information that it knows at any time we can type:

list all (and press **ENTER**)

The computer will respond:

```
square above triangle
&.
```

Let us add some more information about our picture.

NOTE-AS you add this information you may make some mistakes. Turn to the section on editing, deleting and killing to correct them.

While you are adding one of these sentences don't type the closing bracket; press **ENTER**. You will notice that instead of the **&.** prompt a **1.** prompt appears at the start of the line. Don't panic! micro-PROLOG is very tolerant; it allows you to enter a sentence over several lines. The **1.** indicates that the computer is expecting one closing bracket (while a **2.** would mean two closing brackets, and so on). The computer will not return to the **&.** prompt until you have typed).

```
&. add (hexagon above rectangle)
&. add (hexagon right-of square)
&. add (rectangle right-of triangle)
&.
```

In the first sentence we added, we used a relationship **above** to link the two shapes. In these sentences we have introduced a new relationship **right-of.**

Now, let's see what we have told the computer so far:

```
&. list all (remember to follow each command with ENTER)
square above triangle
hexagon above rectangle
hexagon right-of square
rectangle right-of triangle
&.
```

We can also list sentences in terms of their relationships:

```
&. list right-of
hexagon right-of square
rectangle right-of triangle
&.
```

Teaching the computer things by describing information in this sort of way is known as building a database. A database is a collection of data; a set of information.

Using a database-asking questions

Information becomes useful only when we are able to use it. An encyclopaedia is only of use when we consult it to answer questions or confirm knowledge that we possess.

The information stored in a micro-PROLOG database becomes useful when we start asking questions.

Without looking back at the picture, can you remember the positions for each of the shapes? Was the square above the triangle or the other way round? Perhaps this picture is too easy, but what if there had been fifty shapes? Would you have been able to remember all the relative positions of the shapes?

We are often able to answer complex questions by using information stored in a computer. Once we have constructed our database, we can ask micro-PROLOG questions.

is-confirming our hunches

The first sort of question we can ask is that answered with a simple YES or NO. Was the square above the triangle? Was the triangle to the right of the hexagon?

We ask these questions using a SIMPLE keyword is. The question 'Was the square above the triangle?' is phrased in this way:

```
is (square above triangle)
```

If we type this into the computer the computer will respond YES.

It is important to ask the question correctly. The words used - the vocabulary- must be the same as those used when the information was added. If, for example, you try to ask:

```
is (square on-top-of triangle)
```

the computer will respond:

```
Evaluation error 2: No definition for relation
trying on-top-of (square triangle)
```

indicating that it does not understand the words on-top-of.

To see what relationships the computer has been told about, you can type:

```
&.list diet
```

For the database we have constructed so far the computer will respond:

```
above diet
right-of diet
&.
```

These words are a dictionary of relationships understood by the computer. Use these words when asking questions.

Let's ask some more questions.

```
&.is (hexagon above rectangle)
YES
&.is (hexagon right-of square)
YES
```

```
&.is (hexagon above hexagon)
NO
&.is (circle above octagon)
NO
```

The computer will answer questions using only the information available. A fact may be perfectly true in the outside world but unless the computer has been told it, the answer will be NO.

If you don't get the answers you expect, check that you are using the same vocabulary. micro-PROLOG treats small and capital letters differently; hexagon is not the same as Hexagon.

If we want, we can ask a more complex question such as:

```
&.is (hexagon right-of square and
hexagon above rectangle)
YES
```

The computer answers YES only if all parts of the question are true; otherwise it will answer NO.

Notice that in English we sometimes leave words out of complex questions. We might have tried to ask:

```
is (hexagon right-of square and
above rectangle)
```

This is incorrect micro-PROLOG. A complex question is always made up of smaller questions, each of which must be complete.

is can also be used to ask another sort of question: 'Is there something above the triangle?' In such questions the words 'something' and 'anything' stand for things we don't know. If the hexagon is to the right of the square, the 'anything' in the first question could refer to the square. If the square is above the triangle then the square could be the 'something' in the second question.

In micro-PROLOG we use the letters **x, y** and **z** (also **X, Y, Z, x1, X1** etc) to stand for the unknown values.

Let us ask these questions in SIMPLE micro-PROLOG.

```
&.is (hexagon right-of x)
YES
&.is (x above triangle)
YES
```

The **x** in these questions is used in a similar way to the words 'something' or 'anything' in the English questions.

We will now see another use of this technique of allowing 'variables' to stand for part of our questions.

which-selecting Information

Most questions we ask in English are not answered with a simple YES or NO. What is the quickest route to the station? How much do you weigh?

These questions are answered with detailed information. In micro-PROLOG we often want more information from a question. We use another SIMPLE keyword, **which.**

Suppose we wanted to find which shape is above the rectangle using our existing database.

We ask:

```
&.which (x  :  x above rectangle)
```

The computer will respond:

```
hexagon

No (more) answers
```

Let's examine this question carefully.

```
Which (x            :      x    above rectangle)
        ↑                  ↑
   answer pattern     information selector
```

The information selector looks very similar to the last sort of is question we looked at. The x stands for 'anything' the computer can find that 'is above rectangle'. The answer pattern tells the computer what you want printed; in this case, the value of **x.**

Let's see some more **which** questions:

```
&.which (x  :  hexagon right-of x)
square
No (more) answers

&.which (x  :  x above square)
No (more) answers
```

In this example, nothing is above square and so no answers are printed.

```
&.which (x  :  x above y)
square
hexagon
No (more) answers
```

Often more than one answer will be found. In this example, the **y** stands for another unknown. You can read this question in English as 'Which shape is above another shape?'

As with **is** questions, we can use a complex set of conditions to select the information we require.

```
&.which (x : x above rectangle and
x right-of square)
hexagon
No (more) answers
```

We have seen that the first part of the question is used to indicate what we want printed. This can be more than just a single value.

```
&.which (x y: x above y)
square triangle
hexagon rectangle
No (more) answers
```

In this question we are looking for pairs of unknown shapes such that one is above the other.

Text can also be included in what we want printed.

```
&.which (SHAPE x : y right-of x)
SHAPE square
SHAPE triangle
No (more) answers
```

The most common mistake when using the **which** question is to forget that every **which** question must have both an answer pattern and the information selector.

Many people try and type:

```
&.which (x above y)
```

instead of:

```
&.which (x   :   x above y)
```

The first of these questions produces an error message because no answer pattern has been included. The second, which includes the answer pattern x, produces:

```
square
hexagon
No (more) answers
```

Rules-extending the power of the database

So far we have learned how to add information, list it and ask questions. We are now ready to meet the most powerful idea in micro-PROLOG programming; that of rules.

Let's list the database so far:

```
&.list all
square above triangle
hexagon above rectangle
hexagon right-of square
rectangle right-of triangle
```

Which shape is below the square?

We could already obtain the correct answer from the computer using the question:

```
&.which (x   :   square above x)
triangle
No (more) answers
```

We can phrase the question in this way because we know that 'below' and 'above' are opposites. The computer doesn't know about below, but we could tell it by adding new sentences.

```
&.add (triangle below square)
&.add (rectangle below hexagon)
```

This merely duplicates information already contained in the database. We could, instead, define below in terms of above as a general rule.

We know that if x is above y then y must be below x. Let's tell the computer.

```
&.add (x  below y   if   y above x)
```

We add rules in exactly the same way as facts. Once again we are using variables (x and y) to stand for values that are unknown. The rule can be listed using the list command.

```
&. list below
X  below  Y  if  Y  above  X
```

We can use the rule below in exactly the same way as the facts about is-above.

```
&.is (square below triangle)
NO
&.which (x : x below hexagon)
rectangle
No (more) answers
```

Let's add another rule

```
&.add (x  left-of  y  if  y  right-of  x)
```

Try using the complete database by asking questions that include this new rule left-of.

We have now met many of the ideas of micro-PROLOG programming using the SIMPLE front-end system. The ***micro-PROLOG Primer*** includes a much more comprehensive look at these ideas and takes things much further.

editing, deleting and killing

(or changing your mind or your mistakes)

Often when you are developing a program you will make mistakes, mistype a word or wish to use a different word to describe a relation.

One way of removing incorrect sentences is to use the command **delete**. First, list the sentences that include the one you wish to delete.

```
&.list above
square above triangle
hexagon above rectangle
```

Suppose we wish to remove the second sentence, the one with the incorrect spelling of rectangle. We type:

```
&.delete above 2
&.
```

A more radical approach to deleting information is to dispose of an entire set of sentences. We use the word kill.

```
&.kill above
```

would remove all the sentences about above, and respond:

```
Definition for above deleted
```

The ultimate deleting operation is removing everything that you have added using the command **kill all.** The computer checks that you really mean everything.

```
&.kill all
Entire program yes/no? yes
Entire program deleted
&.
```

Usually, it is unnecessary to **delete** or **kill** sentences in order to change them. We can use **edit** to allow us to go back and alter an incorrect word.

```
&.edit above 1
1 (square above triangle)
```

The 1 indicates the position of the sentence in the list of sentences for above. You can alter the order of the sentences by changing this number.

Use the cursor controls (⇦ (**CAPS SHIFT** with **5**) and ⇨ (**CAPS SHIFT** with **8**)) and the delete key (**CAPS SHIFT** with **0**) to correct the line. Characters can also be inserted by using the cursor controls and then typing the new text. When you are satisfied with the corrections press **ENTER**.

saving and loading your programs

You can save the work you have done at any stage in the development of a micro-PROLOG program by recording the program on to the cassette.

Before recording a program, check that the mic input socket on your Spectrum is connected to the mic socket on your recorder using the same colour plugs. The other lead connecting **ear** to **ear** must not be connected. If it is you will not be able to record properly.

To save the database you have created so far type:

save shapes (and press ENTER)
The computer will respond with:
start tape for recording
Hit **ENTER** when ready

When you press ENTER, the program will be recorded on to the tape. As this happens, you will see the same sorts of patterns as those which appeared when you loaded micro-PROLOG and SIMPLE, and each block number will be displayed. The computer will respond with the &. prompt when the save is complete.

shapes is a file name-you can use any name made up of letters and numbers.

Note that the micro-PROLOG command **SAVE** can be used. This differs from **save** (part of SIMPLE) in that no 'ready' message is printed.

loading

Loading your own programs is performed in exactly the same way as loading SIMPLE except that you use type lower case load. Type:

load shapes (and press **ENTER**)

Remember to rewind the tape to before the start of your program. Check that the leads are correctly connected (**ear to ear**). Press **ENTER** on your computer followed by **PLAY** on your cassette recorder. Use the volume/tone settings that you found worked when loading micro-PROLOG and SIMPLE. It is a very good idea to save your programs regularly as you develop them. It may take a few minutes but will save you hours of retyping.

Appendix

Here is a list of the programs you will find on your Micro-PROLOG cassette.

Apart from PROLOG itself and the SIMPLE front-end program described in this booklet, there is an alternative front-end, MICRO, plus a series of utilities designed to assist more advanced users of micro-PROLOG. The purpose and method of use of each program is documented in the micro-PROLOG Reference Manual. For your convenience, programs which are usable with both the SIMPLE and MICRO front-end programs are recorded on both sides of the tape.

Side A		**Side B**	
Tape counter (approx)	Program name	Tape counter (approx)	Program name
000	PROLOG	000	PROLOG
040	SIMPLE	040	MICRO
075	SIMTRACE	075	EXPTRAN
100	EXPTRAN	100	TOLD
110	TOLD	110	MICSHOW
125	SIMSHOW	125	EDITOR
135	PROGRAM	135	MODULES
150	DEFTRAP	150	TRACE
165	SPYTRACE	165	SPYTRACE
180	ERRTRAP	180	ERRTRAP

Summary of micro-PROLOG keyboard

To obtain small letters eg a b c	Press the letter concerned
capital letters eg A B C	**CAPS SHIFT** with letter
CAPS LOCK/UNLOCK for all letters in capitals	**CAPS SHIFT** with **2**
numbers eg 1 2 3	Press number
red symbols on keys eg ().*	**SYMBOL SHIFT** with symbol
red symbols below keys eg | [ ] ©	**SYMBOL SHIFT** with **CAPS SHIFT** followed by **SYMBOL SHIFT** with symbol
DELETE	**CAPS SHIFT** with **?**
⇦ (left cursor)	**CAPS SHIFT** with **5**
⇨ (right cursor)	**CAPS SHIFT** with **8**

One other change has been made to the keyboard. To obtain **BREAK** - ie to stop the computer running a program - Press **SYMBOL SHIFT** with **BREAK**.