

Put your Sinclair in Touch with the Real World!

Out there, there's a whole lot of hardware your computer never knew about ... lights, switches, joysticks and lots more that your computer would just love to touch.

But how?

This book shows you all you need to know as an introduction to hardware design and microcomputer interfacing, on a budget.

It takes you step-by-step from nervously switching on the soldering iron, through to challenging projects such as controlling lights, switches and simple video games. Through these projects, machine code programs are written which control the external hardware. No prior knowledge of machine code is required as each step is explained.

For a full catalogue of top-flight computer books, contact:

Sigma Technical Press
5 Alton Road
Wilmslow
Cheshire SK9 5DY

£5.95

ISBN 0 905104 64 1

SINCLAIR
Spectrum & ZX81

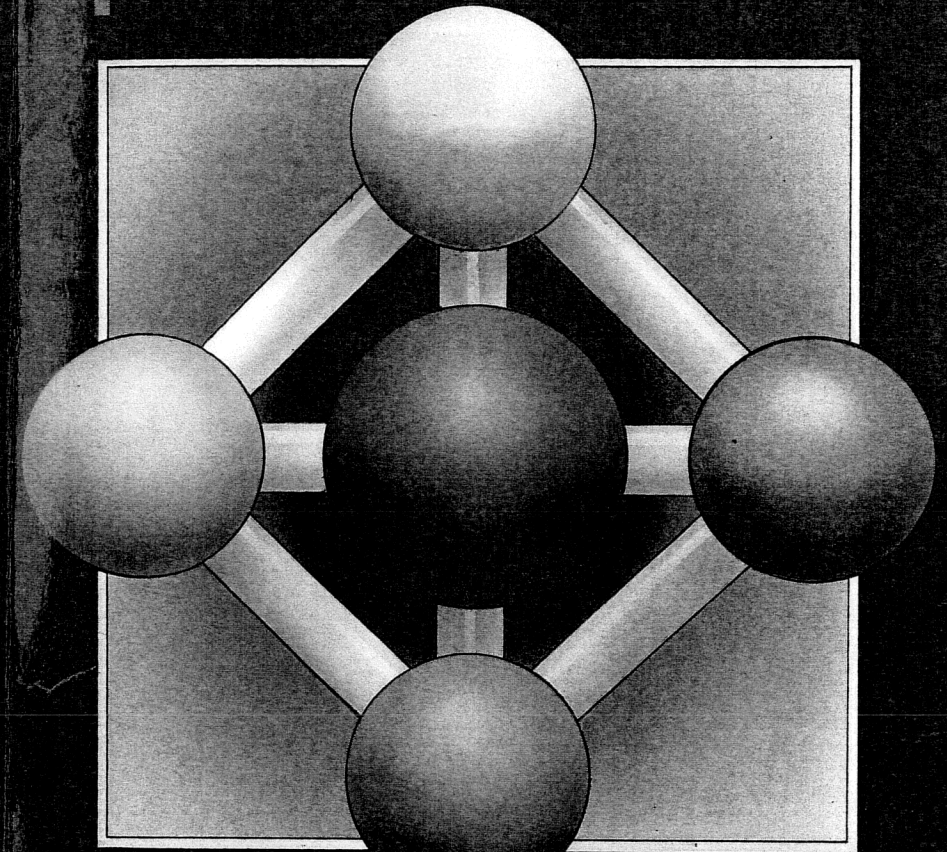
ADD-ONS

MICROCOMPUTER
HARDWARE
PROJECTS

Natasha Graham
Michael Roberts

SINCLAIR

Spectrum & ZX81



ADD-ONS

MICROCOMPUTER HARDWARE PROJECTS

FH
7940

Technical Press

Natasha Graham
Michael Roberts

Microcomputer Hardware Projects:

Sinclair Spectrum and
ZX81 Add-On Units

UNIVERSITÄTSBIBLIOTHEK
HANNOVER
TECHNISCHE
INFORMATIONSBIBLIOTHEK

 Sigma Technical Press

UB/TIB Hannover 89
100 626 734



FH 7940

18523

Copyright © 1983 by Thurnall Electronics

All Rights Reserved

No part of this book may be reproduced or transmitted by any means without the prior permission of the publisher. The only exceptions are for the purposes of review, or as provided for by the Copyright (Photocopying) Act or in order to enter the programs herein onto a computer for the sole use of the purchaser of this book.

ISBN 0 - 905104 - 64 - 1

FH 7940

Typesetting and Production by:

Designed Publications Ltd.
8-10 Trafford Road,
Alderley Edge, Cheshire.

Published by:

SIGMA TECHNICAL PRESS,
5 Alton Road,
Wilmslow,
Cheshire.
UK.

Distributors:

Eruope, Africa:
JOHN WILEY & SONS LIMITED,
Baffins Lane, Chichester,
West Sussex, England.

Australia, New Zealand, South-East Asia:
Jacaranda-Wiley Ltd., Jacaranda Press,
JOHN WILEY & SONS INC.,
GPO Box 859, Brisbane,
Queensland 40001, Australia.

Printed and bound in Great Britain
by Billings & Sons Limited, Worcester.

CONTENTS

Introduction	1
Chapter 1	
Controlling a Light Display	5
Getting Started	6
Preliminary Program for the ZX81	8
Preliminary Program for the SPECTRUM	9
Project 1: Binary Numbers	10
The Program	10
Using the Program	11
Project 2: Making a Metronome	15
The Program	15
Using the Program	16
Project 3: Disco Light Show - Micro Version!	17
The Program	17
Project 4: An Electronic Die	19
The Program	19
Using the Program	20
Project 5: A Binary Counter	21
The Program	21
Using the Program	22

Project 6: An Electronic Timer	23
The Program	23
Using the Program	24
Chapter 2	
Reading a Set of Switches into the Computer	25
Getting Started	26
Preliminary Program for the ZX81	28
Preliminary Program for the SPECTRUM	30
Project 1: Reading the Switches into the Computer	31
The Program	31
Using the Program	31
Project 2: A Combination Lock	33
The Program	33
Using the Program	34
Project 3: Time Your Reactions	35
The Program	35
Using the Program	36
Project 4: Binary Speed Game	37
The Program	37
Using the Program	38
Project 5: The Choice Maker	40
The Program	40
Using the Program	41
Chapter 3	
Using a Joystick	43
Getting Started	44
Preliminary Program for the ZX81	45
Preliminary Program for the SPECTRUM	46
Notes on the Projects in this Chapter	48
Project 1: Drawing Pictures on the Screen	50
The Program	50
Using the Program	51
Project 2: Fast on the Draw	52
The Program	52
Using the Program	53

Project 3: Your Name in Lights	55
The Program	55
Using the Program	56
Project 4: Pack the Blob	57
The Program	57
Using the Program	58
Chapter 4	
Controlling a Set of Switches	59
Getting Started	60
Preliminary Program for the ZX81	62
Preliminary Program for the SPECTRUM	63
Project 1: Switching a Torch On and Off	65
The Program	65
Using the Program	65
Project 2: Controlling Several Lights	67
The Program	67
Project 3: An Alarm Clock	69
The Program	69
Using the Program	70
Chapter 5	
Using a Transistor Driver	71
Preliminary Program for the ZX81	73
Preliminary Program for the SPECTRUM	75
Examples:	
Example 1: Controlling Eight Light-Emitting Diodes	76
The Program	76
Example 2: A Square-Wave Generator	77
The Program	77

Chapter 6

Controlling Several Channels Simultaneously	79
Getting Started	81
Preliminary Program for the ZX81	81
Preliminary Program for the SPECTRUM	81
Examples :	
Example 1: Input and Output	85
The Program	85
Example 2: Sixteen Christmas Tree Lights	86
The Program	86
Example 3: A Game of Chase	88
The Program	88
Example 4: Analogue to Digital Converter	90
The Program	91

Appendix 1

Machine Code	
The Key Steps in the Machine Code Programs	
Program to Output Numbers	

Appendix 2

Decimal and hexadecimal

INTRODUCTION

This book illustrates for Sinclair Spectrum and ZX81 owners the use of different add-ons for their computers. Circuit diagrams are given for those who wish to build their own add-ons; they can also be bought from Thurnall Electronics of Manchester *

Each chapter is devoted to one add-on and is designed to be self-contained. The reader can select which parts of the book are relevant to his or her needs. For each unit a variety of examples is given, with appropriate programs: some simple, some more advanced. These are meant to illustrate to the beginner what the particular add-on can do, and, for the more ambitious, they provide a stepping-stone to more complicated programs. The book is thus an introductory guide which also explores techniques that a reader will acquire when graduating to more advanced work.

Before using any add-on, it is necessary to set up a mechanism by which the computer can communicate with the add-on. There is a standard chip called an 'Input-output controller' which fulfils this task. This is connected to the outside of the Spectrum or ZX81, between the computer and the add-on. Every project in this book requires the use of such an input-output controller. At the end of this introduction we give the circuit diagrams for this unit.

All the projects in this book use programs which are self-contained and easy to enter. Some 'machine code' is necessarily included to instruct the input-output controller, but an understanding of this code is not essential to carry out the projects. However, for those interested, the code is explained in the Appendix.

* Full details and prices from:
Thurnall Electronics
95a Liverpool Road
Cadishead
Manchester M30 5BG
(Telephone: 061 775 4461)

Microcomputer Hardware Projects:

Sinclair Spectrum and
ZX81 Add-On Units

UNIVERSITÄTSBIBLIOTHEK
HANNOVER
TECHNISCHE
INFORMATIONSBIBLIOTHEK

 Sigma Technical Press

UB/TIB Hannover 89
100 626 734

FH 7940

18523

Chapter 6

Controlling Several Channels Simultaneously	79
Getting Started	81
Preliminary Program for the ZX81	81
Preliminary Program for the SPECTRUM	81
Examples :	
Example 1: Input and Output	85
The Program	85
Example 2: Sixteen Christmas Tree Lights	86
The Program	86
Example 3: A Game of Chase	88
The Program	88
Example 4: Analogue to Digital Converter	90
The Program	91

Appendix 1

Machine Code	
The Key Steps in the Machine Code Programs	
Program to Output Numbers	

Appendix 2

Decimal and hexadecimal	
-------------------------	--

INTRODUCTION

This book illustrates for Sinclair Spectrum and ZX81 owners the use of different add-ons for their computers. Circuit diagrams are given for those who wish to build their own add-ons; they can also be bought from Thurnall Electronics of Manchester *

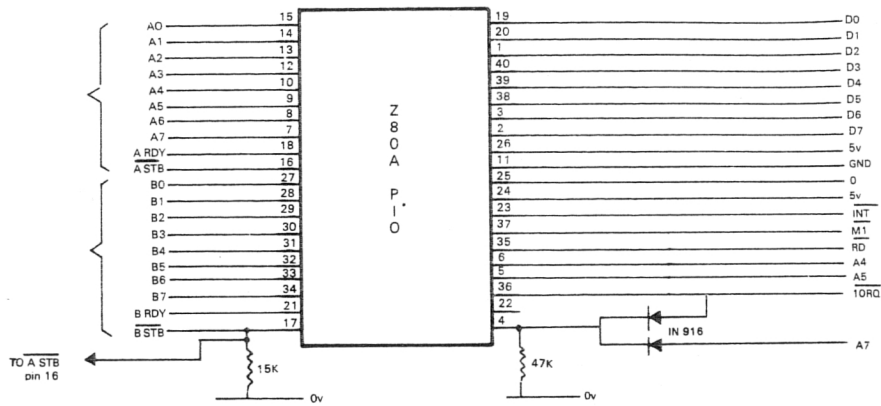
Each chapter is devoted to one add-on and is designed to be self-contained. The reader can select which parts of the book are relevant to his or her needs. For each unit a variety of examples is given, with appropriate programs: some simple, some more advanced. These are meant to illustrate to the beginner what the particular add-on can do, and, for the more ambitious, they provide a stepping-stone to more complicated programs. The book is thus an introductory guide which also explores techniques that a reader will acquire when graduating to more advanced work.

Before using any add-on, it is necessary to set up a mechanism by which the computer can communicate with the add-on. There is a standard chip called an 'Input-output controller' which fulfils this task. This is connected to the outside of the Spectrum or ZX81, between the computer and the add-on. Every project in this book requires the use of such an input-output controller. At the end of this introduction we give the circuit diagrams for this unit.

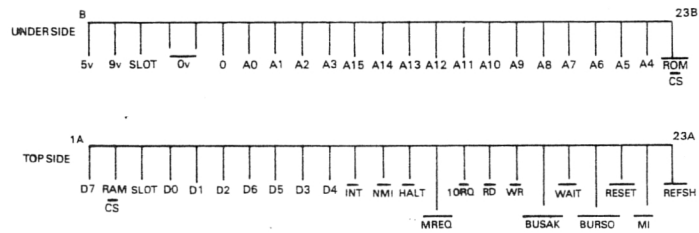
All the projects in this book use programs which are self-contained and easy to enter. Some 'machine code' is necessarily included to instruct the input-output controller, but an understanding of this code is not essential to carry out the projects. However, for those interested, the code is explained in the Appendix.

* Full details and prices from:

Thurnall Electronics
95a Liverpool Road
Cadishead
Manchester M30 5BG
(Telephone: 061 775 4461)



Connections rear of ZX81/80



Output connections, 16 line I/O port

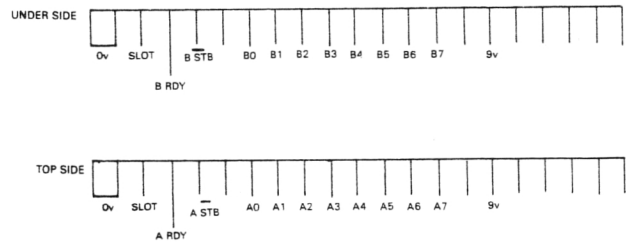
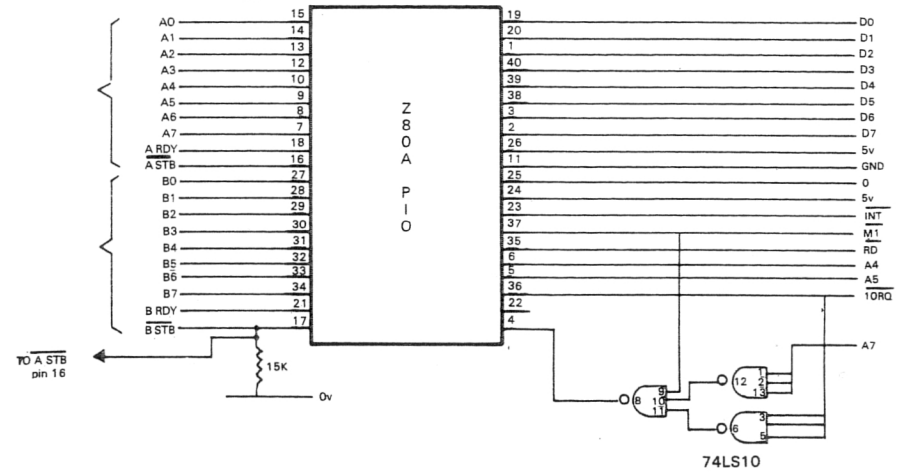
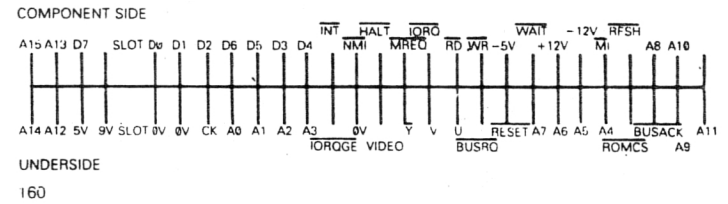


Figure 1: Circuit diagram for the TE10 input-output controller for the ZX81



Connections rear of SPECTRUM



Output connections, 16 line I/O port

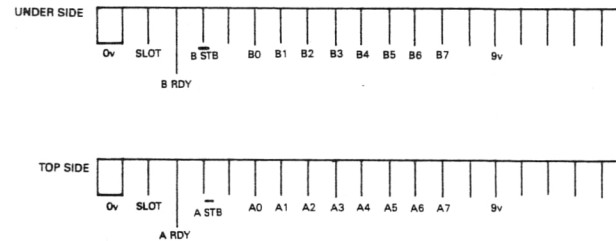


Figure 2: Circuit diagram for the TE130 input-output controller for the Spectrum

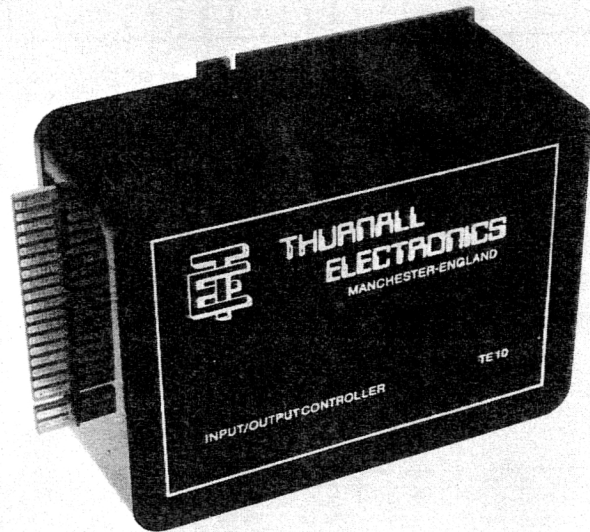
Where necessary, a suitable separate power supply is supplied for the Thurnall units. Alternatively, you can use any "Calculator-type" power supply rated at 9 Volts DC and 300 milli amps.

A separate power supply is only, in fact needed for the TE15 (Transistor driver) and TE18 (indicator) units. The rest, including the TE10, pick up their power from the computer.

ZX81 and Spectrum Listings

Throughout this book, you will see alternative programs and instructions for the ZX81 and Spectrum. These are mutually incompatible. Note that, for brevity, we always refer to NEW LINE to terminate keyboard entry. This only applies to the ZX81. On the Spectrum, you must press the ENTER key.

Where only one program is given, it will work equally well on the Spectrum or ZX81.



Unit

CHAPTER 1

CONTROLLING A LIGHT DISPLAY

This Chapter is concerned with an eight-way indicator unit which is referred to as the TE18 by Thurnall. It is pictured below:

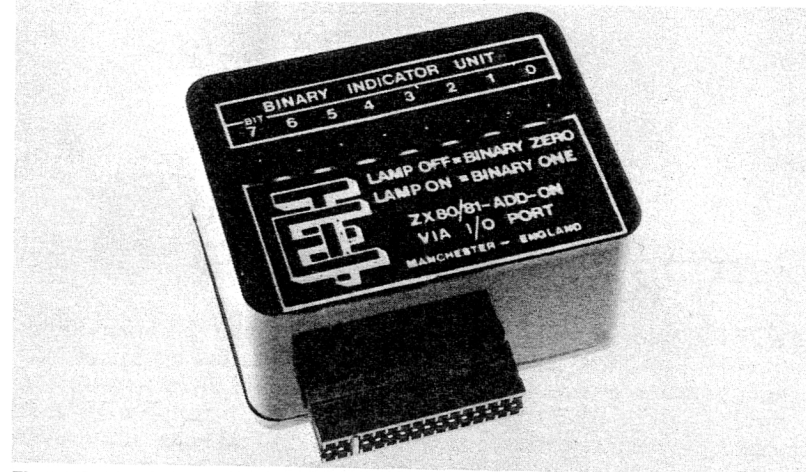


Figure 3: The TE18 indicator unit

As can be seen, the unit has a display of eight lights (or, more strictly, 'light-emitting diodes'). In this chapter we shall explore some of the ways of using these.

The circuit diagram of the TE18 is shown below:

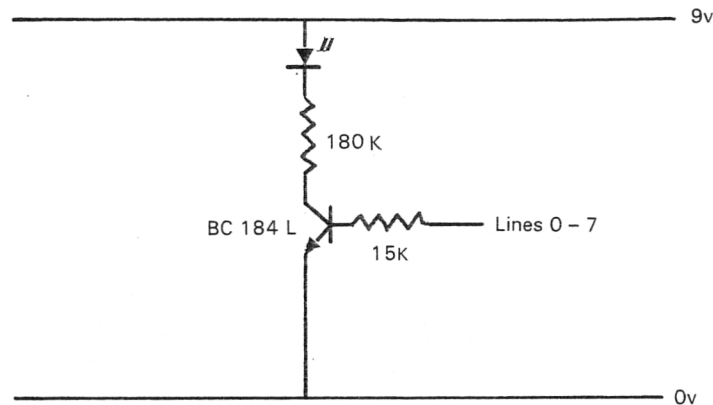


Figure 4: TE18 Circuit Diagram

Note that the TE18 requires an independent power supply.

Getting Started

The TE18 indicator unit is used in conjunction with the input-output controller (see Introduction). To connect these to the computer, you should begin by ensuring that the power supply is disconnected. The controller is then plugged in at the back of the Sinclair. The TE18 is plugged into the side of the controller. The Sinclair add-ons, such as the memory and the printer, which normally plug into the back of the computer can now be plugged into the back of the controller if desired.

The power supply for the indicator unit is plugged into the TE18, as shown in the photograph.

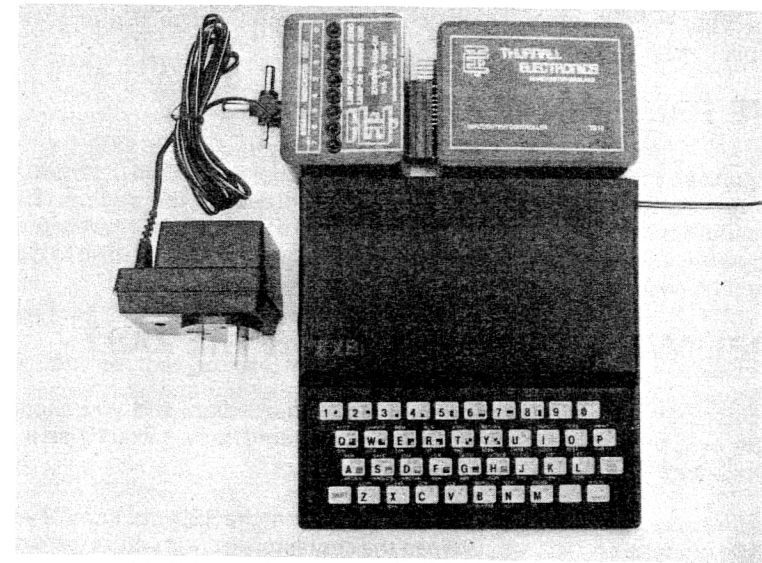


Figure 5a: ZX81, TE10, TE18 and Power Supply

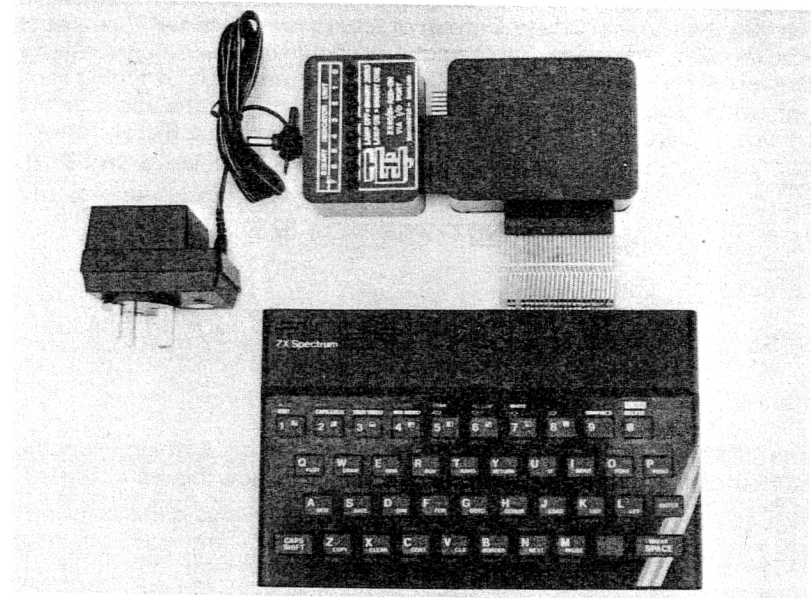


Figure 5b. Spectrum TE130, TE18 and Power Supply


Once you have checked that these are connected properly, the power supplies may be switched on.


THE PRELIMINARY PROGRAMS

Before we can use the TE18 to obey a practical command, we first have to set up a system by which the computer can control the add-on. This procedure is a necessary preliminary step in all the projects in this chapter. Depending on which Sinclair computer you are using, you will have to type in one of two short programs.

PRELIMINARY PROGRAM FOR THE ZX81

This consists of the following five lines of instruction, and you should begin by typing these into the ZX81. You should copy it exactly as it is written here.

One important point to note is that in the first line (the REM statement) you should not type any spaces between the characters except where we have written (SPACE). In the other lines the use of spaces is ignored by the computer, but you may wish to employ them for your own benefit in reading the display. Where a group of letters is **underlined** this means you should type them as one character as shown on the keyboard. Where they are not **underlined**, you should spell out the words by typing each individual letter. If you have difficulty in finding any character on the keyboard, there is an index at the back of the ZX81 manual. Be careful with  (which is GRAPHIC SHIFT Y) and ** in line 5 (which is SHIFT H).



```
1  REM Y?PEEK XE>RND777Y(SPACE)  PEEK XTAN
2  LET OPT=16514
3  POKE OPT+14,95
4  POKE OPT+2, 127
5  LET ZZ=2**15
```

The Preliminary Program for the ZX81

The REM line is a set of instructions in machine code which allow the computer to control the TE10. A full explanation of how this works is given in the Appendix.

When you have typed these instructions, check your display on the TV screen and then type RUN (and NEW LINE, of course). You will notice that if you type NEW LINE again to obtain a listing of the program, the REM

statement will be slightly different. This is because the computer has executed lines 3 and 4 in the program, which are designed to insert characters there which could not otherwise be put in a REM statement. Hence the listing should look like this:

```
1  REM Y?PEEK  E>RND777Y  PEEK ?TAN
2  LET OUT=16514
3  POKE OUT+14,95
4  POKE OUT+3,127
5  LET ZZ=2**15
```

How the Preliminary Program appears after typing RUN

If it does not look like this, then a mistake has been made. The easiest way to rectify this is to unplug the power supply to the computer and begin afresh.

PRELIMINARY PROGRAM FOR THE SPECTRUM

The preliminary program for the Spectrum is as follows:

```
1  LET OPT=USR "A": RESTORE      (This outputs a word)
2  FOR I=OPT TO OPT+15
3  READ A: POKE I,A
4  NEXT I
5  DATA 62,15,211,127,42,77,92,35,35,35,62,0,134,211,95,
   201
6  LET ZZ = 2 115
```

The Preliminary Program

This preliminary program sets up, in the part of the memory reserved for graphics characters, a short machine code routine. This routine allows the computer to operate the input TE130 output controller. An explanation of how this works is given in the Appendix.

Project 1: Teach yourself, or a child, binary numbers

Binary numbers are a system of counting in multiples of two, in contrast to our ordinary decimal system in which we count in units of ten. This project is important and instructive because binary is the system in which all computers work.

Here we use the TE18 indicator unit to display binary numbers visually. This proves a quick and effective method to learn how to convert decimal numbers into binary.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below, which is suitable for both the Spectrum and ZX81.

```
100 REM OUTPUT A BINARY NUMBER
110 PRINT "NUMBER BETWEEN 0 AND 255"
120 INPUT D
130 LET A=ZZ+D
140 LET A=USR OPT
150 GOTO 120
```

Program for Binary Numbers Project to use in conjunction with the Preliminary Program.

The REM line 100 is purely for your own reference. Line 110 prints the instruction to you to type an ordinary (decimal) number of your choice, and 120 puts that number into the variable called D. 130 and 140 instruct the computer to use the preliminary program and output the number on the light display. (This is more fully explained in the appendix). Line 150 allows you to repeat the process automatically.

Now type RUN (and of course ENTER or NEW LINE*). As you are asked on the screen, choose a number between 0 and 255. You will then see the number displayed visually on the TE18 indicator in binary.

NEW LINE is used on ZX81, and ENTER on the Spectrum. From here on, we will always refer just to NEW LINE for brevity.

Binary numbers are expressed in terms of 0's and 1's and in the display, 0 is represented by an "off" light and 1 by an "on" light.

To stop the program, just type in any letter (and NEW LINE of course).

Typing in NEW LINE again will give you back your original program listing.

USING THE PROGRAM

Begin by choosing random numbers and observing the resulting display.

See if you can deduce any pattern. You will note that no two numbers produce an identical display.

Then try typing in a sequence of numbers - for example count from 0 to 10. Note how the lights change.

To help you establish the pattern, write down on a piece of paper which lights are on and off for a particular number.

For example, the number 10:

Light number	on or off
0	off
1	on
2	off
3	on
4	off
5	off
6	off
7	off

Make a table of number sequences and examine the progression. You should end up with a table like this:

Light no.	Decimal number									
	0	1	2	3	4	5	6	7	8	9
0	off	on	off	on	off	on	off	on	off	on
1	off	off	on	on	off	off	on	on	off	off
2	off	off	off	off	on	on	on	on	off	off
3	off	off	off	off	off	off	off	off	on	on
4	off	off	off	off	off	off	off	off	off	off
5	off	off	off	off	off	off	off	off	off	off
6	off	off	off	off	off	off	off	off	off	off
7	off	off	off	off	off	off	off	off	off	off

Obviously, writing "on" and "off" is cumbersome, and we need to simplify the notation. So write 1 for an on light and 0 for an off light. In this new table, put the light numbers across the top and the decimal numbers down the side. The table now looks like this:

Decimal no.	light number							
	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	1
10	0	0	0	0	1	0	1	0
33	0	0	1	0	0	0	0	1
100	0	1	1	0	0	1	0	0
255	1	1	1	1	1	1	1	1

You will probably now have begun to see how binary works. The counting is done in multiple units of 2, or powers of two as they are usually called.

The progression is:

1 2 4 8 16 32 64 128

So - light number 0 represents 1 when it is switched on. Light number 1 represents 2 when on, light number 2 represents 4, and so on, with the last light number 7 standing for 128.

Type in this number sequence:

1 2 4 8 16 32 64 128

and each of the lights should come on in turn. This is how these numbers are expressed in binary:

Decimal number	Binary
0	0
1	1
2	10
4	100
8	1000
16	10000
32	100000
64	1000000
128	10000000

Note that we have ignored any zeros that come before the first 1 in the binary column.

So to convert back from binary to decimal, you add up the numbers relating to each light.

For example:

on off on off on off on off

or:

1 0 1 0 1 0 1 0

10101010	is	1 X 128 = 128
		0 X 64 = 0
		1 X 32 = 32
		0 X 16 = 0
		1 X 8 = 8
		0 X 4 = 0
		1 X 2 = 2
		0 X 1 = 0
10101010		<hr/> = 170

You should now understand the basis of the binary system. To practice, you can play simple games. Ask someone else to type in a decimal number and then try to deduce what it is from the light display. Or think of a light sequence and try to find the number which will produce it.

Project 2: Making a Metronome

To make a simple visual metronome, we need two distinct light patterns which alternate on the TE18 indicator unit. The speed at which they do so must be adjustable. In this program we have chosen two groups of lights: 11110000 and 00001111. (If you have not done the binary numbers project, the former means "lights 7,6,5 and 4" and the latter means "lights 3,2,1 and 0"). When you have input the program these will illuminate alternately.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```

100 REM METRONOME
110 PRINT "TYPE SPEED"
120 INPUT SPEED
130 LET D = 15
140 PAUSE SPEED
150 LET A=ZZ+D
160 LET A=USR OPT
170 LET D = 3600/D
180 GOTO 140

```

Program for Metronome Project to use in conjunction with the Preliminary Program.

The REM line 100 is purely for your own reference. Line 110 prints the instruction to you to type an ordinary (decimal) number which tells the computer how fast the metronome should run. (As an example, start by typing in 30 when you RUN the program.) Line 120 puts that number into the variable called SPEED. The decimal number 15 equals 00001111 in binary, one of the light patterns we wish to use. Line 130 sets the variable D to 15 (00001111). Line 140 causes the computer to pause in deference to your speed instruction. Lines 150 and 160 output the number D to the TE18. (This is more fully explained in the Appendix). Line 170 is the mechanism by which we achieve the alternation of the lights. If D equals 15, as set in line 130, then 3600/15 equals 240 (11110000

in binary), which is the other light pattern. If D then equals 240, then 3600/240 equals 15 (00001111 in binary), the original light pattern.

Line 180 instructs the computer to jump back to line 140 to repeat the process.

Now type RUN (and of course NEW LINE). As you are asked on the screen, choose a number. The metronome will then begin working at that speed.

To stop the program, type BREAK.

USING THE PROGRAM

To use the program, explore the speeds that different numbers produce.

We would advise the range 10 to 60, presto to lento. You can then find a speed to suit your requirements.

Project 3: Disco Light Show- Micro Version

A disco light show on a scale for the average teddy bear requires a random selection of light patterns on the TE18 indicator unit. In this project the computer is instructed to choose binary numbers at random with the operator allowed to control the speed.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this Chapter). For this project you should follow that by typing in the program below:

```
100 REM TEDDIES DISCO
110 PRINT "TYPE SPEED"
120 INPUT SPEED
130 PAUSE SPEED
140 LET D=255*RND
150 LET A=ZZ+D
160 LET A=USR OPT
170 GOTO 130
```

Program for the Disco Light Show to use in conjunction with the Preliminary Program.

Line 100 is purely for your own reference. Line 110 prints the instruction to you to type an ordinary (decimal) number, which tells the computer how fast to change the light patterns during the disco. Line 120 puts that number into the variable called SPEED, and line 130 instructs the computer to pause for that time. Line 140 tells the computer to multiply 255 (the highest number that the TE18 can display, 11111111 in binary) by a random number between 0 and 1 of the computer's choice.

The result is stored in the variable called D, and lines 150 and 160 output this to the TE18. (These two lines are more fully explained in the appendix). Line 170 instructs the computer to repeat the process.

Now type RUN (and of course NEW LINE). As you are asked on the screen, choose a number. We suggest you start with 20. The disco light show will then operate.

To stop the program, type BREAK.

Project 4: An Electronic Die

This is another project which uses the computer's facility to produce random numbers. With this program the TE18 will display a number from 1 to 6 for use with any game, each time you press the key NEW LINE.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below.

```
100 REM DIE
110 PRINT "TYPE NEW LINE"
120 INPUT Z$
130 IF Z$ <> "" THEN STOP
140 LET D=0
150 LET A=ZZ+D
160 LET A=USR OPT
170 LET D=2↑ INT (1+6*RND)
180 LET A=ZZ+D
190 LET A=USR OPT
200 GOTO 120
```

(ZX81 users should use **
in place of ↑)

Program for an Electronic Die to use in conjunction with the Preliminary Program.

Line 100 is purely for your own reference. Line 110 prints the instruction to you to type NEW LINE, so that each time you do so the computer can display another random number. Line 120 puts whatever you type into the string Z\$, and if you type anything other than NEW LINE then line 130 will make the computer stop. Lines 140 to 160 tell the computer to switch off all the lights by outputting the number 0.

Line 170 is the most complicated part of the program. RND produces a random number between 0 and a number just less than 1. 6*RND therefore is a number between 0 and a number just less than 6, and adding 1 to this makes the upper limit just less than 7. The function INT

takes away all numbers after the decimal point, therefore leaving a whole number between 1 and 6 inclusive. We then take 2 to the power of this random number. To see what this yields in binary, and hence on the light display, see the table below.

2**1 = 2 = 0000010
2**2 = 4 = 0000100
2**3 = 8 = 00001000
2**4 = 16 = 00010000
2**5 = 32 = 00100000
2**6 = 64 = 01000000

These six binary numbers correspond to the lights labelled 1 to 6 on the TE18 indicator unit.

Lines 180 and 190 output this to the TE18 (see appendix) and line 200 instructs the computer to repeat the process.

Now RUN the program. To stop it, input anything other than NEW LINE.

USING THE PROGRAM

Start playing the game. Whenever you would usually roll a die, merely press NEW LINE and the computer will select a number for you.

Project 5: A Binary Counter

This project demonstrates how the computer counts in binary numbers by a progressive display on the TE18 indicator unit. The operator can determine the speed at which this occurs.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```
100 REM BINARY COUNTER
110 PRINT "TYPE SPEED"
120 INPUT SPEED
130 FOR D=0 TO 255
140 PAUSE SPEED
150 LET A=ZZ+D
160 LET A=USR OPT
170 NEXT D
```

Program for a Binary Counter for use in conjunction with the Preliminary Program.

Line 100 is purely for your own reference. Line 100 prints the instruction to you to type an ordinary (decimal) number, which tells the computer how fast to count. Line 120 puts that number into the variable called SPEED.

Lines 130 to 170 cause the computer first to set D to 0, then 1, and so on up to 255. As a result of line 140 it pauses by the amount you have instructed, thereby controlling the speed of counting. Lines 150 and 160 output the number D to the TE18 indicator unit. (This is more fully explained in the appendix).

Then type RUN. The program will automatically stop when the number 255 is reached. If you wish to stop it in the middle of the program, type BREAK.

USING THE PROGRAM

If you select a medium speed (such as 25), this program provides a good visual illustration of the progression of binary numbers. (This system of counting was explained in Project 1 of this chapter.) A very slow speed (such as 150) provides chance for the learner to indicate which light comes on next, and then see if the deduction was correct. A fast speed (like 1) shows a computer in a hurry!

Project 6: An Electronic Timer

This project provides a timing device for periods of up to four minutes. It counts down in binary, thereby indicating, continuously, the amount of time left.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```
100 REM TIMER
110 PRINT "HOW MANY MINUTES?"
120 INPUT MINS
130 PRINT MINS
140 SLOW
150 FOR D=MINS*60 TO 0 STEP -1
160 PAUSE 47 : REM 56 FOR THE USA TIMEX-SINCLAIR 1000
170 LET A=ZZ+D
180 LET A=USR OPT
190 NEXT D
```

Program for the Electronic Timer to use in conjunction with the Preliminary Program.

* *Spectrum owners should omit line 140 and use PAUSE 49 (59 in the USA) in line 160.*

Line 100 is purely for your own reference. Line 110 prints the instruction to you to type an ordinary (decimal) number, the number of minutes you wish to count. The number must not be more than 4, and it can include fractions (such as $1 + 1/2$ or 2.5). Line 120 puts that number into the variable called MINS, and line 130 prints it. Line 140 ensures that the computer is in SLOW mode (ONLY applicable to the ZX81). This is necessary for the timer to be accurate.

Lines 150 and 190 cause the computer first to set D to the number of seconds equivalent to the time you have input (MINS*60). Each time this number is decreased by 1 until it reaches 0. Because of the PAUSE in line 160 the interval between each number is one second. Lines 170 and 180 output the number D to the TE18 indicator unit. (This is more fully explained in the appendix).

Then type RUN. The program will count down on the indicator unit until the number 0 is reached. If you wish to stop it in the middle of the program, type BREAK.

USING THE PROGRAM

This timer is suitable for use in games where players are subject to time limits. In a game (like 'Just a Minute') where a stopwatch is required, type BREAK to stop the countdown and CONT to resume. It may also be used when boiling eggs or when making short telephone calls - or on any other occasion you need a timer or stopwatch.

CHAPTER 2

READING A SET OF SWITCHES INTO THE COMPUTER

This chapter is concerned with the TE17, which is an eight-way switch unit. It is pictured below:

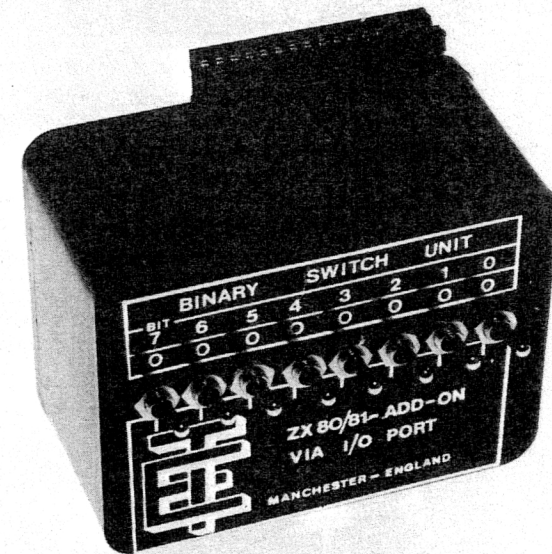


Figure 6: The TE17 Switch Unit

As can be seen, the unit has a set of eight switches and in this chapter he shall explore some of the ways of using these.

The circuit diagram of the TE17 is shown below:

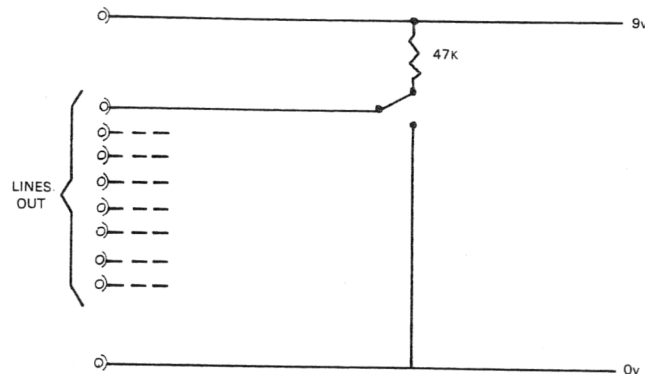


Figure 7: Circuit Diagram of TE17 Unit

Note that this does NOT require a separate power supply.

Getting Started

The TE17 switch unit is used in conjunction with an input/output controller. To connect these to the computer, you should begin by ensuring that the power supply is disconnected. The controller is then plugged in at the back of the Sinclair. The TE17 is plugged into the side of the controller. The Sinclair add-ons, such as the memory and the printer, which normally plug into the back of the computer can now be plugged into the back of the controller if desired. The photograph below shows the units properly connected.

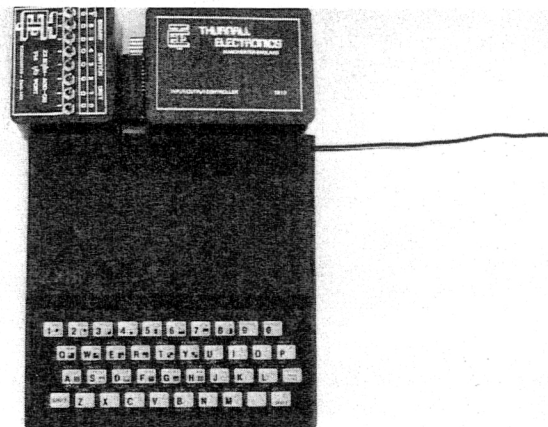


Figure 8a: TE10 Input/Output Unit for the ZX81



Figure 8b: TE130 Input / Output Unit for the Spectrum

The units properly connected.

Once you have checked that these are connected properly, the power supply may be switched on.

THE PRELIMINARY PROGRAM

Before we can use the TE17 switch unit, we first have to set up a system by which the computer can read the add-on. This procedure is a necessary preliminary step in ALL the projects in this chapter. Depending on which Sinclair computer you are using, you will have to type in one of two short programs.

PRELIMINARY PROGRAM FOR THE ZX81

This consists of the following four lines of instruction, and you should begin by typing these into the ZX81. You should copy it EXACTLY as it is written here. The Spectrum version comes later.

One important point is that in the first line (the REM statement) you should not type any spaces between the characters except where we have written (SPACE). In the other lines, the use of spaces is ignored by the computer, but you may wish to employ them for your own benefit in reading the display. Where a group of letter is **underlined** this means you should type them as one character as shown on the keyboard. Where they are not **underlined**, you should spell out the words by typing each individual letter. If you have any difficulty in finding any character on the keyboard, there is an index at the back of the ZX81 manual. Be careful with **K**, **□** and **■**, which are all obtained in GRAPHICS mode.

```
1 REM Y K JPEEK X <=XM S RND ■ (SPACE) (SPACE) TAN
2 LET INPT=16514
3 POKE INPT+4,127
4 POKE INPT+6,95
```

The Preliminary Program.

The REM line is a set of instructions in machine code which allow the computer to read the TE17. A full explanation of how this works is given in the appendix.

When you have typed these instructions, check your display on the TV screen and then type RUN. You will notice that if you type NEW LINE again to obtain a listing of the program, the REM statement will be slightly different. This is because the computer has executed lines 3 and 4 in the program, which are designed to insert characters there which could not otherwise be put in a REM statement. Hence the listing should look like this:

```
1 REM YKJPEEK K<=?MSRND ■ TAN
2 LET INPT=16514
3 POKE INPT+4, 127
4 POKE INPT+6,95
```

How the Preliminary Program appears after typing RUN.

If it does not appear like this, then a mistake has been made. The easiest way to rectify this is to unplug the power supply to the computer and begin afresh.

THE PRELIMINARY PROGRAM FOR THE SPECTRUM

The preliminary program for the Spectrum is as follows:

```
1 LET INPT=USR "C": RESTORE
2 FOR I=INPT TO INPT+10
3 READ A: POKE I,A: NEXT I
4 DATA 62,176,47,211,127,219,95,6,0,79,201
```

The Preliminary Program.

This preliminary program sets up, in the part of the memory reserved for graphics characters, a short machine code routine. This routine allows the computer to operate the input/output controller. A full explanation of how this works is given in the Appendix.

Project 1: Reading the Switches into the Computer

This project is a simple program to illustrate how the switches work. Binary numbers indicated via the switches are converted into decimal numbers by the computer and are displayed on the screen.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM READ SWITCHES
110 PRINT AT 0,0"SWITCHES READ"; USR INPT; "(SPACE) (SPACE)"
120 GOTO 110
```

Program for Reading the Switches into the Computer to use in conjunction with the Preliminary Program.

The REM line is purely for your own reference. In line 110 the PRINT instruction tells the computer to display the number at the top of the screen each time, after putting "SWITCHES READ". USR INPT is the general command to the computer to make a reading of the switches, and it can be used in any BASIC command to obtain such a reading. This uses the machine-code in the REM statement in the preliminary program. (See appendix for a fuller explanation).

The two spaces ensure that the previous number is completely erased. Otherwise, for example, a 1 following a 255 would appear as 155.

USING THE PROGRAM

Binary numbers are a system of counting in multiples of two (explained more fully in project 1 of chapter 2).

On the TE17 the switches equate to the following powers of 2:

Switch	Decimal number	Binary number
0.	1	00000001
1	2	00000010
2	4	00000100
3	8	00001000
4	16	00010000
5	32	00100000
6	64	01000000
7	128	10000000

As you will see from the table, binary numbers are written using zero and one. On the TE17, an 'off' switch is written '0' and an 'on' switch is written '1'.

So, for example, the binary number 11000 means switches 4 and 3 would be 'on'. As switches 4 and 3 correspond to the decimal numbers 16 and 8, 11000 in binary is $16+8=24$ in decimal. Check this using the program.

Think of a binary number, any combination of zeros and ones, and then try to work out its decimal equivalent. To check your deduction, put the binary number on the TE17 switch unit, and the computer will give you the correct answer. For example, if you thought of 10110001, you would put switches 7, 5, 4 and 0 into the 'on' position and switches 6, 3, 2 and 1 to 'off'. The computer would then display 177 (ie $128+32+16+1$).

Project 2: A Combination Lock

This project illustrates the way that a computer might be used in a lock for a safe. The program can also be used as another binary numbers game.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```

100 REM COMBINATION LOCK
110 PRINT "TYPE THE COMBINATION NUMBER"
120 INPUT COMB
130 PRINT COMB
140 IF USR INPT <> COMB THEN GOTO 140
150 PRINT "CONGRATULATIONS"
160 PRINT "DO YOU WANT TO TRY AGAIN?"
170 INPUT Z$
180 IF Z$="N" THEN STOP
190 CLS
200 GOTO 110

```

Program for the Combination Lock to use in conjunction with the Preliminary Program.

The REM statement is purely for your own reference. Line 100 prints the instruction on the screen for you to type a decimal number between 0 and 255. Line 120 inputs this to the computer and line 130 prints it on the screen. You then have to find the equivalent binary number using the switches. (See project 1 if you are unsure how switches can represent binary numbers). Line 140 ensures that the program will not continue until you have found the correct number. You are 'locked out' of the computer. When you do find the combination, lines 150 and 160 will express the computer's felicitations and invite you to try again. For yes, type Y; for no, type N. If you type the latter, line 180 will stop the program.

Line 190 clears the screen. If you typed Y for another try, line 200 returns the computer to the beginning of the program.

To stop the program at any stage type BREAK.

USING THE PROGRAM

This project is a demonstration of a simple way to lock the computer. If you want the combination to be a total mystery, then omit line 130 from the program; this would mean you would not have the decimal number as a guide on the screen.

Project 3: Time Your Reactions

This game tests how fast you can react to a signal from the computer and gives you a result in seconds.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM REACTION TIMER
110 SLOW
120 PAUSE 200+200*RND
130 FAST
140 IF USR INPT>=128 THEN PRINT "YOU ARE CHEATING"
150 FOR N=0 TO 9999
160 IF USR INPT>=128 THEN GOTO 180
170 NEXT N
180 PRINT "YOUR TIME IS " ; N/100;" SECONDS"
190 PRINT "TYPE NEWLINE"
200 INPUT Z$
210 CLS
220 GOTO 100
```

Program for a Reaction Timer to use in conjunction with the Preliminary Program.

* *Spectrum owners should omit line 110 and replace 130 by 130 PRINT "GO". The time in line 180 should be N/114 seconds.*

The REM statement is purely for your own reference. Line 110 sets the computer speed to slow (ZX81 only) so that the screen remains stable. Line 120 makes the computer pause for a time between 4 and 8 seconds at random. Line 130 changes the computer speed to Fast (ZX81 only) and destabilizes the screen; this is the signal to you to put switch number 7 to 'on'. As the computer is in Fast mode it can measure your reaction time more accurately. Line 140 ensures you cannot get away with cheating by flicking the switch in advance of the signal.

The computer counts your reaction time in lines 150 to 170. The computer counts as fast as it can until you set switch number 7, which

represents the number 128 in binary. Your time in fractions of a second will then appear on the screen because of line 180 (in this program the computer counts in, approximately, hundredths of a second).

Because of lines 190 and 200 the computer waits for you to type NEW LINE before clearing the screen in line 120 and beginning the game again.

To stop the program type BREAK whilst it is running.

USING THE PROGRAM

First put all of the switches to the 'off' position. Place a finger in readiness against switch number 7. Type RUN.

When you are ready to begin the game, press NEW LINE. As soon as the screen changes (or "Go" is displayed, on the Spectrum version) flick the switch number 7. See how short a time you can achieve. This is one game at which a child can defeat a parent.

Project 4: Binary Speed Game

This project is a simple variation on the previous one. Those who have used project 1 in chapter 1 (binary numbers) may want a game to test their binary conversion skills. The player asks the computer to choose a random decimal number, and then tries to find its binary equivalent on the switches. The time taken to do this is recorded and displayed on the screen.

The difficulty of the program can be varied as you become more skilled.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM BINARY SPEED GAME           (ZX81 owners should use
110 SLOW                             ** in place of ↑ )
120 PRINT "HOW MANY SWITCHES?"
130 INPUT S
140 LET X=INT((2↑S)*RND)
150 PRINT X
160 FOR N=0 TO 9999
170 IF USR INPT=X THEN GOTO 190
180 NEXT N
190 PRINT "YOUR TIME IS ";INT (N*.374)/10; "SECONDS"
200 PRINT "TYPE NEWLINE"
210 INPUT Z$
220 CLS
230 GOTO 100
```

Program for a Binary Speed Game to use in conjunction with the Preliminary Program.

** Spectrum owners should omit line 100 and change the expression in line 190 to (N/100)*

The REM statement is purely for your own reference. Line 110 sets the computer speed to slow so that the screen remains stable. Through lines

120 and 130 you are able to set the difficulty of the game by choosing how many switches should be used.

As a result of lines 140 and 150 the computer chooses a random number within the range you have specified, and this number (X) is printed on the TVscreen. If you had chosen three switches, for example, then S would be set to 3. The expression $(2^{**}3)$ therefore equals 8. RND produces a random number between 0 and a number just less than 1. Therefore, $8 * \text{RND}$ is a number between 0 and a number just less than 8. The function INT takes away all numbers after the decimal point, leaving a whole number between 0 and 7 inclusive, 7 being the highest number you can express with three switches.

The computer in lines 160 to 180 counts the time you take to find the correct binary equivalent. Your time in seconds will then appear on the screen. Because of lines 200 and 210, the computer waits for you to type NEW LINE before clearing the screen in line 220 and beginning the game again.

To stop the program, type BREAK whilst it is running.

USING THE PROGRAM

First, put all of the switches to the 'off' position. Type RUN.

Decide how many switches you want to bring into play, any number between 2 and 8 inclusive. (If you use only one switch, there is no game as the computer can only give you 0 or 1.) Bear in mind the increasing difficulty the higher the number you choose. This table shows how the maximum range increases with the number of switches used:

No. of switches	Highest number
2	3
3	7
4	15
5	31
6	63
7	127
8	255

Having made your decision, type the number of switches. The computer will respond by giving you a random decimal number in the appropriate range, and you have to convert it to binary on the switches. When you have the correct equivalent, the computer will give you your time in seconds.

You can play this on your own, trying to better your previous score each time, or compete with another player.

Project 5: The Choice Maker

For many games, one needs an aid by which a player can be given a task silently so that others cannot hear; in charades, for example, where the 'actor' needs to be given a subject. In this project, the computer holds a list of subjects/tasks/choices, one of which will be displayed silently on the screen when a player selects a switch.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM CHOICE MAKER
110 DIM A$(8,8)
120 FOR I=1 TO 8
130 INPUT A$(I)
140 NEXT I
150 IF USR INPT<> 0 THEN PRINT AT 0,0;
    A$(1+INT (LN USR INPT/LN 2))
160 GOTO 150
```

Program for a Choice Maker to use in conjunction with the Preliminary Program.

The REM statement is purely for your own reference.

Line 110 sets the dimension of the character string A\$. There will be eight choices, each of up to eight characters in length. Lines 120 to 140 enable you to input your eight different choices. When a selection has been made by switching 'on' one of the switches of the TE17, the mathematics in line 150 uses the computer's logarithm function to deduce which you have chosen and this is then printed at the top of the screen. The process is repeated because of line 160.

To stop the program, type BREAK.

USING THE PROGRAM

First, put all of the switches to the 'off' position. Type RUN.

The computer types double quotes ("") at the bottom of the screen, and you should now type in your first choice. When this has been input into the computer, type in the second choice, and so on until all eight have been fed in. In this program the number of characters for each choice is restricted to eight because there are limits to the ZX81 memory. If you have a memory extension or a Spectrum, then you can change the program to accept longer choices: do this by increasing the second number between the brackets in line 110. Once you have input all your choices, the program is ready to use.

The player can now choose any one switch at random, switch it to the 'on' position, and his subject/task/choice will be displayed on the screen.

CHAPTER 3

USING A JOYSTICK

This chapter illustrates ways of using a joystick. For this we have used the TE20 in the range of Thurnall Electronics, which is pictured below.

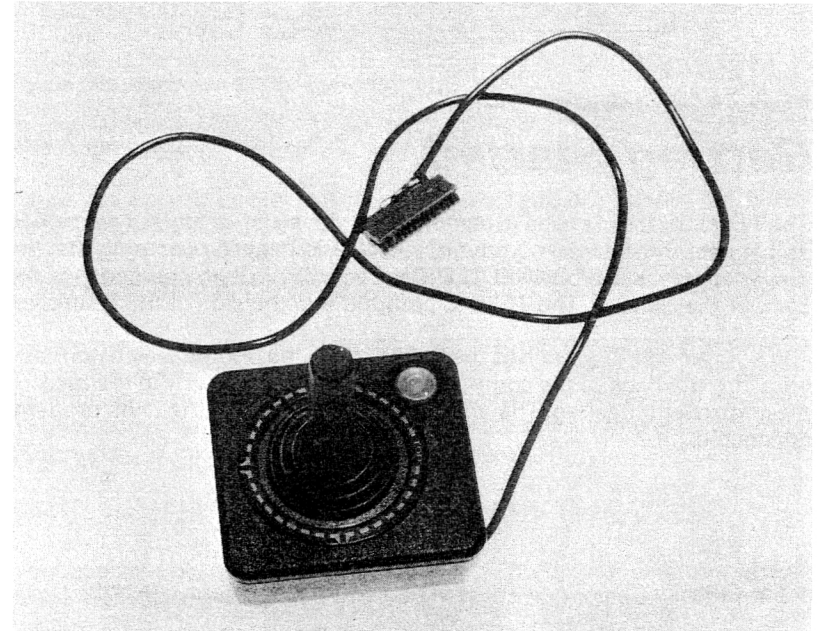


Figure: 9 Photograph of TE20.

The joystick has eight positions and a red push-button.

The circuit diagram of the TE20 is shown below:

Note that no separate power supply is needed.

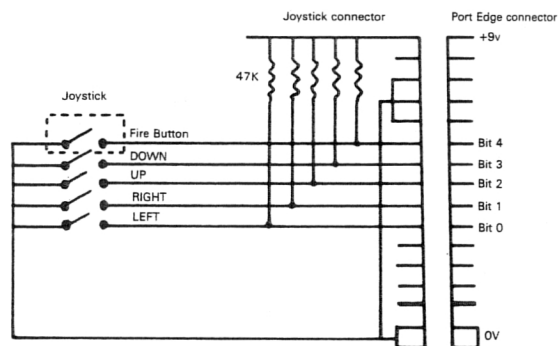


Figure: 10 Circuit diagram of TE20

Getting Started

The TE20 joystick is used in conjunction with an input/output controller. To connect these to the computer, you should begin by ensuring that the power supply is DISCONNECTED. The controller is then plugged in at the back of the Sinclair. The TE20 is plugged into the side of the controller.

The Sinclair add-ons, such as the memory and the printer, which normally plug into the back of the computer, can now be plugged into the back of the controller if desired. The photograph below shows the units properly connected.

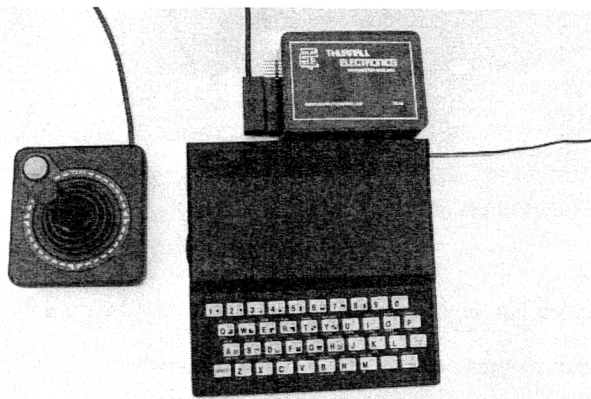


Figure 11a: ZX81, TE10 and TE20



Figure 11b: Spectrum TE130 and TE20

The units properly connected.

Once you have checked that these are connected properly, the power supply may be switched on.

PRELIMINARY PROGRAM

Before we can use the TE20 joystick, we first have to set up a system by which the ZX81 can read the add-on. This procedure is a necessary preliminary step in all the projects in this chapter. Depending on which Sinclair computer you are using, you will have to type in one of two short programs.

PRELIMINARY PROGRAM FOR THE ZX81

This consists of the following four lines of instruction, and you should begin by typing these into the ZX81. You should copy it exactly as it is written here.

One important point is that in the first line (the REM statement) you should not type any spaces between the characters except where we have written (SPACE). In the other lines, the use of spaces is ignored by the computer, but you may wish to employ them for your own benefit in reading the display. Where a group of letters is **underlined**, this means you should type them as one character as shown on the keyboard. Where they are not **underlined**, you should spell out the words by typing each individual letter.

If you have any difficulty in finding any character on the keyboard, there is an index at the back of the ZX81 manual. Be careful with K, S and □, which are all obtained in the GRAPHICS mode.

```
1 REM Y KJPEEK X<=XM S RND □(SPACE) (SPACE) TAN
2 LET INPT=16514
3 POKE INPT+4, 127
4 POKE INPT+6,95
```

The Preliminary Program.

The REM line is a set of instructions in machine code which allow the computer to read the TE20. A full explanation of how this works is given in the appendix.

When you have typed these instructions, check your display on the TV screen and then type RUN. You will notice that if you type NEW LINE again to obtain a listing of the program, the REM statement will be slightly different. This is because the computer has executed lines 3 and 4 in the program, which are designed to insert characters there which could not otherwise be put in a REM statement. Hence the listing should look like this after typing RUN:

```
1 REM Y KJPEEK K<=?M S RND □ TAN
2 LET INPT=16514
3 POKE INPT+4,127
4 POKE INPT+6,95
```

If it does not appear like this, then a mistake has been made. The easiest way to rectify this is to unplug the power supply to the computer and begin afresh.

PRELIMINARY PROGRAM FOR THE SPECTRUM

The preliminary program for the Spectrum is as follows:

```
1 LET INPT=USR "C": RESTORE
2 FOR I=INPT TO INPT+10
3 READ A: POKE I,A: NEXT I
4 DATA 62,176,47,211,127,219,95,6,0,79,201
```

The Preliminary Program.

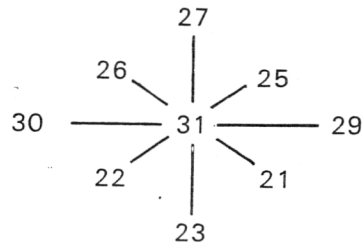
This preliminary program sets up in the part of the memory reserved for graphics characters a short machine code routine. This routine allows the computer to operate the input/output controller. A full explanation of how this works is given in the Appendix.

NOTES ON THE PROJECTS IN THIS CHAPTER

All the projects in this chapter use a particular section of program, which translates the position of the joystick into a direction for the dot (called a cursor) to move on the screen. For those interested, the following paragraphs explain how this works. It is applicable to every program in this chapter.

The joystick contains five switches which represent numbers in a binary progression (binary numbers are explained in project 1 of chapter 1).

These numbers in decimal are 1, 2, 4, 8, and 16. These numbers are counted when the relevant switch is in the 'off' position. Therefore, when all switches are 'off' (the neutral position) the binary number read into the computer is equivalent to 31 in decimal. Different positions of the joystick give different numbers, and these are shown in the diagram below:



As you can see, the numbers are obtained by subtracting 1, 2, 4, or 8 from 31 to give the West, East, North and South positions. Intermediate positions use combinations of values.

The switch for the 'fire' button represents 16 in decimal. Hence, when this button is pressed, 16 is subtracted from all the numbers on the above diagram.

The section of program which interprets all these numbers on the joystick is as follows:

```

130 LET Z=USR INPT
140 LET B=Z<= 15
150 LET Z=Z+ 16*B
160 LET UD=(Z <= 27)-2*(Z <= 23)
170 LET Z=Z-2*UD+ 6*ABS UD
180 LET RL=(Z=29)-(Z=30)
```

The computer has to have a mechanism by which it can recognize whether the 'fire' button has been pressed or not. This is done in line 140. As explained above, when the 'fire' button is pressed then 16 is subtracted from the other numbers. As $31 - 16$ equals 15, all the numbers are then 15 or less. Line 140 sets B to 1, if the number Z is less than or equal to 15, and to zero if it is greater than 15.

The 'Z <= 15' part of this statement is called a 'logical' expression and it is explained in chapter 10, exercise 3 of the Sinclair ZX81 manual, page 71. (Or in the Spectrum manual, chapter 13, page 83). If the button has been pressed, line 150 returns the numbers to their original values.

Line 160 is concerned with the cursor's vertical motion. If the cursor is meant to move upwards (numbers 25, 26 or 27) then UD ('up-or-down') is set to 1; if downwards (numbers 21, 22 or 23) then UD is set to -1; and if it is to remain on the horizontal axis (numbers 29, 30 or 31) UD is set to 0. If you substitute any of the position numbers for Z in line 160 you will see how the simple formula works.

To determine the correct position for the cursor on the horizontal axis, line 170 first converts all the numbers to the corresponding positions on the middle line of the diagram. So 26 and 22 produce 30, 27 and 23 produce 31, and 25 and 21 produce 29. On the right-left scale there are only three numbers now from which to choose. Line 180 sets RL ('Right-or-left') to: 1 for motion to the right, -1 for motion to the left, and 0 for no horizontal motion.

Project 1: Drawing Pictures on the Screen

This project allows you to trace designs on the TV screen. The eight positions on the joystick are the eight directions in which you can move the dot, which will leave a trail if you press the button. You can rub out lines you have drawn by running the cursor back over them.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

* Note for Spectrum users: Line 190 should read:

190 IF B=0 THEN PLOT OVER 1; X,Y.

```
100 REM DRAW
110 LET X=0
120 LET Y=0
130 LET Z=USR INPT
140 LET B=Z <= 15
150 LET Z=Z + 16*B
160 LET UD=(Z <= 27)-2*(Z <= 23)
170 LET Z=Z-2*UD+6*ABS UD
180 LET RL=(Z = 29)-(Z = 30)
190 IF B=0 THEN UNPLOT X,Y
200 LET X=X+RL
210 LET Y=Y+UD
220 LET X=X*(X >= 0)
230 LET Y=Y*(Y >= 0)
240 PLOT X,Y
250 GOTO 130
```

Program for Drawing Pictures on the Screen to use in conjunction with the Preliminary Program.

The REM statement is purely for your own reference. The starting position of the cursor is set by lines 110 and 120. X is the horizontal position and Y the vertical, and X=0, Y=0 is the bottom left hand corner of the screen.

Lines 130 to 180 have already been explained in the preceding note; they translate the position of the joystick into a direction for the cursor to move on the screen.

As a result of line 190, by going backwards over a trail you will rub it out. Lines 200 and 210 make the cursor move in the direction set in the previous lines. RL is the movement right-left, and UD is the movement up-down.

If you move the cursor beyond the left hand or bottom edges this can cause the computer some confusion. Lines 220 and 230 prevent this. If you find that the computer becomes short of memory you can omit these two lines. However, you then have to take care not to go beyond the edges.

Line 240 plots the movements you direct and line 250 returns the computer to the beginning of the program.

To stop the program type SPACE whilst it is running.

USING THE PROGRAM

This program uses your creative talents and what you draw is naturally a matter for your own imagination. However, at first, it may be of help to practise certain exercises so that you become familiar with using the joystick and controlling the cursor. Try drawing a square, a rectangle, or a triangle. Once you are accustomed to the speed at which the computer draws and to controlling the joystick, you can invent designs and patterns, both regular and random.

Project 2: Fast on the Draw

This project is a game for two players. One player draws as many lines as possible in a given time and then the opponent has to try to erase them as fast as possible. The time taken is recorded by the computer. The game is then repeated with the players' roles reversed. The player with the fastest time wins.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program, which differs from the previous one in only a few lines:

```
100 REM FAST ON THE DRAW
110 LET X=0
120 LET Y=X
125 FOR K=X TO 4E4
130 LET Z=USR INPT
140 LET B=Z <= 15
150 LET Z=Z+16*B
160 LET UD=Z <= 27)-2*(Z <= 23)
170 LET Z=Z-2*UD+6*ABS UD
180 LET RL=(Z = 29)-(Z = 30)
190 IF B=0 THEN UNPLOT X,Y
200 LET X=X+RL
210 LET Y=Y+UD
220 LET X=X*(X >= 0)
230 LET Y=Y*(Y >= 0)
240 PLOT X,Y
245 PRINT AT 0,0;K
250 NEXT K
```

Program for Fast on the Draw to use in conjunction with the Preliminary Program. Omit Lines 100, 220 and 230 if you become short of memory.

* Note for Spectrum users: Line 190 should read:

```
190 IF B=0 THEN PLOT OVER 1; X,Y.
```

The REM statement is purely for your own reference. The starting position for the cursor is set by lines 110 and 120. X is the horizontal position and Y the vertical, and X=0, Y=X is the bottom left hand corner of the screen. We have used 'Y=X' instead of 'Y=0', as in the previous project, because this saves on computer memory. They have exactly the same effect here. (After 'X=0' the computer stores the binary equivalent of zero. Even though this is not displayed on the screen, it does take up computer memory. This is why it is better not to repeat the '0' in the next line.)

Lines 125, 245 and 250 are the timer. Line 125 initiates the count (K), line 245 prints it at the top left hand corner of the screen, and line 250 repeats the process. 4E4 (it sounds like "Forever" with an Irish accent!) is a very large number, 40000, and even the worst player will not need more time than that.

Lines 130 to 180 have already been explained in the preliminary note earlier in this chapter. They translate the position of the joystick into a direction for the cursor to move on the screen.

Line 190 allows you to rub out lines if you have not pressed the 'fire' button. Lines 200 and 210 make the cursor move in the direction set in the previous lines. RL is the movement right-left, and UD is the movement up-down.

If you move the cursor beyond the left hand or bottom edges this can cause the computer some confusion. Lines 220 and 230 prevent this, whilst line 240 plots the movements you direct.

As this is a long program, if you have not connected add-on memory, (in the case of the ZX81), you may find it easier to omit some of the lines. The non-essential lines are the REM statement 100 and lines 220 and 230.

However, if you omit these last two lines you will have to take care not to go beyond the edges. Any player who does so will have to contend with the joystick controls being reversed.

Another way to save memory is to replace the zeros in lines 110, 220, 230 and 245 with NOT PI, although this will make the program less readable.

To stop the program, type BREAK whilst it is running.

USING THE PROGRAM

The first player begins by typing RUN and then drawing as many lines as

possible, in as complex a pattern as possible, in a given time. The computer will be counting in the top left hand corner of the screen. At 100 (or 200, whatever you have decided), the second player types BREAK to halt the drawing.

The second player then takes over the joystick and types GOTO 1, and NEW LINE. At this, he or she begins to rub out the pattern on the screen. (Remember to erase lines you do not use the 'fire' button). The count in the top left hand corner at the time the screen is completely cleared should be recorded by the second player typing BREAK.

The game is repeated with the players taking the opposite roles. The winner is the person who has the fastest time.

Project 3: Your Name in Lights

This project allows you to write your name on the screen using the joystick, as you would in a video arcade. In addition it enables you to scan through all of the characters on the ZX81 or Spectrum.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM YOUR NAME
110 LET A$="M"
120 LET N$=""
130 LET Z=USR INPT
140 LET B=Z <= 15
150 LET Z=Z+16*B
160 LET UD=(Z <= 27)-2*(Z <= 23)
170 LET Z=Z-2*UD+6*ABS UD
180 LET RL=(Z=29)-(Z=30)
190 IF B THEN LET N$=N$+A$
200 LET A$ = CHR$(CODE A$ + RL)
210 PRINT AT PI,PI;A$,N$;">"
220 GOTO 130
```

Program for Your Name in Lights to use in conjunction with the Preliminary Program.

** Note to Spectrum users: remember to use the inverse video key (shift 4) to obtain the >*

The REM statement is purely for your own reference. Line 110 sets the letter M as your starting point in the alphabet, since it is near the middle.

The N\$ in line 120 represents your name, which obviously at the beginning has to be set to a blank.

The main loop is from line 130 to 220. This consists of, first, lines 130 to 180 which are the standard joystick section of program, explained in the note earlier in this chapter. They translate the position of the joystick into a direction for the cursor to move on the screen. As a result of line 190,

when you press the 'fire' button the computer adds to your name (in N\$) the letter currently chosen (in A\$).


Line 200 is the letter selection procedure. It makes use of the code numbers for the characters, a full list of which is found in Appendix A of the ZX81 manual. The code for our starting letter M is 50, so that at the beginning CODE A\$ is also 50. If you move the joystick to the right then RL is set to 1 through lines 130 to 180. In this case, (CODE A\$+RL)=51, translates the number code into the correct character.

Line 210 prints the current letter (initially M) at position PI, PI is used here in preference to, say, 3, 3 because it uses less memory. It then prints your name as you select the relevant letters, followed by the symbol >. Line 220 repeats the procedure.

To stop the program, type BREAK.

USING THE PROGRAM

Type RUN, and the letter M will appear on the screen. If the first letter you require is in the first half of the alphabet, push the joystick to the left, and the letter on the screen will change to L, then K, and so on. Stop when you reach the letter you require, and press the 'fire' button. The computer then prints the letter you have selected. If the next letter you want is further on in the alphabet, push the joystick to the right and repeat the above procedure.

If you go beyond A or Z, you will find that the computer goes through other characters. In fact you can explore the whole range of characters on the computer going backwards and forwards by use of the joystick. If you go backwards beyond A, you will go through the numbers and various symbols and punctuation marks. You will find SPACE immediately before  on the ZX81 and ! on the Spectrum. Be careful not to go beyond SPACE, or the computer will register an error. If you go forwards beyond Z on the ZX81 there is a group of unused code numbers. This means that nothing will happen on the screen for a while.






However, persevere, and you will reach many interesting characters, including all the inverse letters. A full list of all these characters can be found in Appendix A of the Sinclair manual.

Project 4: Pack the Blob

This project is a game which tests your ability to erase a pattern on the screen against the clock.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 4 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM PACK THE BLOB
105 PRINT "      "
110 LET X=PI
115 LET Y=40
120 FOR T=0 TO 1E9
130 LET Z=USR INPT
140 LET B=Z <= 15
150 LET Z=Z+16*B
160 LET UD=(Z <= 27)-2*(Z <= 23)
170 LET Z=Z-2*UD+6*ABS UD
180 LET RL=(Z=29)-(Z=30)
190 UNPLOT X,Y
200 LET X=X+RL
210 LET Y=Y+UD
220 PLOT X,Y
230 PRINT AT PI,PI;T
240 NEXT T
```

**Note for Spectrum users: Line 190 should read: PLOT OVER 1;X,Y*

The REM statement is purely for your own reference. It can be omitted if you are short of memory. Line 105 prints the graphics characters between the quotation marks. This is the pattern which you will be deleting in the game. You can invent your own variation.

Lines 110 and 115 set the position of the cursor at the beginning of the

game. Lines 120, 230 and 240 are the timer.

Lines 130 to 180 have already been explained in the preliminary note earlier in this chapter. They translate the position of the joystick into a direction for the cursor to move on the screen.

The rubbing out mechanism is contained in line 190, and lines 200 and 210 make the cursor move in response to the joystick. (RL is the movement right-left, and UD is the movement up-down. They are set in lines 130 to 180). Line 220 plots a blob at the cursor position to show you where it is.

To stop the program, type BREAK.

USING THE PROGRAM

Place your hand in readiness on the joystick, and then type RUN and NEW LINE. The timer will start counting immediately. Erase the pattern as quickly as possible and when the screen is clear, apart from the cursor blob, type BREAK to record your time.

You must not go off the top of the screen or else the computer registers an error and you must begin again.

As an extension of this project, modify the program so that it is impossible for the cursor to leave the screen.

CHAPTER 4

CONTROLLING A SET OF SWITCHES

This chapter is concerned with the TE12, which is a four-channel relay unit. It is pictured below.

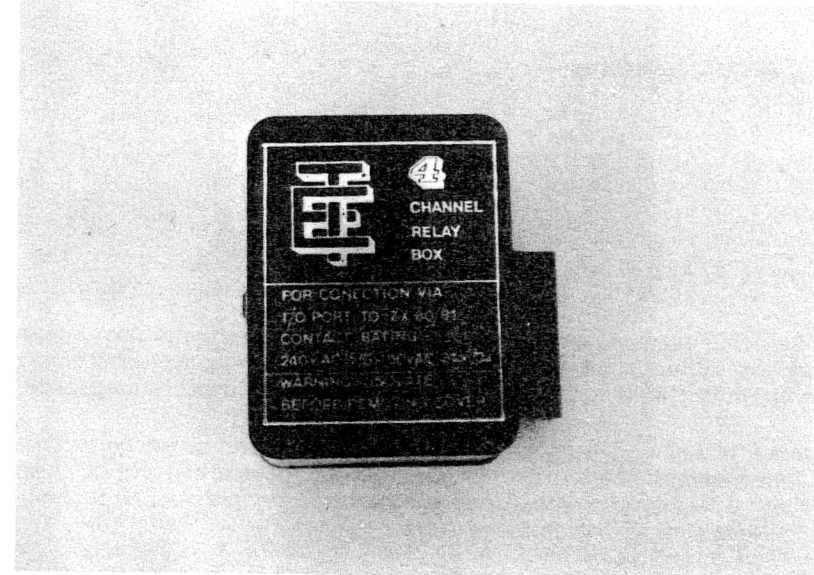


Figure 12: (Photograph of TE12)

The TE12 four-channel relay unit.

Inside the unit are four switches (relays) which can be controlled by the computer. The projects in this chapter illustrate how these can be used to turn on and off an external device such as a light.

The circuit diagram of the TE12 is shown below:

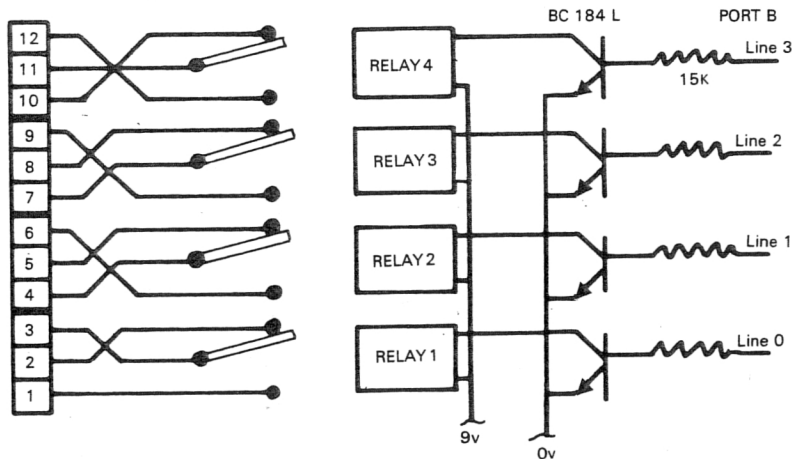


Figure 13: (Circuit diagram)

Getting Started

The TE12 relay unit has connections brought out to a terminal block inside the box. Although the unit will work without leads connected to this block, in order to use the relays to control external units you will require leads.

Hence you should remove the back of the TE12 and screw wires into connections numbered 1 and 3. The other connections can also be used when you want to control several external devices. If you make your own TE12, be sure to include these leads.

The TE12 relay unit is used in conjunction with an input/output controller.

To connect these to the computer, you should begin by ensuring that the power supply is disconnected. The controller is then plugged in at the back of the Sinclair. The TE12 is plugged in at the side of the TE10 controller. The Sinclair add-ons, such as the memory and the printer, which normally plug into the back of the computer can no be plugged into the back of the controller if desired. The photograph below shows the units properly connected.

Once you have checked that these are connected properly, the power supply may be switched on.

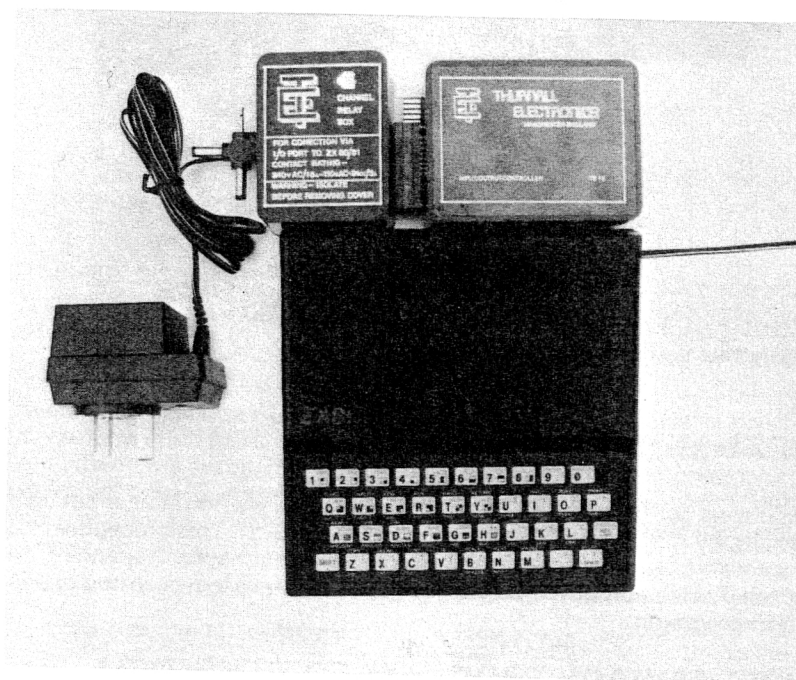


Figure 14a: Photograph of ZX81, TE10 and TE12



Figure 14b: Spectrum TE130 and TE12.

PRELIMINARY PROGRAM

Before we can use the TE12 to obey a command, we first have to set up a system by which the ZX81 can control the add-on. This procedure is a necessary preliminary step in all the projects in this chapter. Depending on which Sinclair computer you are using, you will have to type in one of two short programs.


PRELIMINARY PROGRAM FOR THE ZX81


This consists of the following five lines of instruction, and you should begin by typing these into the ZX81. You should copy it exactly as it is written here.

One important point to note is that in the first line (the REM statement) you should not type any spaces between the characters except where we have written (SPACE). In the other lines the use of spaces is ignored by the

computer, but you may wish to employ them for your own benefit in reading the display.

Where a group of letters is **underlined**, this means you should type them as one character as shown on the keyboard. Where they are not **underlined**, you should spell out the words by typing each individual letter.



If you have difficulty in finding any character on the keyboard, there is an index at the back of the ZX81 manual. Be careful with  (which is GRAPHIC SHIFT/Y), and ** in line 5 (which is SHIFT/H).

```
1 REM Y?PEEK XE>RND777Y(SPACE)  PEEK XTAN
2 LET OUT=16514
3 POKE OUT+14,95
4 POKE OUT+3,127
5 LET ZZ=2**15
```

The Preliminary Program.

The REM line is a set of instructions in machine code which allow the computer to control the TE10. A full explanation of how this works is given in the Appendix.

When you have typed these instructions, check your display in the TV screen and then type RUN (and NEW LINE, of course). You will notice that if you type NEW LINE again to obtain a listing of the program, the REM statement will be slightly different. This is because the computer has executed lines 3 and 4 in the program, which are designed to insert characters there which could not otherwise be put in a REM statement. Hence the listing should look like this:

```
1 REM Y?PEEK  E>RND777Y  PEEK ?TAN
2 LET OPT=16514
3 POKE OPT+14,95
4 POKE OPT+3,127
5 LET ZZ=2**15
```

How the Preliminary Program appears after typing RUN.

If it does not look like this, then a mistake has been made. The easiest way to rectify this is to unplug the power supply to the computer and begin afresh.

PRELIMINARY PROGRAM FOR THE SPECTRUM

The preliminary program for the Spectrum is as follows:

```
1 LET OPT=USR "A": RESTORE
2 FOR I=OPT TO OPT+15
3 READ A: POKE I,A
4 NEXT I
5 DATA 62,15,211,127,42,77,92,35,35,35,62,0,134,211,95,201
6 LET Z = 2115
```

The Preliminary Program.

This preliminary program sets up in the part of the memory reserved for graphics characters a short machine code routine. This routine allows the computer to operate the input/output controller. An explanation of how this works is given in the Appendix.

Project 1: Switching a Torch On and Off

In this project the computer switches a light on or off under your direction. It uses only one of the four relays.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```
100 REM CONTROL A TORCH
110 PRINT AT 0,0;"TYPE 1 FOR ON, 0 FOR OFF"
120 INPUT D
130 LET A=ZZ+D
140 LET A=USR OPT
150 GOTO 110
```

Preliminary Program for Switching a Torch On and Off to use in conjunction with the Preliminary Program.

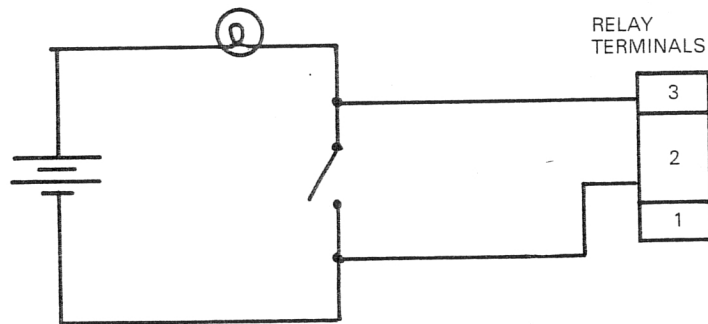
The REM statement is purely for your own reference. Line 110 prints the instruction to you on the screen to type either 1 or 0. Line 120 puts that number into the variable called D. 130 and 140 instruct the computer to use the preliminary program and switch the torch on or off. (This is more fully explained in the Appendix.) Line 150 allows you to repeat the process automatically.

Now type RUN. As you are asked on the screen, choose 1 or 0. Each time you change the number you should hear a soft click as the relays switch.

USING THE PROGRAM

Check that your torch functions normally, and leave it switched off. The relay unit has to be incorporated into the circuit of the torch. The most convenient place to do this is usually across the switch terminals as

shown in the figure below. Check that the light works, and switch it off again.



(Circuit diagram for project 1)

Then type in 0 or 1 to switch the torch on and off.

You can use this as a bedside light, for playing games or to flash Morse code.

Project 2: Controlling Several Lights

In this project we use the computer to switch up to four lights on or off, either under your direction or at random.

In the section entitled 'Getting Started' at the beginning of the chapter we told you how to connect one relay to the outside of the box. For this project you will need to connect up to four relays. For the first relay you connected wires to the screw terminals numbered 1 and 3. For the second, wires should be connected to terminals numbered 4 and 6; for the third, 7 and 9; and for the fourth 11 and 12.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). If you wish to keep the lights under your own direction, you should follow that by the same program as that given in project 1 of this chapter. The only difference is that the PRINT statement in line 110 should read:

```
110 PRINT AT 0,0;"TYPE A NUMBER BETWEEN 0 AND 15"
```

You then use the program as in the previous project to control four lights instead of one. The lights give a binary representation of the number you type in.

If you wish the lights to flash at random at a fixed speed, then you should follow the preliminary program with the following:

```
100 REM FLASHING LIGHTS
110 PRINT "TYPE SPEED"
120 INPUT SPEED
130 PAUSE SPEED
140 LET D=15*RND
150 LET A=ZZ+D
160 LET A=USR OPT
170 GOTO 130
```

Program for Flashing Lights to use in conjunction with the Preliminary Program.

The REM statement is purely for your own reference. Line 110 prints the instruction to you to type an ordinary decimal number, which tells the computer how fast to change the light patterns. Line 120 puts that

number into the variable called SPEED, and line 130 instructs the computer to pause for that time. Line 140 tells the computer to multiply 15 by a random number of the computer's choice between 0 and 1. The result is stored in the variable called D, and lines 150 and 160 output this to the TE12. (These two lines are more fully explained in the Appendix.) Line 170 instructs the computer to repeat the process.

Now type RUN. As you are asked on the screen, choose a number. We suggest you start with 20. The light shown will then operate.

To stop the program type BREAK.

Project 3: An Alarm Clock

This project uses the TE12 and any simple transistor radio to make an alarm clock. When the alarm goes off the radio is switched on. If the radio is physically near the computer the result will be a loud buzzing noise, but if further away it may be possible to receive the normal radio station.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```
100 REM ALARM CLOCK
110 LET D=0
120 LET A=ZZ+D
130 LET A=USR OPT
140 PRINT "HOW MANY MINUTES?"
150 INPUT MINS
160 FOR T=1 TO MINS
170 PAUSE 3000 (3600 IN THE USA)
180 NEXT T
190 LET D=15
200 GOTO 120
```

Program for An Alarm Clock to use in conjunction with the Preliminary Program.

The REM statement is purely for your own reference. Lines 110 and 130 output the number 0 to the relay unit, thereby switching off the radio. (This is explained more fully in the Appendix). Line 140 prints the instruction to you to type in a time in minutes, the period you wish to elapse before the alarm goes off. Line 150 inputs this number into the computer, and 160 to 180 are the waiting mechanism. PAUSE 3000 (3600 in the USA) causes the computer to wait for a complete minute, and this is repeated until your number is reached. Line 190 sets D to 15, and because of the jump in line 200, this number is output to the relay unit. This switches all the relays on and sets the alarm off.

USING THE PROGRAM

As in project 1 of this chapter, you need to connect up one relay, as described in the section entitled 'Getting Started'. Switch your transistor radio on to full volume and check that it is working. Now connect the wires from the relay unit so that they interrupt the power from the battery.

Now type RUN. Input 1 for the number of minutes so that you can check that the mechanism is working. After a minute, you should hear a soft click as the relays change, and the radio should make a noise. To stop it, simply switch the radio off at its switch.

Once checked, you can set the alarm clock to go off in the morning if you wish. If you switch off the television this will not affect the alarm clock and in the interests of safety you are recommended to switch off any unattended TV.

CHAPTER 5

USING A TRANSISTOR DRIVER

This chapter is concerned with the TE15 transistor driver unit, which is very much a unit for the electronics experimenter. It is pictured below:

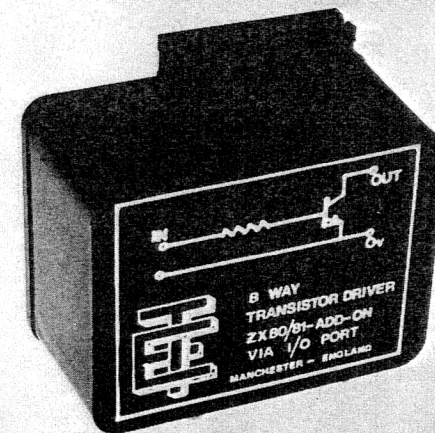


Figure 15: The TE15 Unit

The TE15 transistor driver unit.

The unit contains eight driver transistors (of the BC184L type. They are NPN transistors). Each transistor can withstand 30 volts and pass 300 mA, with a maximum power dissipation of 300 mW. A circuit diagram is shown below. As can be seen, the transistors can be turned on and off by the computer.

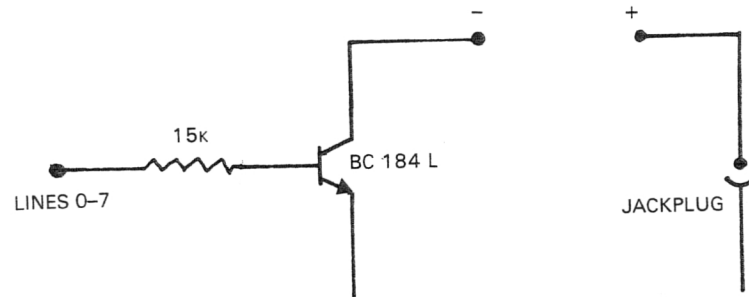


Figure 16: Circuit Diagram of TE15 Driver Unit

As you are contemplating using a transistor driver, we assume that you have a certain degree of electronics knowledge. You will need to do some soldering of any components you wish the transistor driver to control, but in this section we are purely concerned with how the computer controls the transistors in the TE15.



Figure 17a: ZX81, TE10 and TE15 and power supply, showing power supply inputs all connected

Getting Started

The TE15 transistor driver unit is used in conjunction with an input/output controller. To connect these to the computer, you should begin by ensuring that the power supply is disconnected. The controller is then plugged in at the back of the Sinclair. The TE15 is plugged into the side of the controller. The Sinclair add-ons, such as memory and the printer, which normally plug into the back of the computer, can now be plugged into the back of the controller if desired. The power supply for the indicator unit is plugged into the TE15, as shown in the photograph.



Figure 17b: Spectrum TE130 and TE15 and power supply, showing power supply inputs all connected.

The units properly connected.

Once you have checked that these are connected properly, the power supplies may be switched on.

THE PRELIMINARY PROGRAM

The program consists of the following five lines of instruction, and you should begin by typing these exactly as they are written here.

PRELIMINARY PROGRAM FOR THE ZX81

One important point to note is that in the first line (the REM statement) you should not type any spaces between the characters except where we have written (SPACE). In the other lines the use of spaces is ignored by the computer, but you may wish to employ them for your own benefit in reading the display. Where a group of letters is underlined, this means you should type them as one character as shown on the keyboard. Where they are not underlined, you should spell out the words by typing each individual letter. If you have difficulty in finding any character on the keyboard, there is an index at the back of the ZX81 manual. Be careful with ▣ (which is GRAPHIC, SHIFT/Y), and ** in line 5 (which is SHIFT/H).

```
1 REM Y?PEEK XE>RND777Y(SPACE)▣PEEK XTAN
2 LET OPT=16514
3 POKE OPT+14,95
4 POKE OPT+3,127
5 LET ZZ=2**15
```

The Preliminary Program.

The REM line is a set of instructions in machine code which allow the computer to control the TE10. A full explanation of how this works is given in the Appendix.

When you have typed these instructions, check your display on the TV screen and then type RUN (and NEW LINE, of course). You will notice that if you type NEW LINE again to obtain a listing of the program, the REM statement will be slightly different. This is because the computer has executed lines 3 and 4 in the program, which are designed to insert characters there which could not otherwise be put in a REM statement.

Hence the listing should look like this:

```
1 REM Y?PEEK KE>RND777Y XPEEK ?TAN
2 LET OPT=16514
3 POKE OPT+14,95
4 POKE OPT+3,127
5 LET ZZ=2**15
```

How the Preliminary Program appears after typing RUN.

If it does not look like this, then a mistake has been made. The easiest way to rectify this is to unplug the power supply to the computer and begin afresh.

THE PRELIMINARY PROGRAM FOR THE SPECTRUM

The preliminary program for the Spectrum is as follows:

```
1 LET OPT=USR "A": RESTORE
2 FOR I=OPT TO OPT+15
3 READ A: POKE I,A
4 NEXT I
5 DATA 62,15,211,127,42,77,92,35,35,35,62,0,134,211,95,201
6 LET XX = 2115
```

The Preliminary Program.

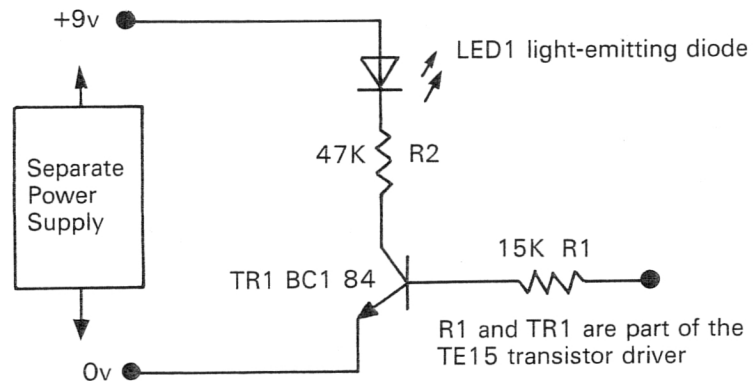
This preliminary program sets up in the part of the memory reserved for graphics characters a short machine code routine. This routine allows the computer to operate the input/output controller. An explanation of how this works is given in the Appendix.

Examples

The TE15 is a very flexible unit, and it can be used to control a variety of circuits of your choice. To get you started we give two simple examples below.

EXAMPLE 1: CONTROLLING 8 LIGHT-EMITTING DIODES

The following circuit diagram shows you how to connect one of 8 light emitting diodes to the TE15.



Circuit diagram for controlling up to 8 light-emitting diodes

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this example you should follow that by typing in the program below:

```

100 REM LED CONTROLLER
110 PRINT "NUMBER BETWEEN 0 AND 255"
120 INPUT DATA
130 LET A=ZZ+DATA
140 LET A=USR OPT
150 GOTO 120
    
```

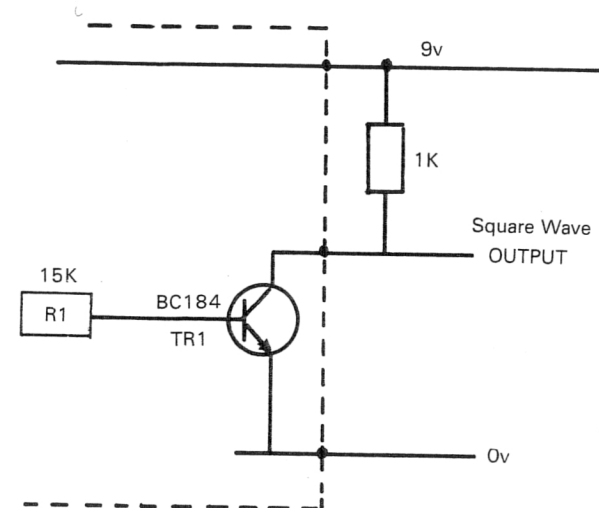
Program for LED Controller to use in conjunction with the Preliminary Program.

The REM statement is purely for your own reference. Line 110 prints the instruction to you to type a decimal number between 0 and 255, and 120 puts that number into the variable called D. In lines 130 and 140 this number is output to the transistor driver in binary, thereby switching on the corresponding diodes. These lines are more fully explained in the Appendix. Because of line 150 the program repeats the process.

EXAMPLE 2: A SQUARE-WAVE GENERATOR

The following circuit diagram shows you how to connect a resistor to the TE15 to make a square-wave generator.

Note that a separate power supply is required.



Circuit diagram for a square-wave generator. (R1 and TR1 are part of the TE15 transistor driver)

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this example you should follow that by typing in the program below:

```

100 REM SQUARE WAVE GENERATOR
110 PRINT "TYPE SPEED"
120 INPUT SPEED
130 LET DATA=1
140 PAUSE SPEED
150 LET A=ZZ+D
160 LET A=USR OPT
170 LET D=NOT D
180 GOTO 140

```

Program for Square-Wave Generator to use in conjunction with the Preliminary Program

The REM statement is purely for your own reference. Line 110 prints the instruction to you to type a decimal number which will dictate how fast the square-wave generator will oscillate. Line 120 inputs this number into the computer, and line 140 is the pause statement which makes the computer wait for the time you specify. The main loop consists of lines 140 to 180. Line 130 initially sets D to 1, and line 170 in the loop makes its value alternate between 0 and 1. Lines 150 and 160 in the loop output the number D to the TE15 and are explained more fully in the Appendix.

CHAPTER 6

CONTROLLING SEVERAL CHANNELS SIMULTANEOUSLY

The input/output controller for the Sinclair computers is capable of controlling two channels simultaneously. This chapter is concerned with ways of using both channels, either to control several add-ons simultaneously, or to control more sophisticated add-ons.

To control several add-ons you need to use an extender card called a "motherboard". You can easily construct your own board (which is simply a distribution board) or hard-wire the extra add-ons directly to your computer.

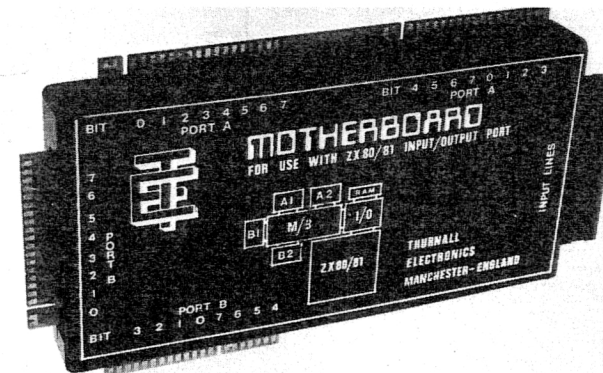


Figure 18: The TE30 Unit

For those with enough knowledge of electronics to build simple circuits, the last part of this chapter also shows you how to use an eight-channel analogue-to-digital converter. This unit requires you to control both channels in order to use it. It is pictured below and also given is its circuit diagram.

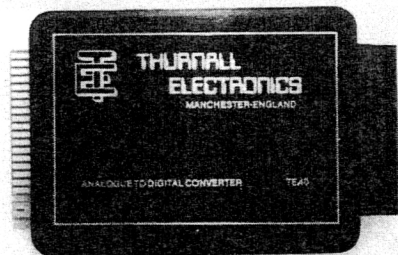


Figure 19: A/D Converter

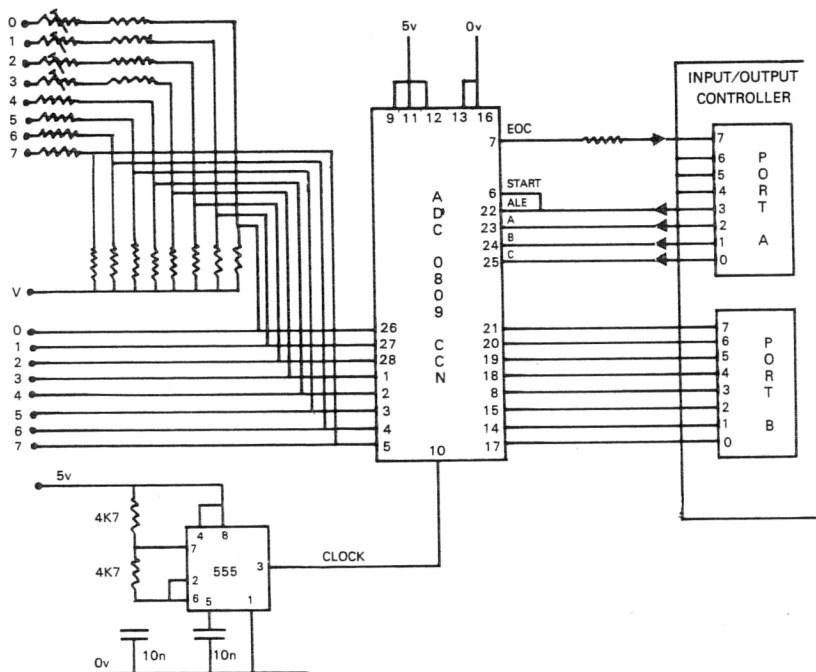


Figure 20: Circuit Diagram of A/D Converter

Getting Started

As described in previous chapters, you must switch off the power supply before connecting up the add-ons. To remind, the controller is plugged in at the back of the computer, and the TE30 motherboard is connected to the side of the controller. Up to four add-ons can then be attached to the edge connectors of the motherboard.

THE PRELIMINARY PROGRAM

The program consists of the following five lines of instruction, and you should begin by typing these in exactly as written here. Remember not to include spaces except where specifically instructed.

PRELIMINARY PROGRAM FOR THE ZX81

```

1 REM Y=JPEEK X<=XM RND ((SPACE)(SPACE) TAN
2 REM Y?PEEK XE >RND77Y (SPACE) PEEK XTAN
3 LET INPT=16514
4 LET OPT=16534
5 LET ZZ=2**15

```

The Preliminary Program.

The REM statement in lines 1 and 2 are the same as those used in earlier chapters for input and output respectively. A full explanation of the machine code is given in the Appendix.

PRELIMINARY PROGRAM FOR THE SPECTRUM

The preliminary program for the Spectrum is as follows:

```

1 LET OPT=USR "A": RESTORE
2 FOR I=OPT TO OPT+15: READ A: POKE I,A: NEXT I
3 DATA 62,15,211,127,42,77,92,35,35,35,62,0,134,211,95,201
4 LET INPT=USR "C"
5 FOR I=INPT TO INPT+10: READ A: POKE I,A: NEXT I
6 DATA 62,176,47,211,127,219,95,6,0,79,201
7 LET ZZ = 2115

```

The Preliminary Program.

This preliminary program is a combination of those used for input and output in earlier chapters. An explanation of the machine code can be found in the Appendix.

The Motherboard Programs

There now follow four alternative pieces of BASIC program, one or more of which has to be used in any program you devise for the motherboard.

The four choices are: input port A; input port B; output port A; output port B.

```

100 REM INPUT PORT A INTO VARIABLE Z
110 POKE INPT+4,111
120 POKE INPT+6,79
130 LET Z=USR INPT

```

Program to Input a Number from Port A, to use in conjunction with the Preliminary Program.

```

200 REM INPUT PORT B INTO VARIABLE Z
210 POKE INPT+4,127
220 POKE INPT+6,95
230 LET Z=USR INPT

```

Program to Input a Number from Port B, to use in conjunction with the Preliminary Program.

```

300 REM OUTPUT VARIABLE D TO PORT A
310 POKE OPT+3,111
320 POKE OPT+14,79
330 LET A=ZZ+D
340 LET A=USR OPT

```

Program to Output a Number (D) to Port A, to use in conjunction with the Preliminary Program.

```

400 REM OUTPUT VARIABLE D TO PORT B
410 POKE OPT+3,127
420 POKE OPT+14,95
430 LET A=ZZ+D
440 LET A=USR OPT

```

Program to Output a Number (D) to Port B, to use in conjunction with the Preliminary Program. (D must be supplied to the Program).

You should note that there are two edge connectors for each Port. One is connected to lines 0 through 7 in sequence, and the other goes from 4 through 7, and then 0 through 3.

If you are using an eight line add-on (such as the switch unit or the indicator unit) then you should take care to plug it in to the edge connector whose numbers are in sequence. If you are using four-line units (such as the relay unit or the joystick if you do not press the 'fire' button) then you can plug one unit into each edge connector of each Port, as one will use lines 0 through 3 and the other will use lines 4 through 7.

Examples

EXAMPLE 1: INPUT AND OUTPUT

This project inputs a number from a unit attached to port A, and outputs that number to the unit attached to port B. For this example we have chosen the TE17 eight-way switch unit for input, and the TE18 eight-way indicator unit for output. The same program can be used with for example two TE12 four-channel relay boxes in place of the indicator unit, or whenever you need one input and one output add-on.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```
100 REM INPUT PORT A INTO VARIABLE D
110 POKE INPT+4,111
120 POKE INPT+6,79
130 LET D=USR INPT
400 REM OUTPUT VARIABLE D TO PORT B
410 POKE OPT+3,127
420 POKE OPT+14,95
430 LET A=ZZ+D
440 LET A=USR OPT
500 GOTO 100
```

Program for Input and Output, to use in conjunction with the Preliminary Program.

The REM statements in lines 100 and 400 are purely for your own reference and they can be omitted if you are short of memory. Lines 110 to 130, and 410 to 440 were outlined in the previous section of this chapter, on the preliminary program. Line 500 makes the program loop so that it continually reads port A and outputs the number to port B.

Using the TE17 and TE18, the lights should mimic the positions of the switches. This is a simple demonstration of the computer using input and output at the same time. If you use the relay units instead, then you can use one relay unit for each of the port B outputs. The relays will then mimic the positions of the switches and could be used for example to switch external lights on an off.

EXAMPLE 2: SIXTEEN CHRISTMAS-TREE LIGHTS

This project uses the motherboard and four relay units to control sixteen lights. These may be used for example on a Christmas tree. This program can be used whenever you wish to use all output add-ons.

The relays in the TE12 relay box are rated up to 240 volts, 1.5 amps.

Whilst we would advise you to use battery operated lights, this means that you could use a mains supply with the unit. However, we would issue the following WARNING. Under no circumstances should you attempt to use mains power with the TE12 unless you have a good working understanding of the dangers of electricity. MAINS ELECTRICITY CAN KILL.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the program below:

```
100 REM CHRISTMAS TREE LIGHTS
300 REM OUTPUT A RANDOM NUMBER TO PORT A
305 LET D=INT (256*RND)
310 POKE OPT+3,111
320 POKE OPT+14,79
330 LET A=ZZ+D
340 LET A=USR OPT
400 REM OUTPUT A RANDOM NUMBER TO PORT B
405 LET D=INT (256*RND)
410 POKE OPT+3,127
420 POKE OPT+14,95
430 LET A=ZZ+D
440 LET A=USR OPT
500 PAUSE 20
510 GOTO 100
```

Program for Sixteen Christmas Tree Lights, to use in conjunction with the Preliminary Program.

The REM statements in lines 100, 300 and 400 are purely for your own reference and they can be omitted if you are short of memory. Line 305 sets D to be a random number between 0 and 255. The INT ensures it is a whole number. Lines 310 to 340 output this to port A, as outlined in the previous section of this chapter on the preliminary program. Lines 405 to 440 do the same for port B. In line 500 the computer pauses before commencing another random pattern as instructed by line 510.

To see how to connect the lights up to the relay box see chapter 5, projects 1 and 2. If you have practical experience with mains electricity (and ONLY IF YOU DO) then each light should be connected in series with its relay and to the mains power. But please to be careful.

EXAMPLE 3: A GAME OF CHASE

This project uses the motherboard and two joysticks to make a simple game for two players. One player chases and tries to catch the other on the screen.

THE PROGRAM

You will already have typed in the preliminary program of lines 1 to 5 (see the beginning of this chapter). For this project you should follow that by typing in the following program:

```
100 REM ZX CHASE
110 REM INITIALIZE
120 LET X1=0
130 LET Y1=X1
140 LET X2=PI
150 LET Y2=PI

200 REM MOVE PLAYER 1 ON PORT A
210 POKE INPT+4,111
220 POKE INPT+6,79
230 UNPLOT X1,Y1 or PLOT OVER 1;X1,Y1 (Spectrum)
240 GOSUB 1000
250 LET X1=ABS (X1+RL)
260 LET Y1=ABS (Y1+UD)
270 PLOT X1,Y1

300 REM MOVE PLAYER 2 ON PORT B
310 POKE INPT+4,127
320 POKE INPT+6,95
330 UNPLOT X2,Y2 or PLOT OVER 1;X2,Y2 (Spectrum)
340 GOSUB 1000
350 LET X2=ABS (X2+RL)
360 LET Y2=ABS (Y2+UD)
370 PLOT X2,Y2

400 REM HAS ONE PLAYER CAUGHT THE OTHER?
410 IF X1=X2 AND Y1=Y2 THEN STOP
420 GOTO 200
```

```
1000 REM DECODE JOYSTICK
1010 LET Z=USR INPT
1020 LET B=Z<=15
1030 LET Z=Z+16*B
1040 LET UD=(Z<=27)-2*(Z<=23)
1050 LET Z=Z-2*UD+6*ABS UD
1060 LET RL=(Z=29)-(Z=30)
1070 RETURN
```

Program for a Game of Chase, to use in conjunction with the Preliminary Program.

The REM statements in lines 100, 110, 200, 300, 400 and 1000 are purely for your own reference. The program is sufficiently large that you will require 16k of memory to run it.

Lines 120 to 150 set the starting positions for the opposing players' cursors. Lines 210 and 220 set the port A to input, which is for player 1. In line 230 the first player's position is unplotted and the new position is registered by line 270. To find this new position, line 240 sends the computer to lines 1000 to 1070, which decode the movements indicated by the joystick. See chapter 4 for an explanation of this section of program.

Lines 250 and 260 calculate the new position for the cursor.

Lines 300 to 370 repeat this process for player 2 on port B.

When one player's cursor catches the other, line 410 stops the game. Line 420 returns the program to line 200 to continue the chase, if no player has caught the other.

EXAMPLE 4: ANALOGUE TO DIGITAL CONVERTOR (See Figures 19 and 20)

The TE40 analogue to digital convertor is capable of reading up to eight different voltage levels. A voltage of zero gives a reading of zero, and 5V gives a reading of 255. For voltages between 0 and 5 volts, the reading is scaled proportionately.

A simple way of applying a voltage to input the number 0 is to use a variable resistor in the circuit shown below:

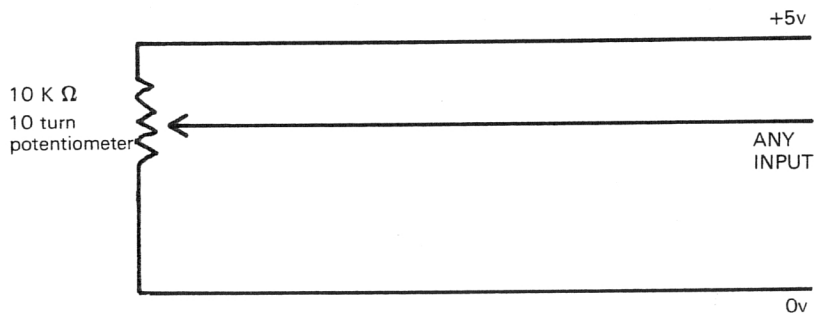


Figure 21: Circuit diagram of a test box

This unit is intended for use by those with some knowledge of electronics.

You can apply your own voltages which you will want to read into the computer, to any of the eight inputs.

The steps in using the analogue to digital convertor are as follows:

1) Output Channel Number to Port A

The channel number must be between 0 and 7. It selects which input is to be read.

2) Output (Channel Number +8) to Port A

This sets the convertor into operation.

3) Wait until Port A Reads 128 in Input Mode

When the convertor has finished converting, it sends a signal into Port A.

4) Read the Number from Port B

This is the voltage to be read.

The program below performs these tasks:

```

9 REM N IS CHANNEL NUMBER
10 INPUT N
20 LET D=N
30 GOSUB 300
40 LET D=N+8
50 GOSUB 300
60 GOSUB 100
70 IF Z<128 THEN GOTO 60
80 GOSUB 200
90 PRINT "READING IS", Z
95 STOP

100 REM INPUT PORT A INTO VARIABLE Z
110 POKE INPT+4,127
120 POKE INPT+6,79
130 LET Z=USR INPT
140 RETURN

200 REM INPUT PORT B INTO VARIABLE Z
210 POKE INPT+4,127
220 POKE INPT+6,95
230 LET Z=USR INPT
240 RETURN

300 REM OUTPUT VARIABLE D TO PORT A
310 POKE OPT+3,111
    
```

```
320 POKE OPT+14,79
330 LET A=ZZ+D
340 LET A=USR OPT
350 RETURN
```

Appendix 1

MACHINE CODE

At various points in the text you will have been referred to this appendix for a fuller explanation of the machine code used in the preliminary programs.

The BASIC language used by the Sinclair computers does not contain an instruction by which the computer can control the input/output port and other add-ons. The preliminary programs therefore have to bypass BASIC and instruct the microprocessor chip directly; and the language used to do this is called 'machine code'.

The machine code in the ZX81 has been placed within REM statements because these are the only lines which the ZX81 automatically ignores in BASIC. In the Spectrum, machine code is stored in the 'user defined graphics' memory area.

If you wish to learn about machine code in detail, two introductory books on the subject are:

'Programming the Z80' by Rodney Zaks and
'Z80 and 8080 Assembly Language Programming' by Rathe Spracklen.

It is not essential however to understand machine code fully in order to use the programs in this book.

For those with some knowledge of machine code, the following listings show the machine code programs that we have used in this book.

One point to note is that the programs have been written with a view to ease of entry into the REM statements. This is because some characters (such as 4F) cannot be entered directly into these statements.

THE KEY STEPS IN THE MACHINE CODE PROGRAMS

The chip in the TE10 has two ports, A and B; numbers can be input and output using either port. The add-ons which we illustrate in this book are all usually wired to use port B. The motherboard gives access, however, to both ports.

The numbers are in binary (8 bit) form, but for our convenience, we give them in hexadecimal notation.

Each port of the chip can be used either for input or output of a number. To do this there are two fundamental programming steps: first, you have to instruct the chip which port to use and whether you wish to input or to output, and second to perform the input or output operation.

TO INPUT A NUMBER

The two key steps in the machine code are:

- 1) Set port B to input mode.

```
LD a,4F      Load the accumulator with 4F (hexadecimal)
OUT A,7F     Output 4F to address 7F
              (If using port A, substitute 6F for 7F)
```

- 2) Read the input.

```
IN A,5F      Read the input into the accumulator
              (If using port A, substitute 4F for 5F)
```

TO OUTPUT A NUMBER

The two key steps in machine code are:

- 1) Set port B to output mode.

```
LD A,0F      Load the accumulator with 0F
OUT A,7F     Output 0F to address 7F
              (If using port A, substitute 6F for 7F)
```

- 2) Output the number

```
OUT A,5F     The accumulator must contain the output number.
              (If using port A, substitute 4F for 5F)
```

Here are the programs in more detail.

PROGRAM TO INPUT NUMBERS

Character	Mnemonic	Comment
Y	LD A,BO	
<input checked="" type="checkbox"/>	J	CPL A
	PEEK	OUT A,7F
X		Accumulator now contains 4F
<=		Output 4F to port's B buffer
X		Poke 7F into here in line 3
M	IN A<,5F	Read input into accumulator
<input checked="" type="checkbox"/>		Poke 5F into here in line 4
\$	LD(nn),A	Load the accumulator into here
RND		Pointer to this address
<input checked="" type="checkbox"/>	RND	
space	LD BC,nn	Load registers B and C
space	O	
space	O	
TAN	RET	Finish the routine

This program first tells the input/output port that it should set port B to input mode. This is achieved through the OUT A,7F with the accumulator containing the value 5F. Port B is then read into the computer by the IN A,5F instruction. This data is then transferred to the BC register.

The BASIC part of the program sets the variable IN to be the address of the first instruction in the REM statement (the Y). The function USR IN therefore executes this machine code routine. At the end of a USR function, its value is the number in the BC register at the end of the routine - in our case this is the reading from port B.

PROGRAM TO OUTPUT NUMBERS

Character	Mnemonic	Comment
Y	LD A, 0F	
?		
PEEK	OUT A, 7F	Output 0F to port's B buffer
X		Poke 7F in here from BASIC
E	LD HL, (DEST)	DEST points to the variable A which contains the number to be outputted
<		
RND		
7	INC HL	
7	INC HL	
7	INC HL	HL now points to the number which must be outputted
Y	LD A, 00	
space		
█	ADD A, (HL)	A contains number to be outputted
PEEK	OUT A, 5F	Output the number to the port
X		Poke 5F in here from BASIC
TAN	RET	Finish the routine

This program first tells the input-output port that it should set Port B to output mode. This is achieved by the OUT A, 7F instruction with the accumulator containing 0F. The program then uses the BASIC system variable called DEST to locate the variable A which contains the number to be output. This number is loaded into the accumulator and output to the port with the OUT a, 5F instruction.

The BASIC part of the program sets the variable OUT to be the address of the first instruction in the REM statement (the Y). LET A=USR OUT therefore executes the program and outputs the number A2**15 to the port B.

Appendix 2

Decimal and Hexadecimal

Number Systems

Normally, we count in tens, but microprocessors appear to count to the base of 8 or 16; this is because these number systems are more compatible with the electronic building blocks used today. The Z80 and all other microprocessors use the base 16 or hexadecimal system of numbering.

Strictly speaking, the microprocessor operates in the binary system Eg 11010010. But entering such a number is prone to error and is very consuming, hence the Octal and the Hexadecimal systems were devised to assist the operator, to reduce the number of key or switch operations.

The equivalent numbers are shown below.

<u>Decimal (10)</u>	<u>Hex (16)</u>
0	00
1	01
2	02
3	03
4	04
4	04
5	05
6	06
7	07
8	08
9	09

So far the numbers are the same, except that in all microprocessor work, numbers are always in pairs and therefore a leading 0 is used. Beyond 9, the picture changes:

10	OA
11	OB
12	OC
13	OD
14	OE
15	OF

Note that 10 to 15 are represented by single letters A to F, since if this was not done the number pair would be increased to 3 digits. Beyond 15 we have:

16	10
17	11

15 is the highest single number in the HEX system and so if we start again at 16 with a 0 and to show that we have counted up to 16 ONCE a 1 is used in front of the zero, hence 32 is 20, 48 is 30, 64 is 40 and so on. Arithmetical operations in HEX are quite straightforward.

<u>Decimal (10)</u>	<u>HEX (16)</u>
12	0C
+ 32	+ 20
<u>44</u>	<u>2C</u>

i.e. $(2 \times 16) + (12 \times 1) = 32 + 12 = 44$

29	1D
+ 18	+ 12
<u>47</u>	<u>2F</u>

i.e. $(2 \times 16) + (15 \times 1) = 32 + 15 = 47$

Subtraction is also quite simple; multiplication and division are rarely used in simple programs.

(This appendix originally appeared in "Z 80 Instant Programs" published by Sigma Technical Press)

A message from the publisher

Sigma Technical Press is a rapidly expanding British publisher. We work closely in conjunction with John Wiley & Sons Ltd. who provide excellent marketing and distribution facilities.

Would you like to join the winning team that published these highly successful books? Specifically, **could you successfully write a book that would be of interest to the new, mass computer market?**

Our most successful books are linked to particular computers, and we intend to pursue this policy. We see an immense market for books relating to such machines as:

DRAGON
THE BBC COMPUTER
APPLE
TANDY
SINCLAIR
ATARI
IBM PC
SIRIUS
NEWBRAIN
COMMODORE
and many others

If you think you can write a book around one of these or any other popular computer — or on more general themes — we would like to hear from you.

Please write to:

Graham Beech
Sigma Technical Press
5 Alton Road,
Wilmslow,
Cheshire, SK9 5DY,
United Kingdom.

Or, telephone 0625-531035