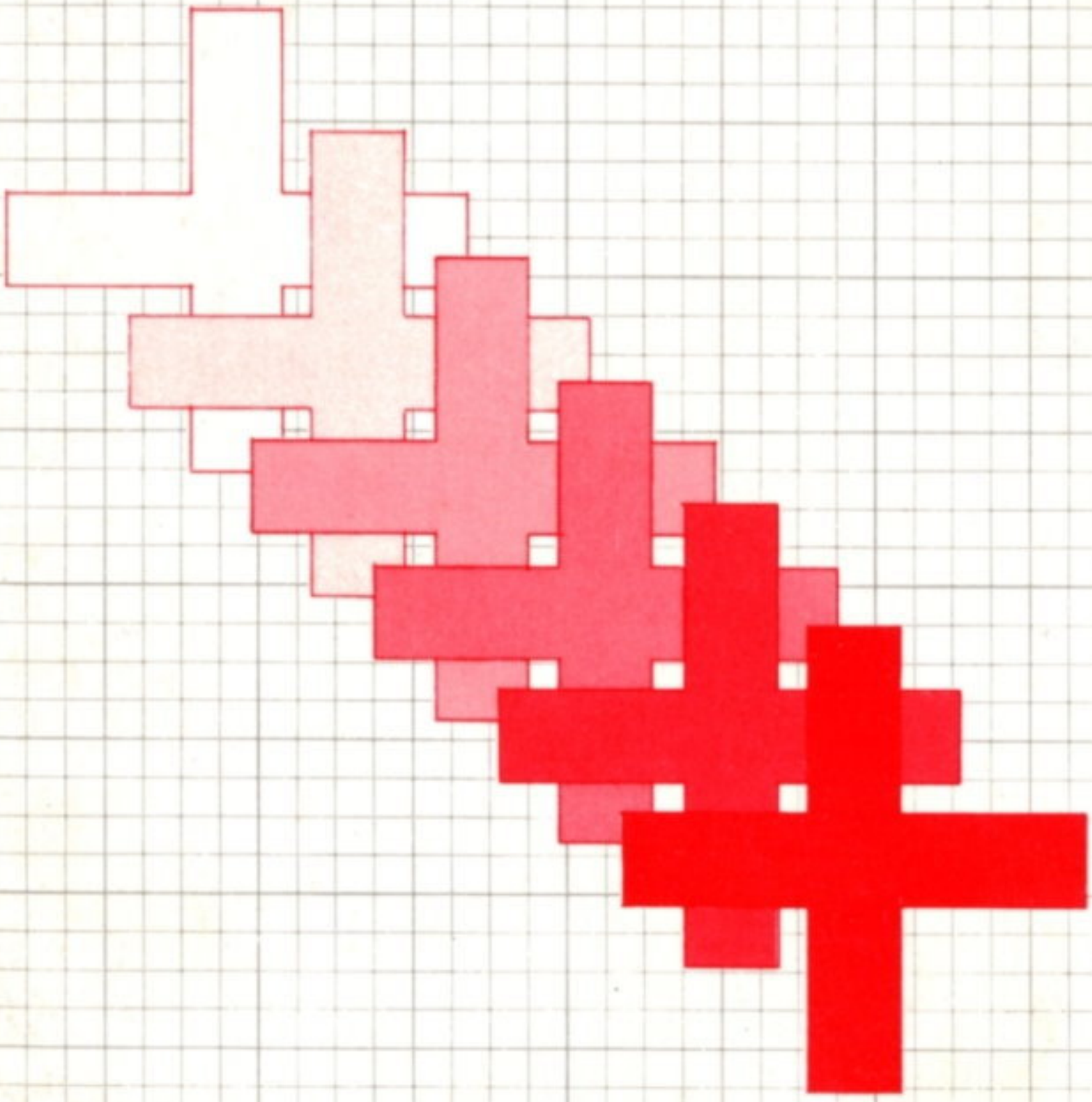
Easy Add-on Projects for Commodore 64, VIC-20, BBC Micro & Acorn Electron

OWEN BISHOP



Easy Add-on Projects for Commodore 64, VIC-20, BBC Micro and Acorn Electron

OTHER TITLES BY THE SAME AUTHOR

BP73 Remote Control Projects
BP82 Electronic Projects Using Solar Cells
BP104 Electronic Science Projects
BP124 Easy Add-on Projects for Spectrum, ZX81 & Ace

OTHER BOOKS OF INTEREST

BP130 Micro Interfacing Circuits – Book 1
BP131 Micro Interfacing Circuits – Book 2
BP135 Secrets of the Commodore 64
BP139 An Introduction to Programming the BBC Model B Micro
BP142 An Introduction to Programming the Acorn Electron
BP147 An Introduction to 6502 Machine Code

Easy Add-on Projects for Commodore 64, VIC-20, BBC Micro and Acorn Electron

by

Owen Bishop

BERNARD BABANI (publishing) LTD
THE GRAMPIANS
SHEPHERDS BUSH ROAD
LONDON W6 7NF
ENGLAND

PLEASE NOTE

Although every care has been taken with the production of this book to ensure that any projects, designs, modifications and/or programs etc. contained herein, operate in a corrrect and safe manner and also that any components specified are normally available in Great Britain, the Publishers do not accept responsibility in any way for the failure, including fault in design, of any project, design, modification or program to work correctly or to cause damage to any other equipment that it may be connected to or used in conjunction with, or in respect of any other damage or injury that may be so caused, nor do the Publishers accept responsibility in any way for the failure to obtain specified components.

Notice is also given that if equipment that is still under warranty is modified in any way or used or connected with home-built equipment then that warranty may be void.

All the projects in this book have been designed and tested by the Author using models of the various microcomputers that were available in Great Britain at the time of writing. The Author and Publishers accept no responsibility for failure of a project to operate or any damage that may be so caused, should the manufacturers of the microcomputers radically change the specification of their machines in the future or for a particular overseas market.

© 1985 BERNARD BABANI (publishing) LTD

First Published – March 1985

British Library Cataloguing in Publication Data
Bishop Owen
Easy add-on projects for Commodore 64, VIC-20, BBC Micro and Acorn Electron.
1. Computer interfaces
2. Microprocessors
I. Title
001.64'04 TK7887.5

ISBN 0 85934 109 7

Printed and Bound in Great Britain by Cox & Wyman Ltd, Reading

Contents

	Page
INTRODUCTION	1
Interfacing to Microcomputers	1
Logic Levels in Computers	2
User Port	2
Decoder	6
Programming Summaries	15
Project 1: PULSE DETECTOR	19
How It Works	19
Building It	21
Using It	21
Detecting a "High Pulse"	23
Project 2: PICTURE DIGITISER	24
How It Works	24
Building It	29
Final Check	33
Test Program	33
Programming	35
Project 3: FIVE-KEY PAD	41
How It Works	43
Building It	45
Testing	46
Project 4: MODEL CONTROLLER	50
How It Works	50
Building It	55
Programming	57
Project 5: BLEEPER	62
How It Works	62
Building It	64
Programming	67

	Page
Project 6: LAMP FLASHER	68
How It Works	68
Building It	71
Programming	71
Project 7: LIGHT PEN	73
How It Works	73
Building It	74
Programming	81
Project 8: MAGNETIC CATCH	85
How It Works	85
Building It	88
Programming	92
Project 9: LAP SENSOR	94
How It Works	94
Building It	97
Programming	98
Project 10: PHOTO-FLASH	101
How It Works	102
Programming	105
Project 11: GAMES CONTROL	109
How It Works	109
Building It	112
Programming	114
Project 12: RAIN DETECTOR	116
How It Works	116
Building It	117
Programming	118
Project 13: WEATHERCOCK	120
How It Works	120
Building the Vane	123
Building the Circuit	124
Programming	125

Page

Project 14: ANEMOMETER 127
How It Works 128
Building It 130
Programming 132

Project 15: THERMOMETER 135
How It Works 135
Building It 138
Calibration and Programming 139

Project 16: BAROMETER 143
How It Works 143
Building It 147
Programming 150

Project 17: SUNSHINE RECORDER 155
How It Works 155
Building It 158

Appendix A: THE DECODER 165
How It Works 165
Power Supplies 169
Building the Decoder 171
Assembling the P.C.B. 171
IC Sockets 177
Fitting the Case 177
Decoder/Computer Interconnectors 178
Testing 183

Appendix B: DECODER WITH AN INTERFACE FOR THE ACORN ELECTRON 185
How It Works 185
Testing the Electron Version of the Decoder 186
Programming 187

Appendix C: SUPPLIERS OF P.C.B.s AND KITS 188

Appendix D: PIN AND LEAD-OUT DETAILS 190

Introduction

Interfacing to Microcomputers

This book describes how to build electronic circuits which you can attach to your VIC-20, Commodore 64, Acorn Electron or BBC Microcomputer. All the devices have been tested, most of them with most of the machines. Owners of the BBC Microcomputer Model A should note that they will need the User Port upgrade for their machine to allow it to operate with the devices in this book. For Project 11 they will need the Analogue Port upgrade too. All the devices have been tested with all of these computers.

All the projects are simple and inexpensive ones, requiring only a few transistors or integrated circuits. The integrated circuits are of the less expensive kind. The wiring required for each circuit has been kept to a minimum, an unusual feature for a book of this kind, for microprocessor projects tend to need rather more wires than projects of other kinds. The result of this simplified approach used here is that all the projects can be built easily and cheaply, even by a relative beginner. Once built, they are simple to operate. Writing programs to control them need not be complicated. Each project includes a simple program or two to get you started. Of course, those readers who are more expert at programming can have a lot of fun in writing elaborate programs based on these projects, but the beginner can start with the short program and perhaps add extra features later.

The circuit designs apply equally well to the VIC-20, the Commodore 64, the Acorn Electron and the BBC Microcomputer. If you have a VIC-20 now, for example, and later change to one of the other computers, the projects can be operated from your new computer without having to alter them. If you have previously built similar projects for the ZX81, Spectrum, or Jupiter Ace, as described in book number BP124, *Easy Add-on Projects for Spectrum, ZX81 and*

Ace, you can use most of these on the VIC-20, Commodore or BBC Microcomputer simply by connecting them to the decoder device described in Appendix A of this book.

Logic Levels in Computers

Computers deal with numbers and other information, as a series of 0s and 1s. The projects show many examples of this. The 0s and 1s, or binary digits (shortened to "bits") are represented in the computer by voltage levels.

0 is represented by 0 V, we refer to this as "low";
1 is represented by +5 V, we refer to this as "high".

Numbers and other kinds of information, including instructions, are handled in the computer by making one of these two voltages (0 V or +5 V) appear on different wires (or lines).

User Port

Although these computers may differ in appearance and in their facilities, their microprocessor (the 6502) is identical in each. In addition, the VIC-20 and BBC Microcomputer both have a 6522 Versatile Interface Adapter (VIA) for communicating with the outside world at the user port. The Commodore 64 has a 6526 Complex Interface Adapter (CIA) chip instead of the 6522. The 6526 has very many features in common with the 6522 and, for the purposes of the circuits described in this book, can be considered to be identical.

The user ports of all three computers have connections to port B of the VIA (or CIA), as shown in Figs. 0.1 and 0.2. In this book we use only the 8 data lines of port B, numbered from PB0 to PB7. In effect, these are extensions of the data bus of the computer, except that there is the VIA or CIA to control the transfer of data between the data bus and any device plugged into the user port. We will see how the VIA or CIA does this in a moment. Another terminal of the user port which we use is the ground terminal, or 0 V line. The 0 V line of all devices plugged in to the user port is always con-

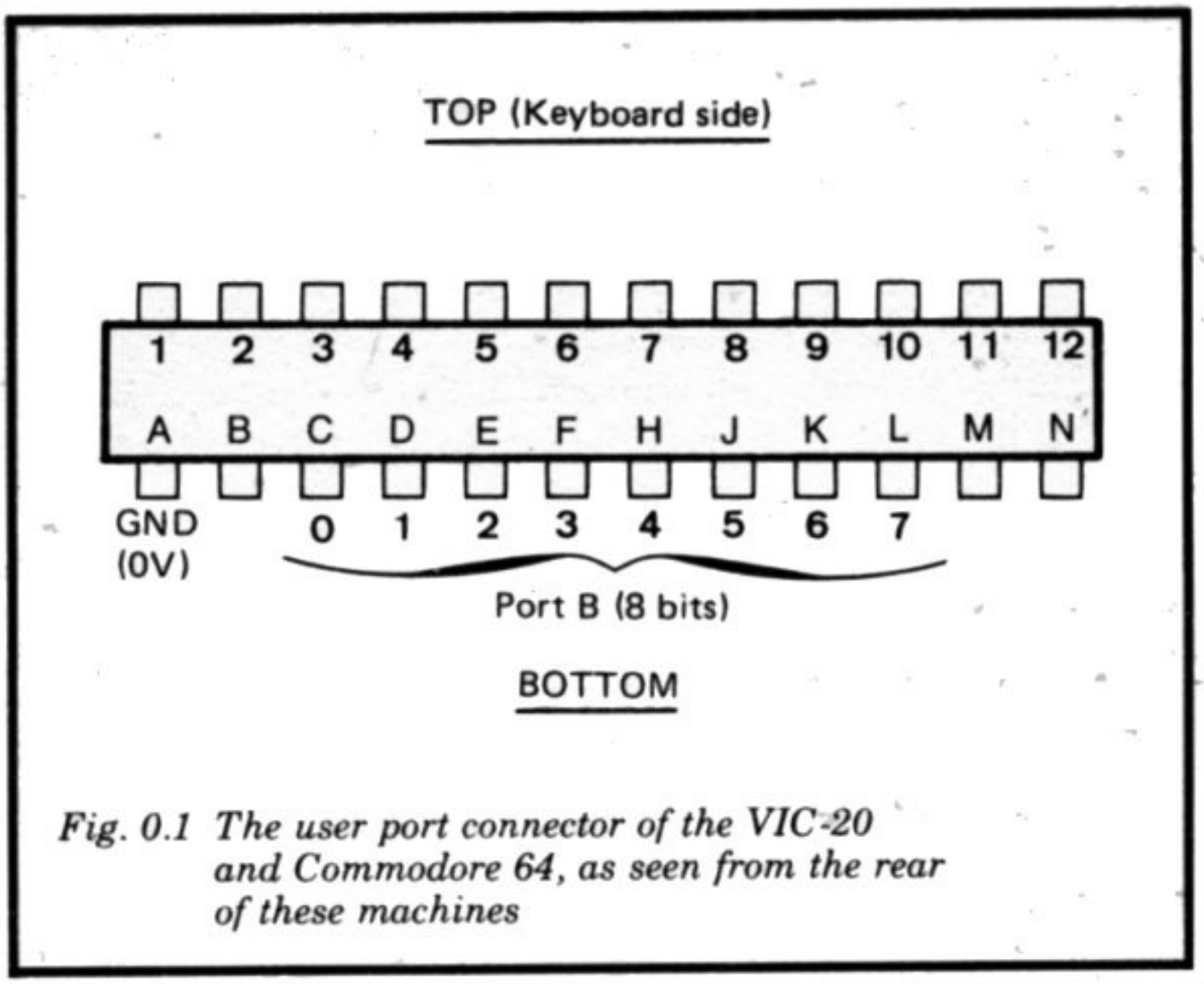


Fig. 0.1 The user port connector of the VIC-20 and Commodore 64, as seen from the rear of these machines

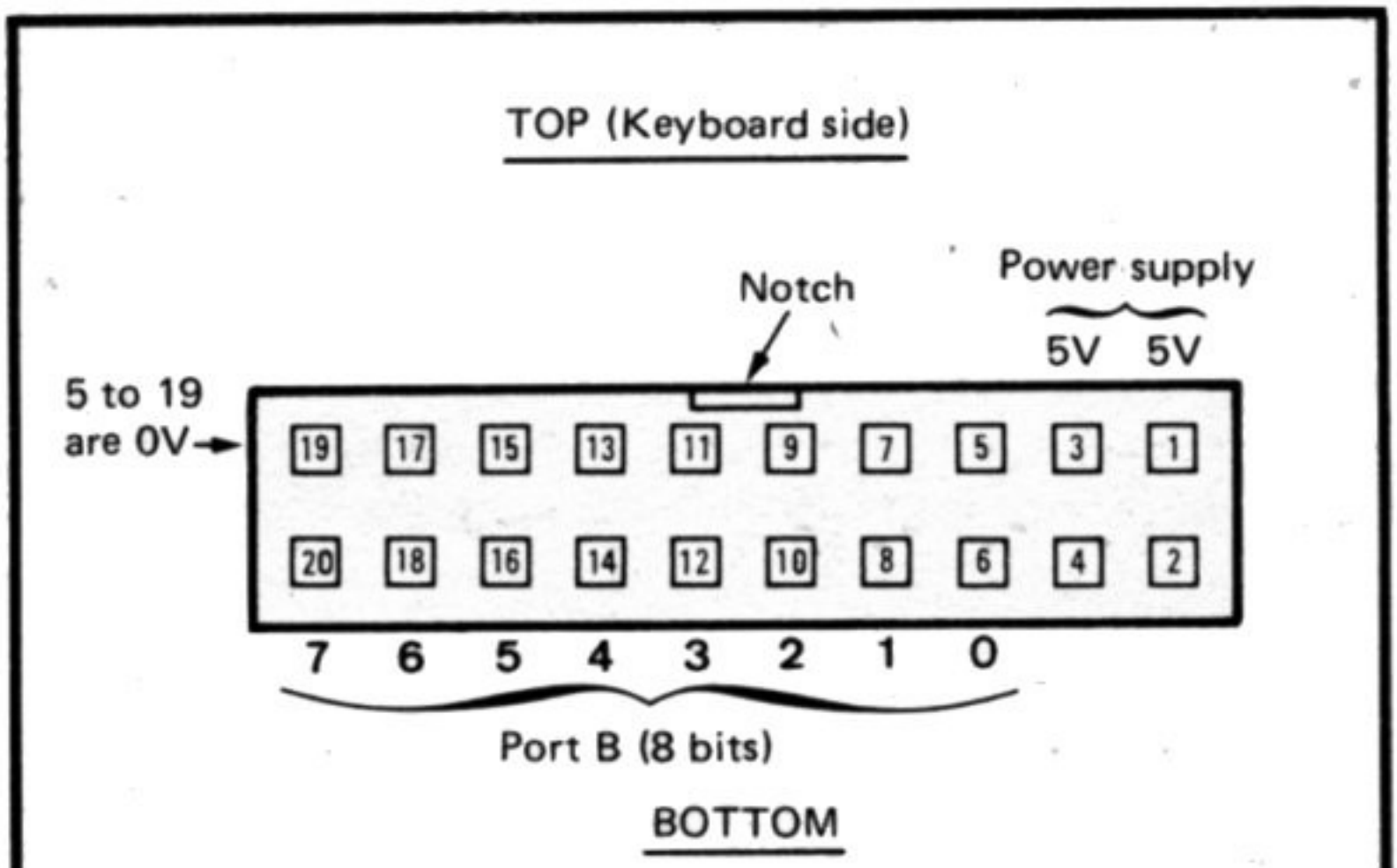


Fig. 0.2 The user port connector of the BBC microcomputer (model B) as seen from the front, when the front edge of the machine is lifted

nected to this terminal, so that the computer and the add-on devices share a common reference line for zero volts. Finally, with the BBC Microcomputer only, we use the 5 V terminal to supply power to the decoder and attached devices.

The 8 lines of port B of the VIA or CIA can act as *outputs*, sending data from the data bus of the computer to the add-on devices. Alternatively, they can act as *inputs*, receiving data from the add-on devices and sending it by way of the data bus to the microprocessor. The lines can be all outputs, or all inputs, or we can have a mixture of inputs and outputs. There is a special register, called the Data Direction Register for port B (DDRB, for short) which determines whether any given line is an input or an output. Incidentally, although we are referring here only to port B, the VIA also has a port A with similar properties. This is taken over permanently for special functions of the computer. In the VIC-20, for example, it handles input from the Joystick (*see* Project 11) and serial port; in the BBC Microcomputer it controls the printer.

When the computer is first switched on, all lines of the ports are set to be inputs. This is to prevent the computer from sending unwanted signals to add-ons before it has been properly programmed. DDRB holds 8 bits, one bit for each line of the port. If a bit is made "0" (as at power-up) the corresponding line is an input. If the bit is made a "1", it is an output. We control which lines are to be outputs and which are to be inputs by sending the appropriate pattern of bits to DDRB.

In this book, we always use lines PB0 to PB3 as input lines, and lines PB4 to PB7 as output lines. Consequently, we need to set DDRB like this:

	← *outputs* →				← *inputs* →			
Line	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Bit value	1	1	1	1	0	0	0	0

Readers familiar with binary numbers will recognise that the binary number 11110000 is equivalent to the number 240 in decimal (Table 0.1). This is the value which must be sent to DDRB. In the VIC-20 and Commodore 64 we do this

by POKEing the value 240 to the address of the DDRB. In the VIC-20, DDRB has the address 37138, so the program line required is:

```
1Ø POKE 37138,24Ø
```

This statement should appear at the beginning of every

Table 0.1 Some binary numbers and their equivalents

Binary	*Hexadecimal*	*Decimal*
0	00	0
1	01	1
10	02	2
11	03	3
100	04	4
101	05	5
110	06	6
111	07	7
1000	08	8
1001	09	9
1010	0A	10
1011	0B	11
1100	0C	12
1101	0D	13
1110	0E	14
1111	0F	15
1 0000	1D	16
1 1001	11	17
1 0010	12	18
10 0000	20	32
11 0000	30	48
100 0000	40	64
1000 0000	80	128
1111 0000	F0	240
1111 1111	FF	255

program which involves the use of the add-on devices described in this book. The only exceptions are the Five-key Pad (Project 3), the Light Pen (Project 7) and the Games Control (Project 11), which are plugged into the VIC-20's or Commodore 64's controller port instead of the user port, as will be explained for each of these projects.

The address of the DDRB in the Commodore 64 is 56579, so the equivalent program line is:

```
10 POKE 56579,240
```

The BBC Microcomputer has DDRB among the SHEILA addresses. The DDRB address is 65122, or &FE62 in hexadecimal. There are two ways of sending data to this address. The simpler way is to use the indirection operator (?), as in this program line in BBC BASIC:

```
10 DDRB = 65122 : ?DDRB = 240
```

This is an easy and satisfactory method, but will not work if you install a second processor. If you have a second processor, or intend to do so eventually, it is better to use an OSBYTE call, as explained in the User Manual. This can be done from a machine code routine but you will probably prefer to make the call from a BASIC program. Put the required values in the integer variables A%, X% and Y% and then use the CALL statement:

```
10 A%=&97: X%=&62: Y%=240: CALL &FFF4
```

In the line above, A% holds the number (in hexadecimal) of the OSBYTE call for *writing* to an address in the SHEILA range, and X% holds the offset (in hexadecimal) of DDRB within that range. Y% holds the value (in decimal) to be put in DDRB, though you could instead express the value in hexadecimal (Y% = &F0).

Decoder

If you have only one device attached to the microcumputer, 8 lines of the user port are usually enough for controlling the device and for receiving data from it. However, we often need

to have several add-ons working simultaneously. For example, you might want to operate the Model Controller, the Bleeper and the Lap Sensor together. The Lap Sensor can function as a position detector suitable for sensing the action of a model and reporting back to the micro. A slot-car racing system or model railway may be operated automatically by using these two devices. You would probably want several sensors in operation on the same model. The Bleeper could act as a remote warning device to let you know if the system needed manual attention. Projects 12 to 17 provide another example where several add-ons may be required to act together. These six devices are all concerned with recording weather conditions, so a complete weather station needs all six in action at once.

For this reason a Decoder circuit has been designed (*see* Appendix A) which allows up to 18 projects to be connected to the computer and to be operated at the same time. This also allows you to have several identical devices (e.g. several thermometers placed in different rooms) all working at the same time, but each being independently controlled by the computer.

The remainder of this Introduction gives an overall view of the way to control your add-ons when they are attached to the Decoder. It deals with the general principles as applied to each of the three machines. Details of how to control a particular add-on are given under each project description, throughout the book. There you will find specific programs for each of the computers. Thus, if you prefer to go straight on to build and use one of the projects in order to get the "feel" of the system, miss out the remainder of this Introduction now, and return to it later. A summary of the main points about programming the Decoder appears at the end of this Section. A description of the electronic aspects of the decoder apppears in Appendix A, together with constructional details.

Add-ons may be connected to the Decoder in three different ways:

(i) Addressed Outputs

Several of the projects need only a single electrical pulse to trigger them into action. An addressed output provides such a pulse. The idea of addressing is that we can have several such outputs (the Decoder provides 11) with a different device connected to each one. When we want to trigger a given device into action, we address the output to which that device is connected. This address is not to be confused with the address of port B in the computer's memory. All signals to addressed outputs go first to the address of port B, so *all* POKEs are made to the same RAM address. We select which add-on is to be triggered by sending (or POKEing) its special "Decoder" address. Table 0.2 shows the eleven Decoder addresses (1–11) available for triggering add-ons. Note that in each case the address is the number of the addressed output multiplied by 16. Since all numbers are multiples of 16, they appear on the upper 4 lines (PB4 to PB7) of Port B, which we always define as output lines, as explained earlier.

For example, if the Bleeper is connected to addressed output no. 3, we trigger it to make its sound by sending the value 48 (00110000) to port B. Sending a value to port B involves the same programming techniques as sending values to DDRB, except that we use the address of port B instead of the address of DDRB. The address of port B is:

In the VIC-20:	37136
In the Commodore 64:	56577
In the BBC Microcomputer:	65120

To trigger the Bleeper when attached to a VIC-20, the statement is:

```
2Ø POKE 37136,48
```

If you have several different add-ons connected to different addressed outputs of the Decoder, you can trigger any or all of them by POKEing the corresponding Decoder address to 37136. In the Commodore 64, you use "20 POKE 56577,48", etc. Remember that these commands will *not* work unless you begin the program by defining the upper 4 lines of port B as output lines, as previously explained.

Table 0.2 Addresses of addressed outputs of Decoder

Address	*Value to be POKEd to port B*	*Function*
0	0	For making all other A/Os high
1	16	
2	32	
3	48	
4	64	
5	80	Available for triggering add-ons
6	96	
7	112	
8	128	
9	144	
10	160	
11	176	
12	192	Enable Picture Digitiser
13	208	Enable Weathercock
14	224	Enable 1-bit data inputs (Group 1)
15	240	Enable 1-bit data inputs (Group 2)

With the BBC Microcomputer you can trigger an addressed output by the equivalent of a POKE:

```
2Ø PORTB = 6512Ø : ?PORTB = 48
```

Alternatively, call the OSBYTE routine, as already explained:

```
2Ø A%=&97: X%=&6Ø: Y%=48: CALL &FFF4
```

X% holds the offset of port B, while Y% holds the Decoder address of the add-on which is to be triggered.

Normally the voltage at an addressed output is high (+5 V). When it is addressed as described above, the voltage

falls low (0 V). It is this fall which triggers the add-on into action. The voltage stays low until another one of the addressed outputs is addressed. With some devices, the fact that the output stays low may not be of any consequence. Other devices such as the Bleeper, for example, will be repeatedly triggered for as long as the output remains low. To prevent this happening, the output must be made high again immediately. This is done by addressing another addressed output. If you do not want to trigger any other device at that moment, use address 0, for the output corresponding to this address does not have any device attached to it. On the VIC-20 the program line for a brief triggering pulse at addressed output 3 is:

```
3Ø POKE 37136,48:POKE 37136,Ø
```

The corresponding pair of statements may be used with the other computers.

(ii) One-bit Data Inputs:

Several of the projects send only one bit of data to the computer at any one time. The output from the add-on is either "0" or "1". We define port B with 4 input lines (PB0 to PB3), and can attach an add-on to each of these lines. The Decoder provides this facility, and also allows a second group of 4 add-ons to send data along the same 4 lines. Obviously, we cannot allow more than one device to put data on to the same data line at the same time, so there must be some way of letting only one add-on communicate with a given data line at any time. The system used is shown in Fig. 0.3. Each add-on is connected to one of the input lines of port B by a three-state buffer. When such a buffer is enabled (by placing a low voltage level (0 V) on its "enable" input) its output follows its input. That is to say, if the input from the add-on is "0", the output to the line is "0"; if the input is "1", the output is "1". When its "enable" input is made high (+5 V), the output of the buffer goes to its third state, which we call "high impedance". In this state, the buffer is, in effect, disconnected from the line. It passes no data to the line and might just as well not be there. We arrange it so that only one

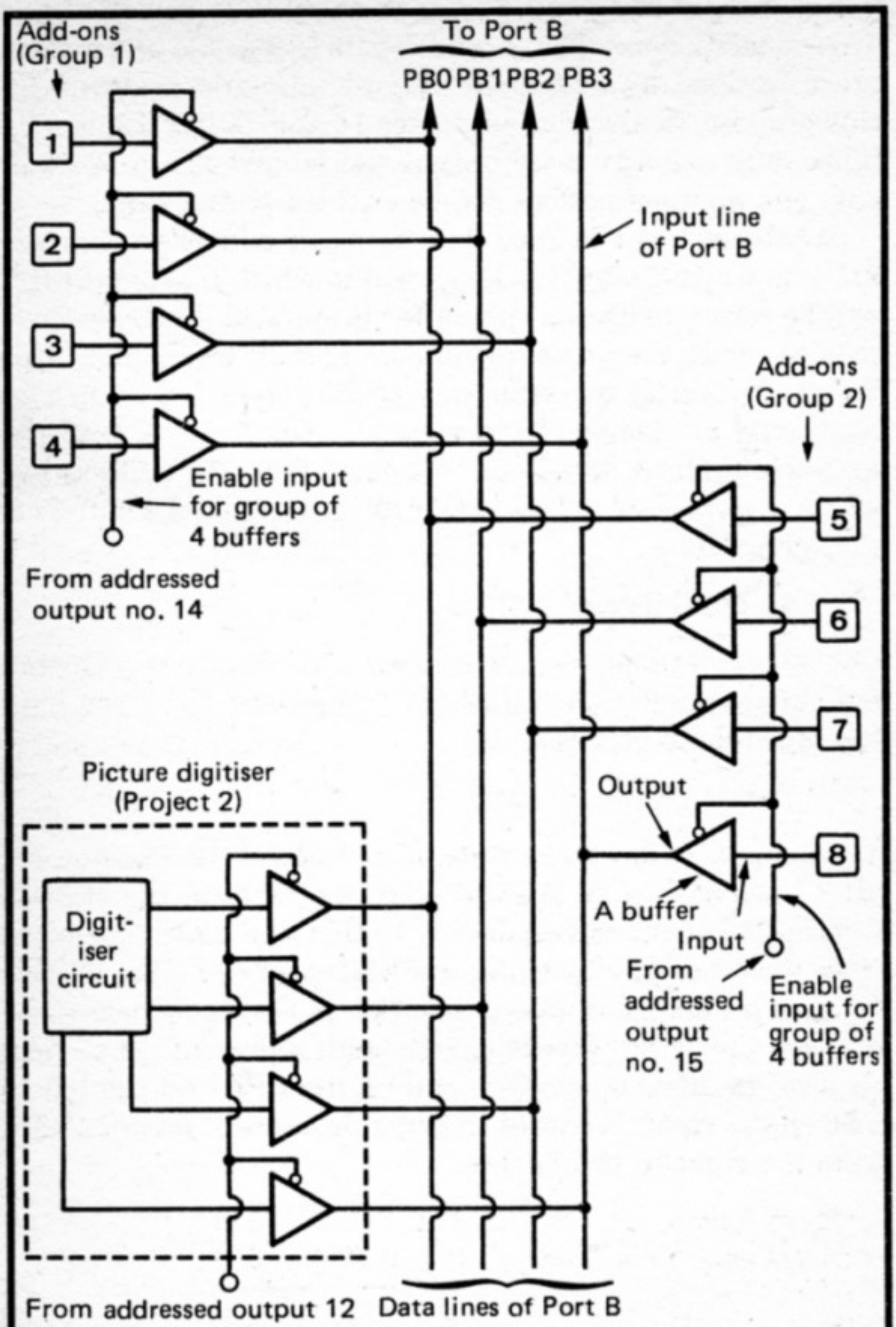


Fig. 0.3 How up to 8 add-ons with one-bit outputs may be connected to the input lines of Port B. The Figure also shows the Picture Digitiser with 4-bit output from its own set of buffers

group of three-state buffers can be enabled at any one time. This is easily done. The enable inputs of a group of 4 buffers are controlled by addressed outputs specially reserved for this purpose, at Decoder addresses 14 and 15 (*see* Table 0.2). Since only one addressed output can be low at any instant, only one group of buffers can be enabled at any time.

When we want to read data from an add-on we have to know to which group it belongs and to which line of port B it will be connected when the buffer is enabled. Before we can read the data, the correct set of buffers must be enabled. This is done in exactly the same way as that described earlier for triggering an add-on. If for example, you want to read the data coming from an add-on of group 1 (Decoder address 14), which is connected to line PB2, you first enable the buffers of that group by:

```
4Ø POKE 37136, 224
```

The statement above applies to the VIC-20; use corresponding statements for the other two computers. Next you read the registers of port B:

```
5Ø R = PEEK (37136)
```

The value of R shows the state of *all 8 bits* of the register, i.e. all 8 lines of Port B. We are interested only in the state of line PB2. The bit corresponding to this line may be selected from the other bits by using the AND operator. We call this "masking", and here is how it works. In the Table below, the digits of the value of R (245 in decimal) represent the state of each of the lines of port B, running from PB7 on the left to PB0 on the right. We want to know the state of line PB2 (3rd from the right in the Table):

R (in binary)	1 1 1 1 0 1 1 0
AND with 4 (in binary)	0 0 0 0 0 1 0 0
Result	0 0 0 0 0 1 0 0

Each digit of R is ANDed with the corresponding digit of "4" in binary. Only if *both* digits of a pair are "1" is the result a "1". The value of the result is "4", in decimal, showing that

PB2 is "high". Had line PB2 been at a low level, and assuming that the other lines were as before, the value, of R would have been:

R (in binary)	1 1 1 1 0 0 1 0
AND with 4	0 0 0 0 0 1 0 0
Result	0 0 0 0 0 0 0 0

In this case, the result is zero.

We can do the reading and ANDing in a single operation by using the program line:

```
5Ø R = PEEK(37136)AND4
```

If PB2 carries a "1", R equals 4; while if PB2 carries a "0", R equals 0.

By using this technique we can read the state of any bit, and the result is not affected by the state of any of the other bits. Table 0.3 shows which value to AND with the PEEKed value to find the state of any given input line. In each case, the result of ANDing is a zero if the line carries a "0". If the line carries a "1", the result of ANDing equals the value ANDed.

The technique used above applies also to the Commodore 64, except that the address to be PEEKed is 56577. On the BBC Microcomputer we can use the indirection operator and the AND operator:

```
5Ø PORTB = 6512Ø: R = ?PORTB AND 4
```

It is essential to leave the spaces on either side of "AND". If you want to make your program suitable for use with a second processor, you will have to use the OSBYTE call &96 for reading the state of port B. Since we need to obtain a value from this routine, the USR function is required instead of CALL. The program line is:

```
5Ø A%=&96:X%=&6Ø:R=( USR(&FFF4)AND&4ØØØØ)
   /&1ØØØØ
```

With any of the computers the appropriate addressed output is made low in order to enable one or other of the groups

Table 0.3 Selecting port B lines, by ANDing

Line	*AND with*
PB0	1
PB1	2
PB2	4
PB3	8

of buffers. After that, the data from any one or more of the add-ons in the enabled group may be read as often as desired, for their outputs are continuously being fed to the lines of Port B. However, if it is required to trigger some other add-on, a new value must be POKEd to DDRB. This disables both groups of buffers and the flow of data from the add-ons is interrupted until one or other of the groups of buffers is enabled again.

(iii) 4-bit Data Inputs:

Two of the projects, the Picture Digitiser and the Weathercock, need to send to the micro more than one bit at a time. Each of these projects incorporates its own set of three-state buffers. The outputs of these buffers are connected directly to the lines of port B (Fig. 0.3). Each project requires an addressed output to enable its buffer IC. The programs given under these projects assume that addressed outputs 12 and 13 are used for this purpose. Reading the data from these projects is just the same as reading the data from single-bit inputs, except that a different value has to be ANDed with the value obtained by PEEKing. The values to be used and the way the results are to be interpreted are explained in the project description.

Programming Summaries

(Details of how to read 4-bit inputs are given under the projects concerned.)

VIC-20

(1) Initialise DDRB at the beginning of every program:

```
1Ø POKE 37138,24Ø
```

(2) Trigger add-ons attached to addressed outputs of the Decoder by:

```
2Ø POKE 37136,48
```

The number 48 refers to address 3 (Table 0.2). The Table shows which number to POKE for other addresses. To terminate the triggering action:

```
3Ø POKE 37136,Ø
```

or trigger some other addressed output.

(3) Read one-bit data inputs in group 1 by:

```
4Ø POKE 37136,224
5Ø R = PEEK(37136)AND4
```

The number 224 refers to group 1; for a buffer in group 2, use the value 240. The number 4 refers to the buffer wired to PB2 (Table 0.3). The Table shows which number to AND for other buffers in the group.

Commodore 64

(1) Initialise DDRB at the beginning of every program:

```
1Ø POKE 56579,24Ø
```

(2) Trigger add-ons attached to addressed outputs of the Decoder by:

```
2Ø POKE 56577,48
```

The number 48 refers to address 3 (Table 0.2). The Table shows which number to POKE for other addresses. To terminate the triggering action:

```
3Ø POKE 56577,Ø
```

or trigger some other addressed output.

(3) Read one-bit data inputs in group 1 by:

```
4Ø POKE 56577,224
5Ø R = PEEK(56577)AND4
```

The number 224 refers to group 1; for a buffer in group 2, use the value 240. The number 4 refers to the buffer wired to PB2 (Table 0.3). The Table shows which number to AND for other buffers in the group.

BBC Microcomputer (direct memory addressing)

(1) Initialise DDRB at the beginning of every program:

```
1Ø DDRB=65122:?DDRB=24Ø
```

(2) Trigger add-ons attached to addressed outputs of the Decoder by:

```
2Ø PORTB=6512Ø:?PORTB=48
```

The number 48 refers to address 3 (Table 0.2). The Table shows which number to use for other addresses. To terminate the triggering action:

```
3Ø ?PORTB=Ø
```

or trigger some other addressed output.

(3) Read one-bit data inputs in group 1 by:

```
4Ø PORTB=6512Ø:?PORTB=224
5Ø R = ?PORTB AND4
```

The number 224 refers to group 1; for a buffer in group 2, use the value 240. The number 4 refers to the buffer wired to PB2 (Table 0.3). The Table shows which number to AND for other buffers in the group.

BBC Microcomputer (using OSBYTE routines)

(1) Initialise DDRB at the beginning of every program:

```
1Ø A%=&97:X%=&62:Y%=24Ø:CALL&FFF4
```

(2) Trigger add-ons attached to addressed outputs of the Decoder by:

```
2Ø A%=&97:X%=&6Ø:Y%=48:CALL&FFF4
```

The number 48 refers to address 3 (Table 0.2). The Table shows which number to use for other addresses. To terminate the triggering action:

```
3Ø Y%=Ø:CALL&FFF4
```

or trigger some other addressed output.

(3) Read one-bit data inputs in group 1 by:

```
4Ø A%=&97:X%=&6Ø:Y%=224:CALL&FFF4
5Ø A%=&96:R=(USR(&FFF4)AND&4ØØØØ)/&1ØØØØ
```

The number 224 refers to group 1; for a buffer in group 2, use the value 240. The figure "4" in the number &40000 refers to the buffer wired to PB2 (Table 0.3). The Table shows which number to substitute for the figure "4" in "&40000" to read other buffers in the group.

(4) Procedures. If, as is likely, it is required to trigger or read add-ons many times during a program, it is preferable to adapt the routines above and use them in procedures. A procedure for triggering an addressed output is:

```
5ØØØ DEF PROCtrigger(addon)
5Ø1Ø A%=&97:X%=&6Ø:Y%=addon*16
5Ø2Ø CALL &FFF4
5Ø3Ø Y%=Ø
5Ø4Ø CALL &FFF4
5Ø5Ø ENDPROC
```

This procedure puts a brief low pulse on the addressed output line, then restores the level to high again. Call this procedure, using the number of the addressed output as the parameter. For example, to trigger an add-on connected to addressed output 3, the program line is:

```
1ØØ PROCtrigger(3)
```

A procedure to read a one-bit data input is:

```
6000 DEF PROCread(group,line)
6010 A%=&97:X%=&60:Y%=208+group*16
6020 CALL &FFF4
6030 A%=&96
6040 R=(USR(&FFF4)AND(2↑line*&10000))
     /&10000
6050 A%=&97:Y%=0
6060 CALL &FFF4
6070 ENDPROC
```

This procedure returns the result of ANDing in variable R. It ends by disabling the buffers. Call this procedure using the group number and line number (of port B) as parameters. For example, to read the output from a device in group 2 attached to the buffer wired to line PB3, use this program line:

```
300 PROCread(2,3)
```

Acorn Electron

See pages 185–7.

PROJECT 1

Pulse Detector

This simple device is a very useful one for testing the other interfaces. The trouble with micros is that they work very fast. The micro produces pulses of current which may last for only a fraction of a microsecond. This is far too short a time to affect an ordinary voltmeter. Even with an oscilloscope it is almost impossible to pick out the pulse you are interested in. This circuit is designed to detect a low pulse produced by a microcomputer. The words "low pulse" mean that the voltage at the point of testing is normally "high" (anything between +2 V and +5 V), but goes "low" (falls below +0.8 V) for a short period. This circuit is quick enough to detect low pulses lasting as little as a fiftieth of a microsecond, which is fast enough for almost all pulses that a microcomputer can produce.

How It Works

The detecting section of the circuit consists of two logic gates, wired as in Fig. 1.1 to make a flip-flop. The output from this section (at pin 3) may be "high" or "low", but must be one or the other. Remember that anything between +2 V and +5 V counts as "high", while anything between 0 V and +0.8 V counts as "low". Voltages can not be higher than +5 V because that is the voltage at which we supply power to the circuit.

With the flip-flop in its "reset" state, waiting for a pulse to arrive, the output at pin 3 is "low". Both of its inputs, the one labelled $\overline{\text{INPUT}}$ and the one labelled $\overline{\text{RESET}}$ are high. $\overline{\text{INPUT}}$ is high because it is connected to the point at which a low pulse is expected at any moment. Although $\overline{\text{RESET}}$ is not connected to anything (for the button S1 is not being pressed), it *acts* as if it is high. This is a property of the TTL integrated circuits used in most of the projects in this book.

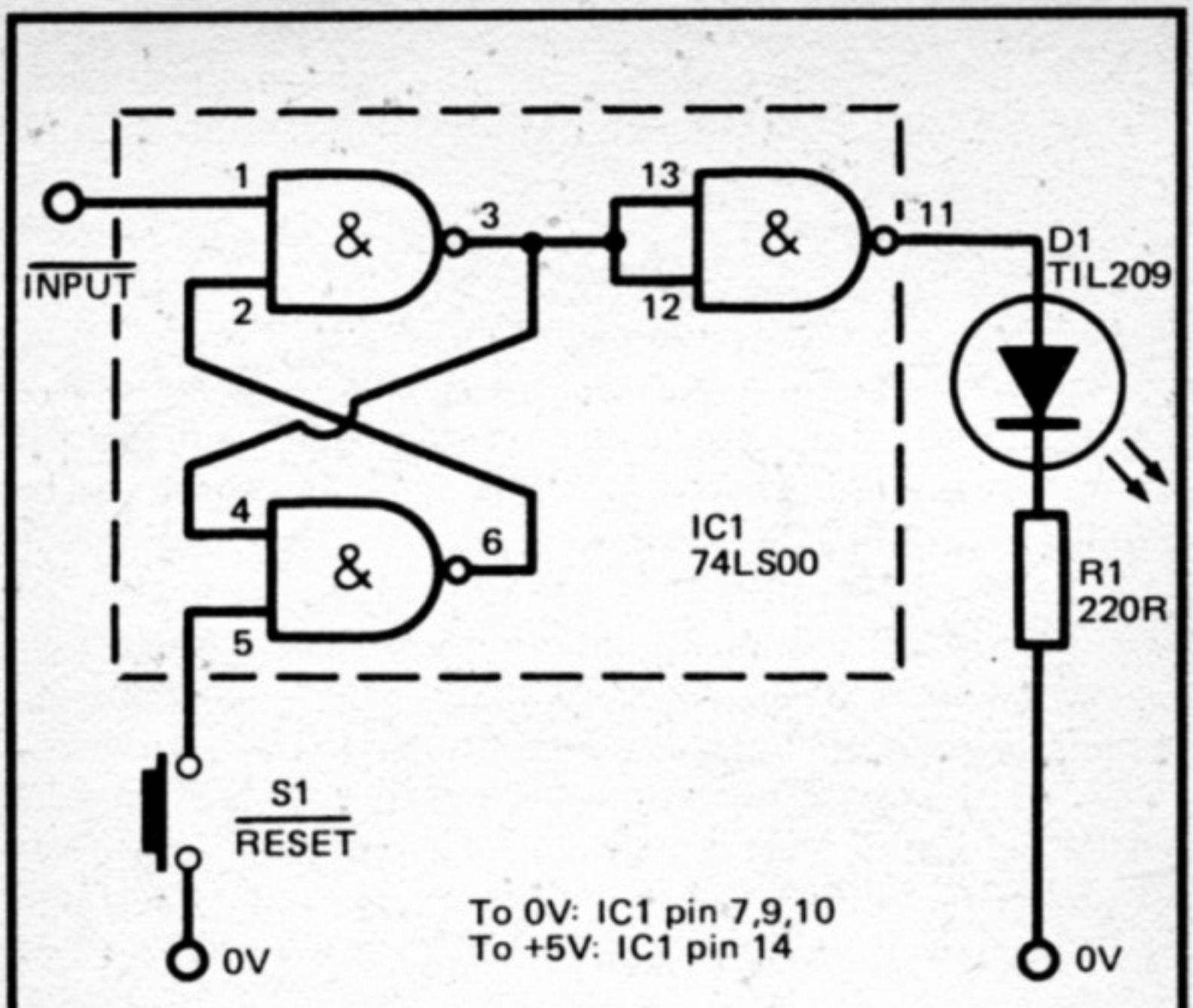


Fig. 1.1 The circuit diagram of the pulse detector

Any input which is unconnected acts as if it was receiving a high-voltage level.

The "low" output from pin 3 goes to another logic gate, which "inverts" it. That is to say, the output of this gate (at pin 11) is "high". The light-emitting diode, D1 is lit. You may wonder why we do not drive the LED directly from pin 3. The reason is that, if we do this, the flip-flop does not respond as quickly and might miss the short low-pulse it is waiting for.

When the low pulse arrives, the flip-flop changes state. It becomes "set". The output at pin 3 goes "high", which makes the output at pin 11 go "low", and the LED goes out. To reset the flip-flop and put the LED on again, we press the "reset" button for an instant. Provided that the pulse is finished and the voltage at $\overline{\text{INPUT}}$ has returned to a "high" level, pressing S1 makes the flip-flop return to its "reset" state.

Building It

This project can be built up on any small scrap of stripboard. There is hardly any need to put it in a case but it is easy to find a small plastic container which will suit the purpose of enclosing it. The push button and LED are mounted on the case.

There is no need for a power-supply switch. The best arrangement is to have three wires about 20 cm long to connect the Detector to the circuit (or the line of the micro) which is being tested. Solder a crocodile clip to the flying end of each wire. Preferably use clips with insulating plastic covers, and have clips of three different colours so that you know which is which. One of the wires is for connecting the Detector to the 0 V line of the micro, or test circuit. This wire connects also to the IC and to the other points shown on Fig. 1.1. The second wire is for taking power to from the +5 V rail of the micro, or tested circuit. This wire connects to pin 14 of the IC. The third wire is the $\overline{\text{INPUT}}$ wire going to pin 1.

Using It

Switch off the micro or the circuit you are testing. Clip the 0 V and +5 V leads to the 0 V and +5 V rails of the micro or other circuit. Clip the $\overline{\text{INPUT}}$ wire to the line you want to test for low pulses. Then switch on the power supply.

The LED may or may not light to begin with, depending on which state the flip-flop goes to when switched on. If the LED does not light, press S1 for an instant. If the LED does not light then, it may be that the tested line is not actually "high", or that a series of "low" pulses are coming along it so often that the flip-flop becomes "set" almost immediately after you reset it. If this is the case, there is nothing more you can do, for the Detector is not designed to work with a rapid series of "low" pulses.

If the LED comes on and stays on when you press S1, operate the micro or test circuit in the way which you believe is going to produce that single "low" pulse. If or when the pulse arrives, the LED goes out.

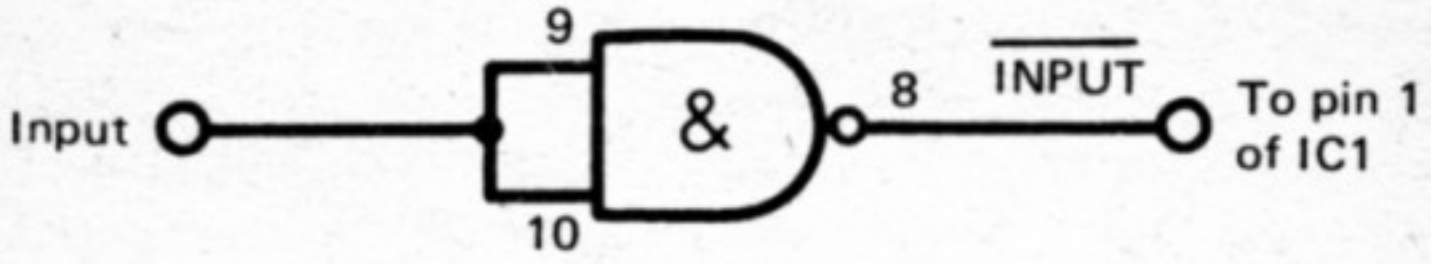


Fig. 1.2 Adapting the Pulse Detector to detect high pulses

Parts Required for the Pulse Detector

Resistor

R1 220 R carbon, 0.25 W, 5% tolerance

Semiconductor

D1 TIL209 or similar light-emitting diode

Integrated Circuit

IC1 74LS00 quadruple 2-input NAND gate

Miscellaneous

S1 Push-to-make push-button switch

Scrap of stripboard (2.5 mm matrix)

1 mm terminal pins (5 off)

Crocodile clips, miniature size (3 off, of different colours) (Better still are miniature clip-on probes, such as "E-Z Hooks")

14-pin IC socket (optional, but recommended)

Small plastic case (optional)

Connecting wire

Detecting a "High" Pulse

Fig. 1.2 shows how the Detector may be adapted to detect a "high" pulse on a line that is normally "low". As it happens, you are more likely to be looking for "low" pulses in most of the projects in this book, but occasionally the important pulse is a "high" one. All that the gate of Fig. 1.2 does is to invert the pulse before sending it on to the flip-flop. The line to which $\overline{\text{INPUT}}$ is connected must be one that is normally "low". The output from pin 8 will therefore normally be "high". When a "high" pulse arrives at $\overline{\text{INPUT}}$, pin 8 will give out a "low" pulse and so trigger the flip-flop.

The gate used for the adapted version is the spare gate of IC1. Instead of joining its inputs to the 0 V line, they are joined to each other and used to receive the $\overline{\text{INPUT}}$ signal. You could wire up your Detector so that it is possible to make temporary connections by means of crocodile clips whenever you want to look for "high" pulses instead of "low" ones.

PROJECT 2

Picture Digitiser

Using this device, you can scan a photograph or drawing and see it appear on the screen of your micro. The scanner is rather a low-resolution one, so it works best when the picture has large bold areas of contrasting shades. Fine detail will be lost but, provided that you choose a suitable subject, it is fascinating to see the result building up block by block on the screen. The scanner is moved by hand, so it works fairly slowly and depends on you to move it accurately. It clearly demonstrates the principle behind professionally-made (and highly expensive) picture-digitising equipment. There is a great amount of fun to be had from the simple Digitiser.

How It Works

The idea behind this device is that the picture is broken down into a number of picture elements, or pixels. The picture area consists of rows and columns of pixels, as in Fig. 2.1. The scanner moves (or, in this instance, is moved) along each row of pixels in turn. It measures the brightness of each pixel, and reports this to the micro. Brightness is something which could be anything between a brilliant white and the darkest of blacks. There is an unlimited number of shades of grey in between. If the picture is a coloured one, the different colours can be considered to be equivalent to different shades of grey.

The micro cannot accept an unlimited number of different possible shades. To keep the circuit simple, this scanner is designed to recognise only 4 different shades. The lightest is white, the darkest is black and there are two shades of grey (light-grey and dark-grey or equivalent colours) in between. When the scanner measures the overall brightness of a pixel it tells the micro to which of these 4 possible shades the pixel

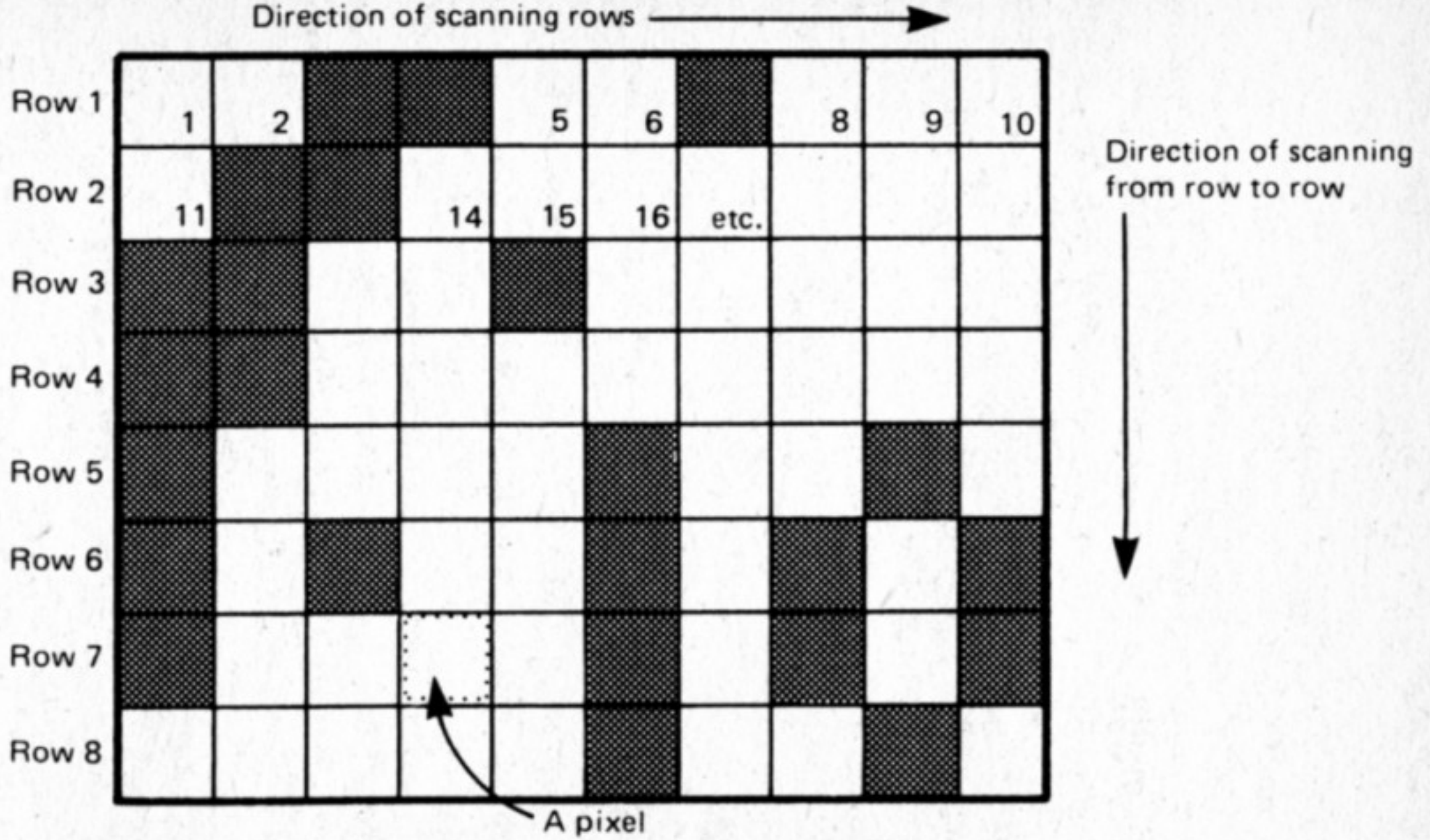


Fig. 2.1 Part of the face of a clock (near 10 o'clock) drawn as a set of pixels. In this example the pixels are either black or white, but pixels can be in shades of grey or in colour. The numbers show the order in which the pixels are measured by a scanner and then plotted on a screen

shade is closest. The micro plots each pixel on the screen as it is reported. It plots small squares which show a corresponding range of brightness. White is a solid block of light on the screen, and black is a black area of screen. The greys are obtained in various ways, depending on the computer.

Owners of the Commodore 64 are fortunate in that this machine already has true grey colours in its repertoire. You can use any two of its three shades of grey, as you prefer. The VIC-20 can display a medium-grey block by using the left-side character on the "+" key. For a second grey on the VIC-20 and for both greys on the BBC Microcomputer, we employ user-defined graphics characters.

The circuit (Fig. 2.2) relies on a photodiode (D1) to detect the light reflected from the picture. The light comes from a torch bulb, which is mounted with the photodiode in the scanner (Fig. 2.3). If light is falling on D1, a small leakage current flows through R1 and D1. The more light reaching D1, the larger the current. The larger the current, the greater the potential difference across R1. The greater the PD, the lower the voltage at point A. This is because the voltage at the top end of R1 is fixed at +5 V, so any increase in PD across it must make the voltage fall at its other end.

The level of voltage at point A is measured by three comparators (IC1, IC2 and IC3). These each compare the voltage at their *inverting input* (−) with the voltage at their *non-inverting input* (+). In this circuit, the inverting inputs are all connected to point A, while the non-inverting inputs are each connected to the wiper of a variable resistor (RV1, RV2 and RV3). Each resistor is set so that the voltage at its wiper has a fixed value between 0 V and +5 V. The setting is different for each variable resistor. For each comparator, if the voltage at the inverting input (from A) is higher than the voltage at the non-inverting input (from the variable resistor), the output of the comparator rises sharply towards +5 V. If the voltage from A is lower than the voltage from the variable resistor, the output falls sharply towards 0 V.

The variable resistors are set so that when D1 is receiving light reflected from white paper, the voltage at A is lower than the voltage from any of the variable resistors. The out-

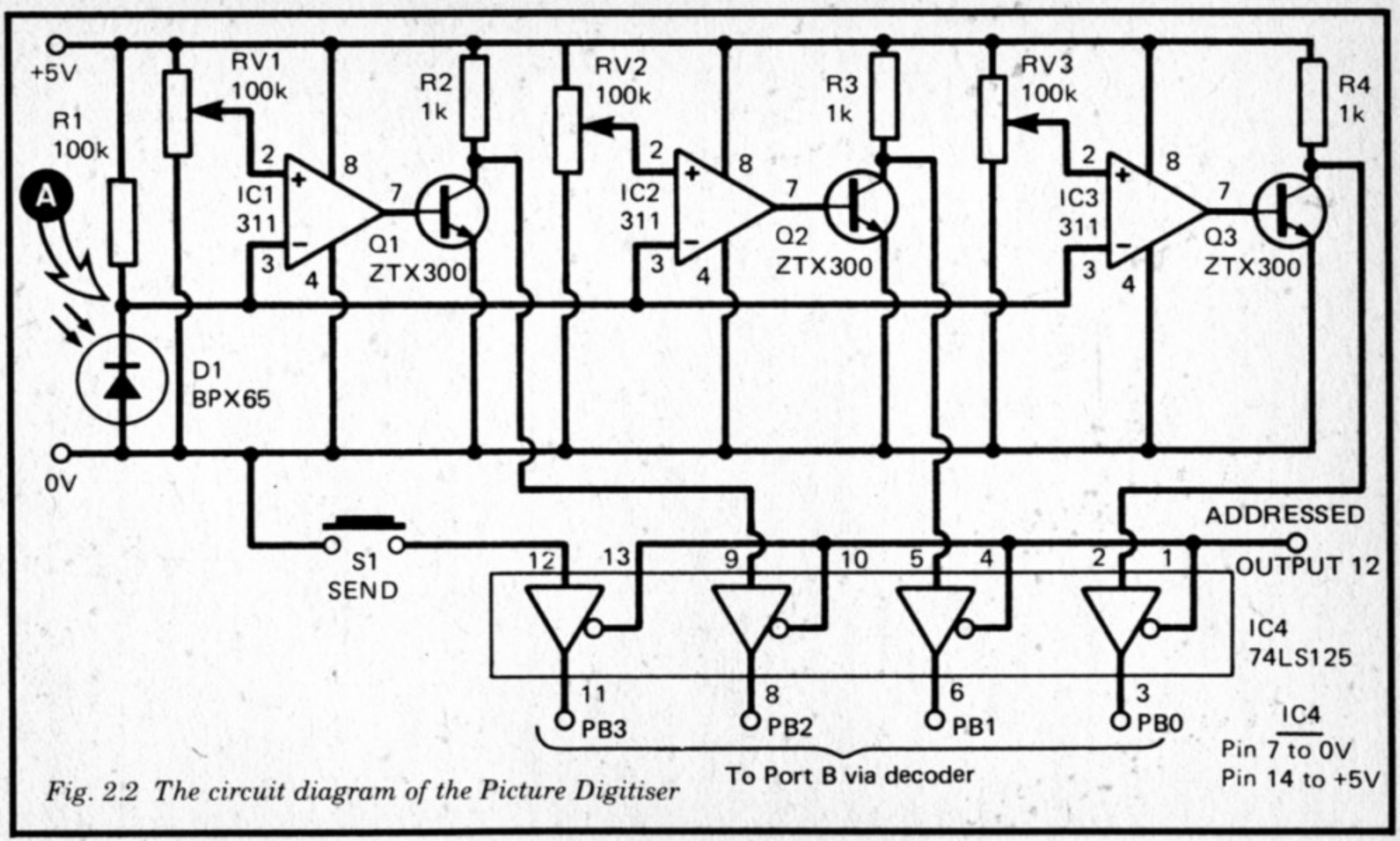


Fig. 2.2 The circuit diagram of the Picture Digitiser

puts of all three comparators are "low". If very little light is reaching D1 (reflection from black paper), the voltage from A is greater than any voltage from the resistors, so all outputs are "high". In between (shades of grey, or colours), one or more of the comparators has a "high" output and the others have a "low" output.

The output of each comparator goes to a transistor. When the comparator has a high output, the transistor is switched fully on. The voltage at its collector falls to 0 V (= "low"). The transistor *inverts* the output from the comparator. This voltage is fed to a buffer gate in IC4. When the buffer is enabled (*see* next section) and "low" level is sent to the buffer, a "low" level is put on to the input line of port B. Conversely, when the comparator has a "low" output, a "high" level appears on the line.

The various possible outputs of the circuit are listed in Table 2.1. The top section of the table shows what happens when the "Send" button (S1, Fig. 2.2) is *not* pressed. The output from its buffer is "high" (*see* p. 19), so the micro reads the four data lines as a number greater than 7. This tells the micro that, whatever data may be appearing on the other lines, the scanner is not actually ready in position over a pixel, so it should ignore these other lines. When the "Send"

Table 2.1 Data output from the Picture Digitiser

"Send" button	*Shade of Pixel*	*Port B* 3 2 1 0	*Decimal equivalent*
Not pressed	Any	1 X X X	More than 7
Pressed	White	0 1 1 1	7
	Light grey	0 0 1 1	3
	Dark grey	0 0 0 1	1
	Black	0 0 0 0	0

0 = "low" 1 = "high" X = "low" or "high"

button is pressed, its buffer output goes "low". Now, the micro interprets the other lines as levels of brightness, depending on how it has been programmed (*see* later).

The enable input of IC4 (pin 1) is connected to one of the addressed outputs on the Decoder (Appendix A). When this output goes low, all 4 buffers are enabled and the data from the Digitiser appears on the input lines (PB0 to PB3) of port B. The data may then be read from port B. Although the buffer could in fact be connected to any one of the addressed outputs (*see* Table 0.2), the program examples given later assume that it is connected to output 12. If you have reasons for changing it to another output, alter the program accordingly.

Building It

The first thing to do is to make the scanner (Fig. 2.3). There are several ways in which you can construct this, depending on the materials you have available and your skill at working with these materials. You may decide to base it on a small metal or plastic box, boring holes and fixing partitions to arrive at the design shown in the Figure. Or you may decide to begin with a block of wood (or a large cork stopper, which is easier to carve) and drill out the various channels required. The essential points of the scanner are:

(i) It rests flat on the picture.
(ii) It has an aperture about 5 mm square on the lower side; this defines the area of the pixel.
(iii) A small lamp shines light on to the picture through this aperture.
(iv) It must be possible to raise or lower the lamp so as to adjust the amount of light reaching the picture. The lamp should be a firm fit in its mounting, so that it does not readily slide out of its fixed position.
(v) The photodiode is to be fixed in position and aimed so that it catches reflected light from the picture, but does not receive light coming directly from the lamp.
(vi) It is best if all surfaces in the scanner are painted matt-black to cut out unwanted reflections of light.

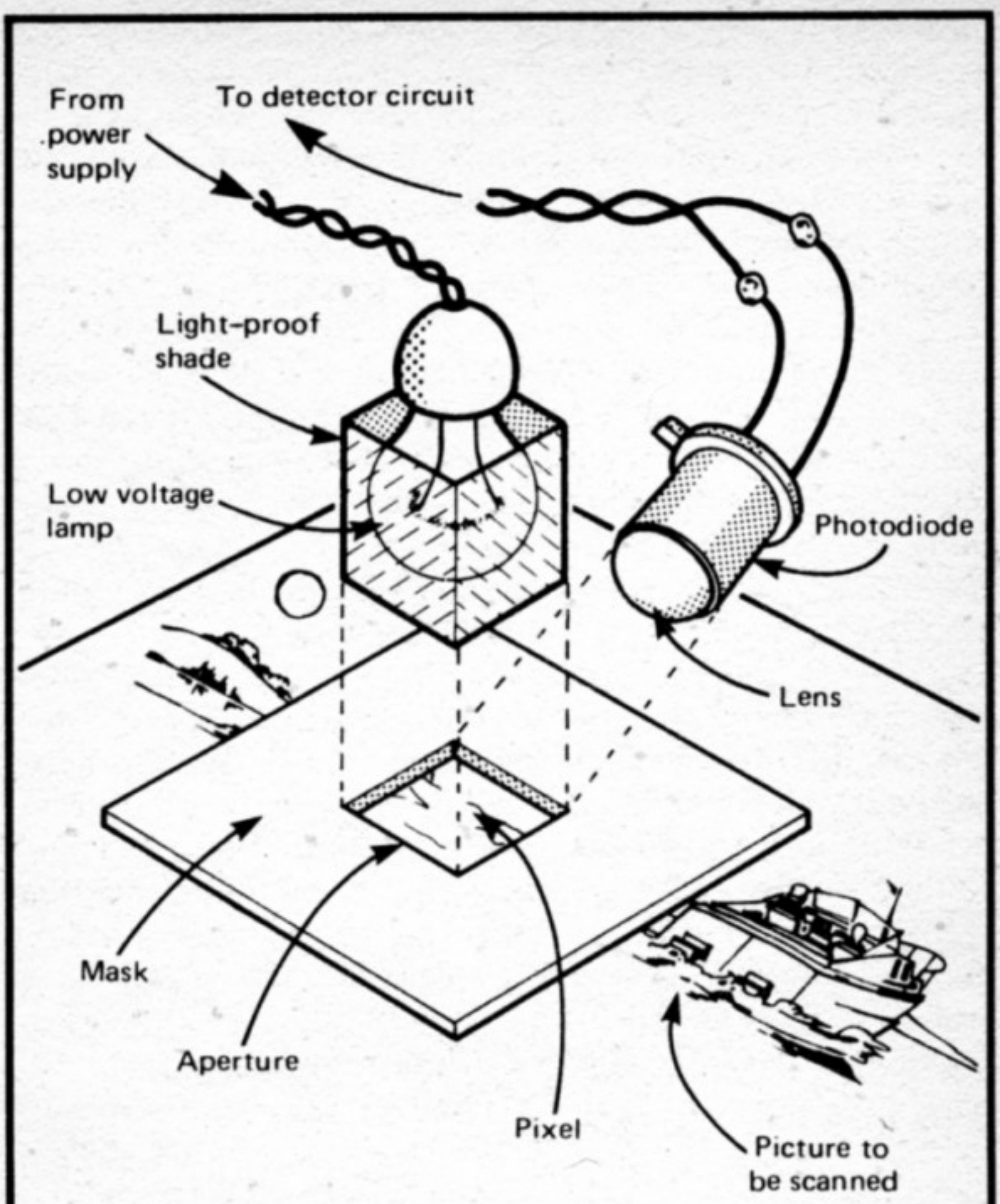


Fig. 2.3 The essential parts of a scanner, but not showing how they are put together

The prototype scanner was made from a large cork of the type used in the demijohns in home wine-making. Such a cork is cheaply obtainable from a shop selling home wine-making equipment. Buy one which already has a central hole. Shape the hole at the lower end to make it square. The small filament lamp (on the end of flexible leads) fits tightly

in the upper end of the hole and can be slid up and down. A slanting hole is cut from one side of the cork, sloping down towards the aperture. After soldering it to flexible leads, the photodiode is wrapped round with insulating tape (with tape between its wires too, to prevent short-circuiting) and wedged into the hole. Before putting the lamp and photodiode into their final positions, the insides of all holes are blackened using a felt-tip marker pen.

Black insulating tape is wrapped around the cork, except on the lower surface, to keep light from entering from the outside. Marks are made on the scanner to assist in registering its position during scanning (*see* later).

You will need to provide terminal pins on the board for the leads to the photodiode and to the Decoder (7 leads: 0 V, +5 V, addressed output 12, and four data lines). You will also need to provide pins for the power supply to the lamp.

Wire up R1 and D1 first, and one of the comparators (say, IC1 with RV1). Test the circuit with a voltmeter connected to point A. In fairly low room lighting the voltage at A is close to +5 V but, when D1 is moved towards the lamp (switched on), the voltage drops almost to 0 V. If you fail to obtain a changing voltage, it is possible that you have connected D1 the wrong way round. Set RV1 to about the middle of its track; the voltage at its wiper and at pin 2 of IC1 is then about 2.5 V. Move D1 towards the lamp; the output of IC1 (pin 7) is close to +5 V at first but suddenly swings to 0 V when D1 is put closer to the lamp. Now wire up R2 and Q1. Check that the voltage at the collector of Q1 (i.e. where Q1 is linked to R2) changes from 0 V to +5 V as D1 is moved towards a lamp.

Repeat the above sequence of tests on the other two comparators as you assemble these sections of the circuit.

Next, wire up the buffer (IC4). Temporarily connect the addressed input line to the 0 V line to enable the buffers. You can then check that the outputs change as expected when D1 is moved towards the lamp (*see* Table 2.1). Adjust the settings of RV1, RV2 and RV3 so that the outputs change in order as D1 is moved towards and away from the lamp.

The device needs *two* sockets to connect it with the

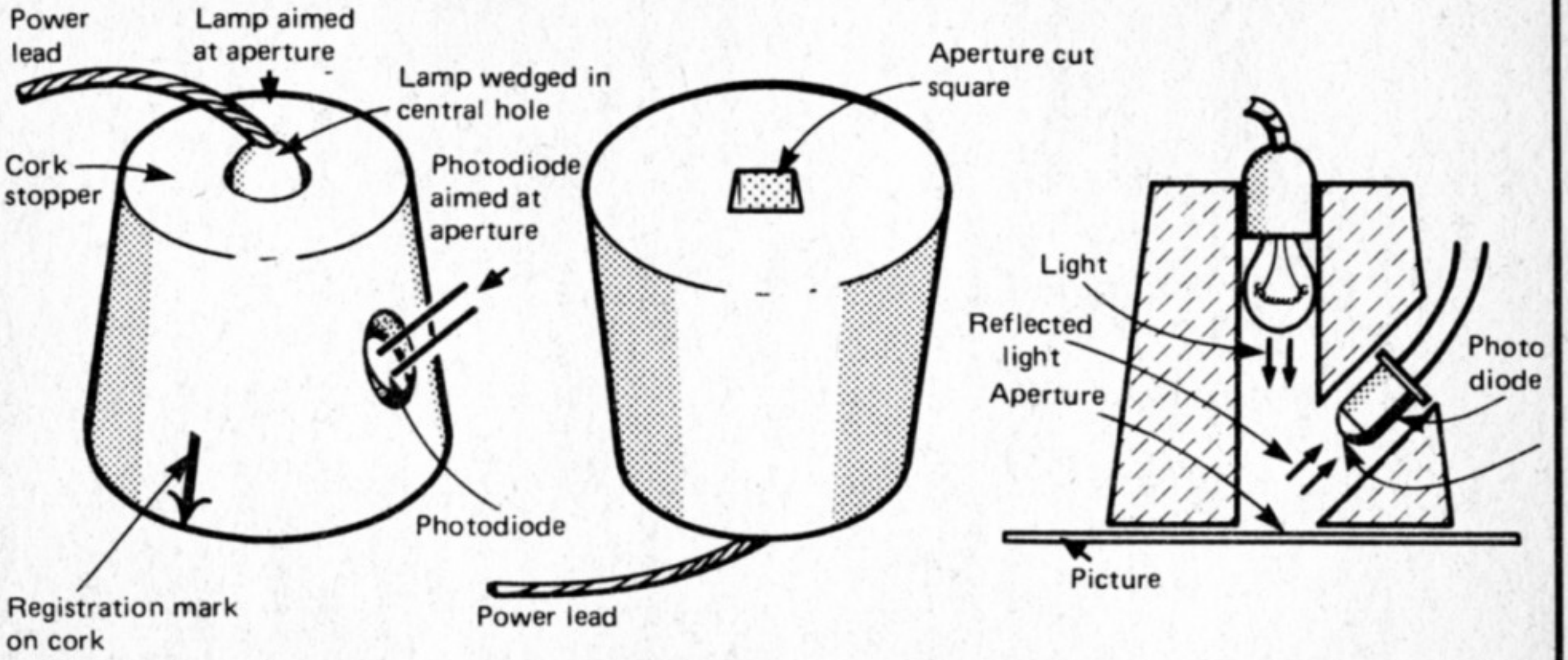


Fig. 2.4 An easy way of making a scanner from a cork stopper; (a) in use, except that connections to the photodiode are not shown; (b) upside down, to show aperture; (c) section

Decoder. One is a 6-way socket to fit one of the two 6-way sockets on the Decoder board. The lines to this socket are: 0 V, +5 V, PB0, PB1, PB2 and PB3, in that order. The other socket is a 3-way socket but with only one connector. This should be at one end of the plug (i.e., not the central connector) and is normally to be plugged into addressed output no. 12.

Final Check

Before connecting the circuit to the Decoder, it is important to test it thoroughly. Use a multimeter or a circuit tester to check that there are no short circuits between next-door data lines, or between data lines and the power lines. It is easy for a thin thread of solder to form a bridge between such lines. Also check that there is no short circuit between the +5 V power line and the 0 V line. It is also worth testing each data line and other control line to make sure that there really is a connection between the plug at one end of the line and the IC or other component which is at the other. A "dry joint" which causes a break in a line can cause all kinds of problems in getting the circuit to work.

If it passes the tests above, the Digitiser should now be ready to plug into the Decoder, which itself is plugged into the micro. Switch on the power supply to the micro. If the display on the screen is not as expected, *switch off immediately* and repeat the checks.

Test Program

Below is a short program for the VIC-20 which tests that the Digitiser is working properly:

```
10 POKE 37138,240:REM DEFINE DDRB
20 POKE 37136,192:REM ENABLE DIGITISER
30 FOR J=1 TO 80
40 PRINT PEEK(37136)AND15;
50 NEXT
60 GET A$:IF A$="" THEN 60
70 GOTO 30
```

The value PEEKed from port B (line 40) is ANDed with 15 (binary 1111) so that the state of all four input lines is detected. This program is adapted to the Commodore 64 by substituting address 56579 for address 37138, and substituting address 56577 for 37136. The equivalent program for the BBC Microcomputer is:

```
10 DDRB=65122:?DDRB=240
20 PORTB=65120:?PORTB=192
30 FOR J=1 TO 80
40 PRINT ?PORTB AND 15;
50 NEXT
60 A$=GET$
70 GOTO 30
```

The program reads the output from the Digitiser 80 times and displays the results on the screen. It then waits until you press any key. Then it takes a further set of 80 readings. Table 2.1 shows what values to expect. Place the scanner on white paper and press the "Send" button. The reading should be 7. Try it on black paper also and on various shades of grey. You can adjust the variable resistors slightly so that the readings change at the levels of grey which you decide on.

The only possible readings when the button is pressed are 0, 1, 3 and 7 depending on the brightness of the pixel. When the button is not pressed, you will get 8, 9, 11 and 15. If you obtain other readings, there is something wrong. Write down, in binary, the numbers you *expect* to get. By comparing these you may find that one of the data lines is "stuck", always giving "high" or "low" when it should be changing. If so, examine the wiring of this line, looking for short circuits to next-door data lines or to the power lines. Look also for gaps and breaks in the line, dry solder joints and other possible bad connections.

The logic circuits are not "tricky" in the sense that they need careful adjustment to get them to work. If you obtain your ICs *new* from well-known suppliers, they are very unlikely to be faulty, so if the circuit does not give the expected results, it is nearly always a fault in the construction.

Programming

The program below shows how the Commodore 64 is used to read data from the scanner and plot it on the screen:

```
10 G=1024:H=55296:REM BASE ADDRESSES OF
    VIDEO AND COLOUR RAMS
20 POKE 56579,240
30 POKE 56577,192
40 PRINT"▽"
50 R=PEEK(56577)AND15
60 FOR J=1 TO 100:NEXT
70 IF R>7 THEN 50
80 IF R=0 THEN C=0
90 IF R=1 THEN C=12
100 IF R=3 THEN C=15
110 IF R=7 THEN C=1
120 POKE H+K,C:POKE G+K,160
130 K=K+1
140 IF K=1000 THEN140
150 R=PEEK(56577)AND8
160 FOR J=1 TO 100:NEXT
170 IF R=0 THEN150
180 GOTO 50
```

The scanner is moved across the picture as shown in Fig. 2.5. First, a ruler is laid across the picture, with its end aligned with two scales drawn down the sides of the picture. The scanner is moved step by step from left to right along the ruler. The "Send" button is pressed each time the scanner is put in a new position. At the end of the line, the ruler is moved down one step, and the scanner returned to the left-hand end. The next line is scanned. The program builds up the picture by printing row after row of coloured blocks on the screen. The blocks are printed by POKEing 160 (the code for a reversed "space") into video RAM while at the same time POKEing the required colour into the corresponding address in the colour RAM. The colour codes are represented by variable C, which is given the values 0 (black), 12 (medium-grey), 15 (light-grey) or 1 (white) according to the

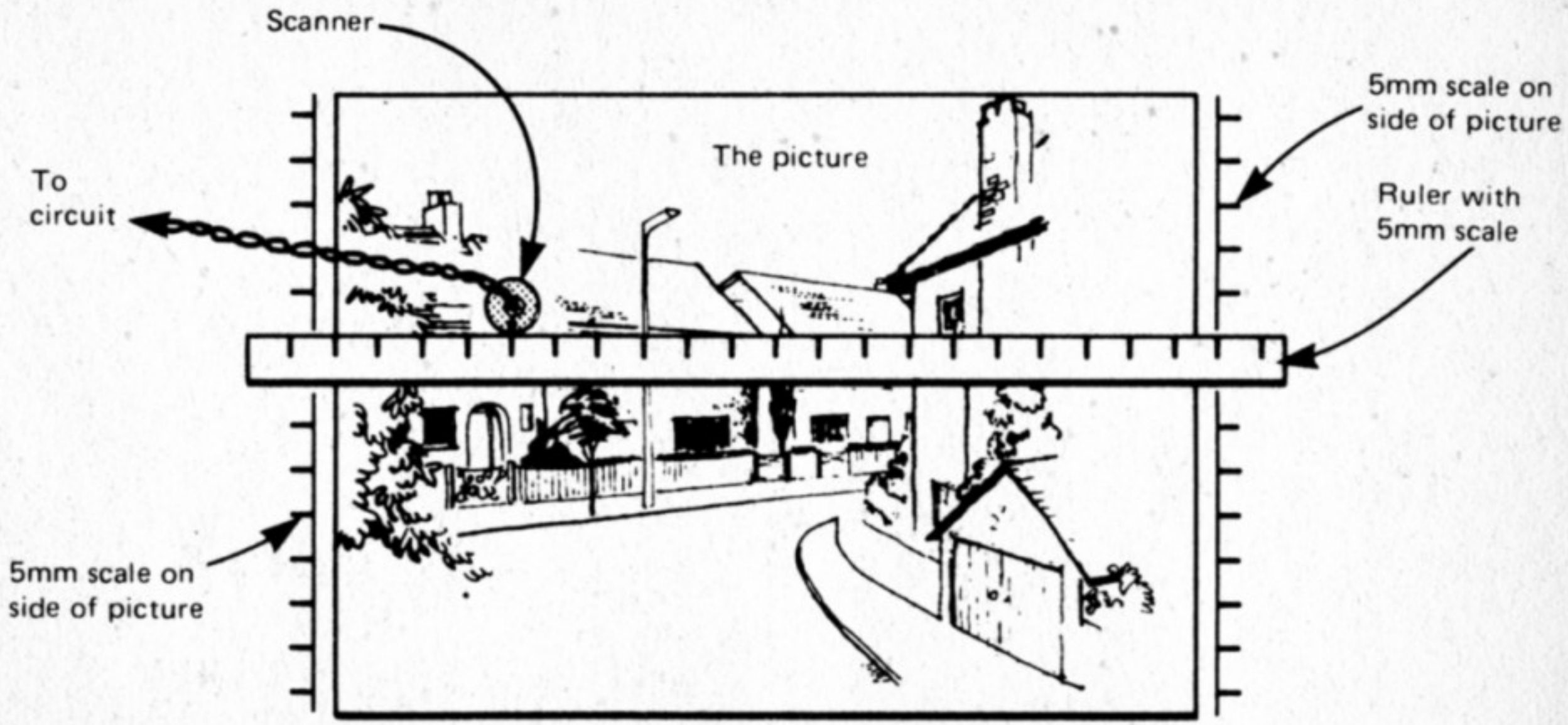


Fig. 2.5 Scanning a picture

reading R obtained from the Digitiser.
The program for the VIC-20 starts with:

```
1Ø G=768Ø:H=384ØØ
```

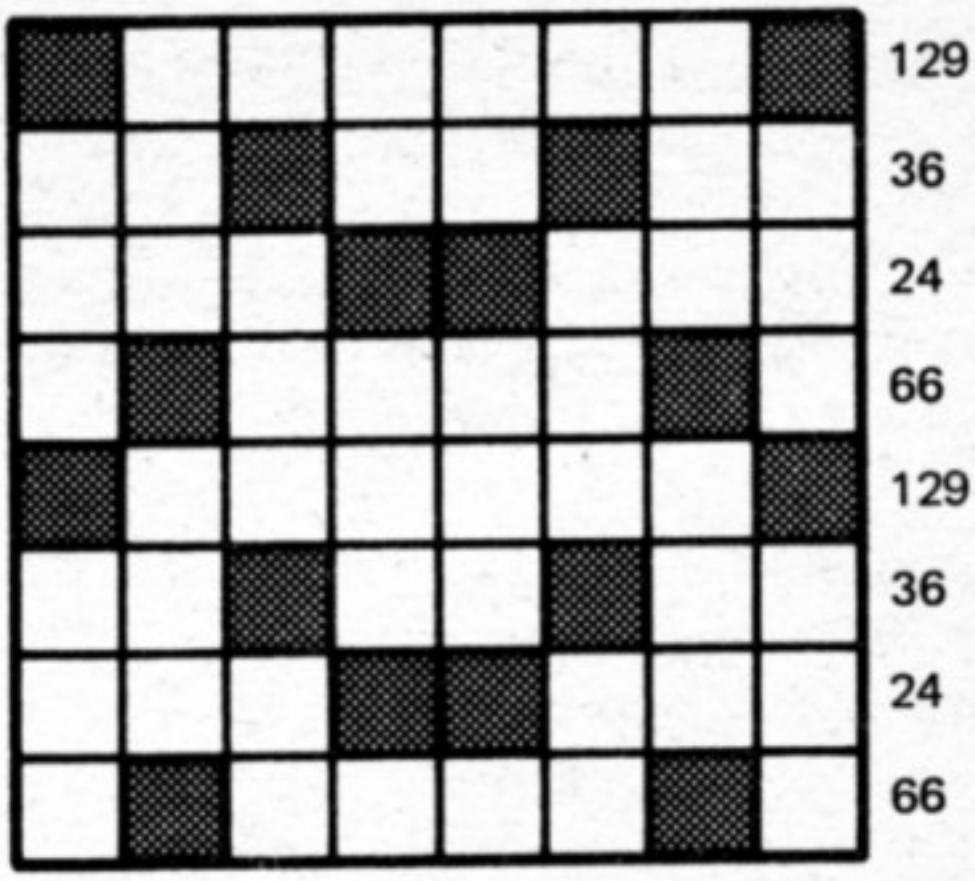
However, if you have added 8K or more of extra RAM to your system, the values of G and H are 4096 and 37888, respectively. Lines 10–70 are the same as above, except for the addresses to be POKEd and PEEKed (p. 15). Since the VIC does not have greys, a shaded graphics block must be used instead. If you are content to have only one grey, use the grey block on the "+" key. Lines 80–120 are:

```
8Ø IF R=Ø THEN B=16Ø
9Ø IF R=1 THEN B=1Ø2
1ØØ IF R=3 THEN B=1Ø2
11Ø IF R=7 THEN B=32
12Ø POKE H+K,Ø:POKE G+K,B
13Ø K=K+1
14Ø IF K=5Ø6 THEN 14Ø
15Ø R=PEEK(37136)AND8
```

Lines 160–180 are the same as in the program for the Commodore. Lines 90–120 select a different block character to be printed, where B is its screen code. At line 120 the colour RAM is POKEd with 0 to make it display the block in black (although you could use any other colour if you prefer) while video RAM is POKEd with the appropriate kind of block (space, shaded or solid). If you want two shades of grey, you will need to define two special characters to give light- and dark-greys. Fig. 2.6 suggests suitable designs. There is not room here to describe how to do this. The VIC-20 User Manual supplied with the computer does not deal with user-defined graphics, but full explanations may be found in *Get More from the VIC-20* by Owen Bishop (Granada) and *The VIC-20 Programmer's Reference Guide* (Commodore).

This program requires the picture to be scanned in 23 rows, with 22 pixels in each row.

The next program, written for the BBC Microcomputer, also relies on user-defined graphics. The BBC User Guide gives full details. In the following program it is assumed that



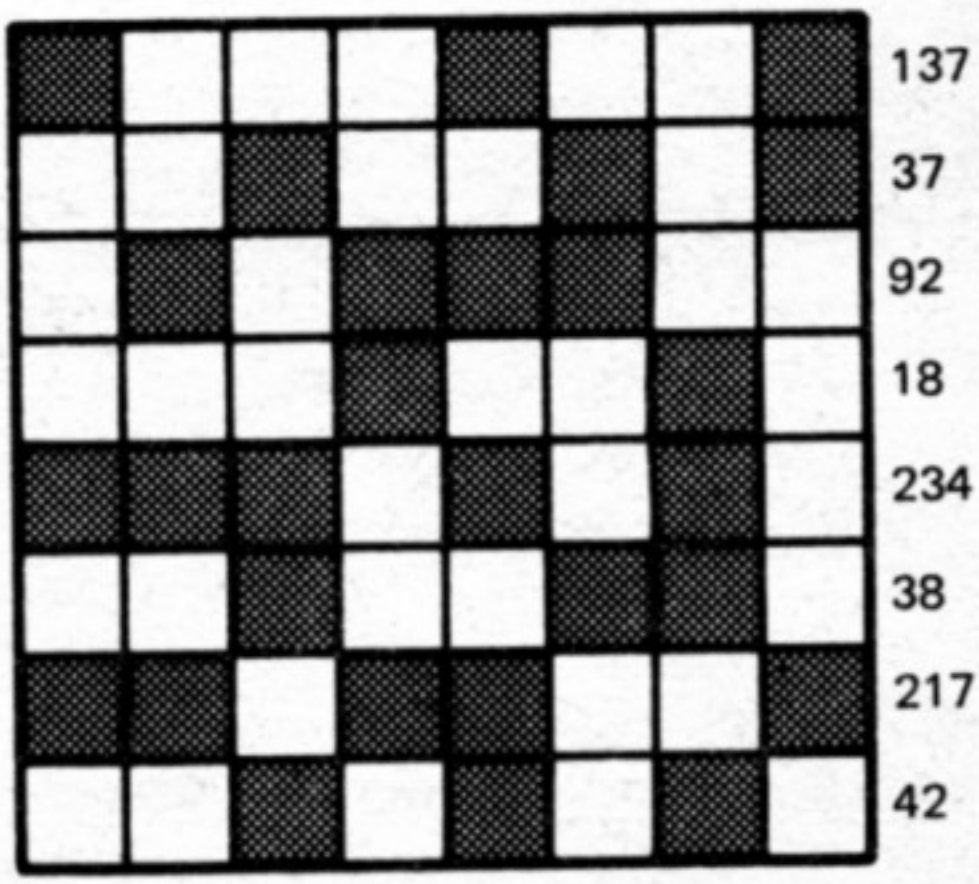


Fig. 2.6 User-defined grey character for the VIC-20 and BBC microcomputers. Enter the values given on the right of each block

ASCII code 224 is used for white (a solid block), 255 for light-grey and 226 for dark-grey. Black is obtained by using a space:

```
10 MODE 4:VDU 23;8202;0;0;0;:X=0:Y=0
20 A%=&97:X%=&62:Y%=240:CALL&FFF4
30 X%=&60:Y%=192:CALL&FFF4
40 CLS
50 REPEAT PROCdigitiser(15)
60 J=1 TO 100:NEXT
70 UNTIL R<8
80 IF R=0 THEN code=32
90 IF R=1 THEN code=224
100 IF R=3 THEN code=225
110 IF R=7 THEN code=226
120 PRINTTAB(X,Y)CHR$(code)
130 X=X+1:IF X=40 THEN X=0:Y=Y+1
140 REPEAT PROCdigitiser(8)
150 FOR J=1 TO 100:NEXT
160 UNTIL R<>FALSE
170 IF X<32 THEN 50
180 END
500 DEF PROCdigitiser(mask)
510 A%=&97:X%=&60:Y%=&192
520 CALL &FFF4
530 A%=&96
540 R=USR(&FFF4)AND(mask*&10000))/&10000
550 A%=&97:Y%=0
560 CALL &FFF4
570 ENDPROC
```

This program will operate over the tube, if required. The VDU statement in line 10 disables the cursor. The procedure is a little more complicated than necessary since it enables and disables the buffers of the Digitiser each time it is read. It was written in this way to make it more generally useful in other programs using the Digitiser. This program requires the picture to be scanned in 32 rows with 40 pixels in each row.

This program reads from the scanner and plots the results

on the screen in 25 rows, each consisting of 40 pixels. Assuming the aperture of the scanner measures 5 mm × 5 mm, the maximum size of the picture scanned is 20 cm wide and 12.5 cm deep. If you want to scan a bigger picture, enlarge the aperture of the scanner.

Parts Required for the Picture Digitiser

Resistors, carbon, 0.25 W, 5% tolerance

R1 100 k
R2–R4 1 k (3 off)

Variable Resistors

RV1–RV3 100 k miniature horizontal preset (3 off)

Semiconductors

D1 BPX65 or similar photodiode
Q1–Q3 ZTX300 or similar general-purpose npn transistors (3 off)

Integrated Circuits

IC1–IC3 311 comparator (3 off)
IC4 74LS125 quad bus buffer gate with three-state output

Miscellaneous

S1 Push-to-make push-button switch
Circuit board
8-pin IC sockets (3 off)
14-pin IC sockets (1 or 2 off)
6-way socket to fit 6-way plug of Decoder board
3-way socket to fit 3-way plug of Decoder board
Lamp for scanner, low voltage (2.5 V to 6 V), 0.3 A (the type which has a lens moulded into the end of the glass bulb is strongly recommended, as it focusses a bright spot of light on to the paper; available from most shops which sell electric torches)
Socket for lamp
Materials for making scanner
Connecting wire, including thin flexible wire for connections to scanner

PROJECT 3

Five-key Pad

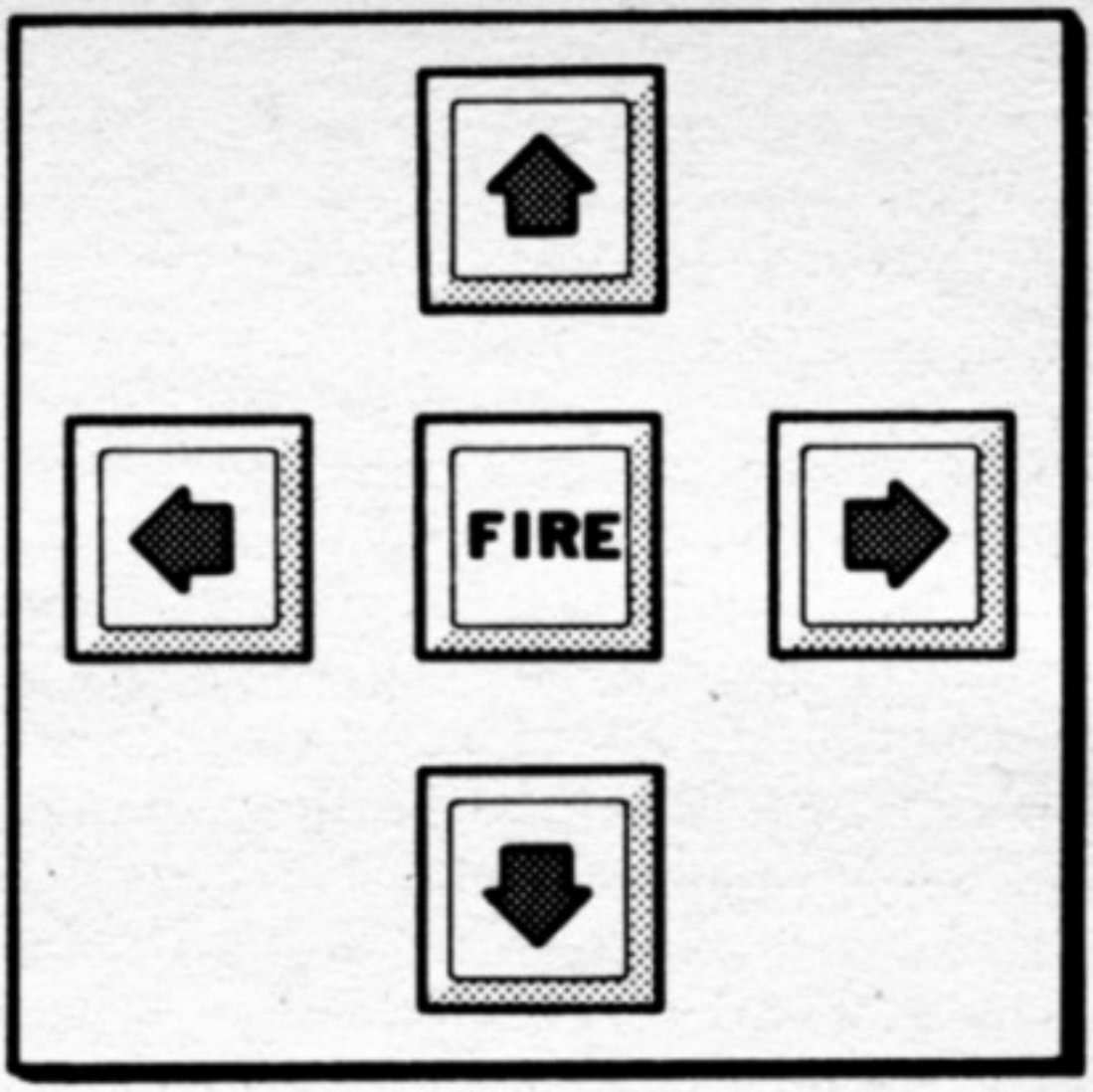
Although it is usual to control the movement of the cursor, or to aim and fire your space-gun against the aliens, by using the keys the standard keyboard of the micro, it is very useful to have a separate keyboard specially designed for this purpose. This keyboard has 5 keys, which you may use in lots of different ways.

For instance, mark four of them with arrows pointing up, down, left and right, and use them for steering the cursor or anything else around the screen. The keys are arranged as in Fig. 3.1 so that it is easy to remember which is which and to find the right key quickly. Another way of marking the four keys would be according to the points of the compass – N, S, E and W.

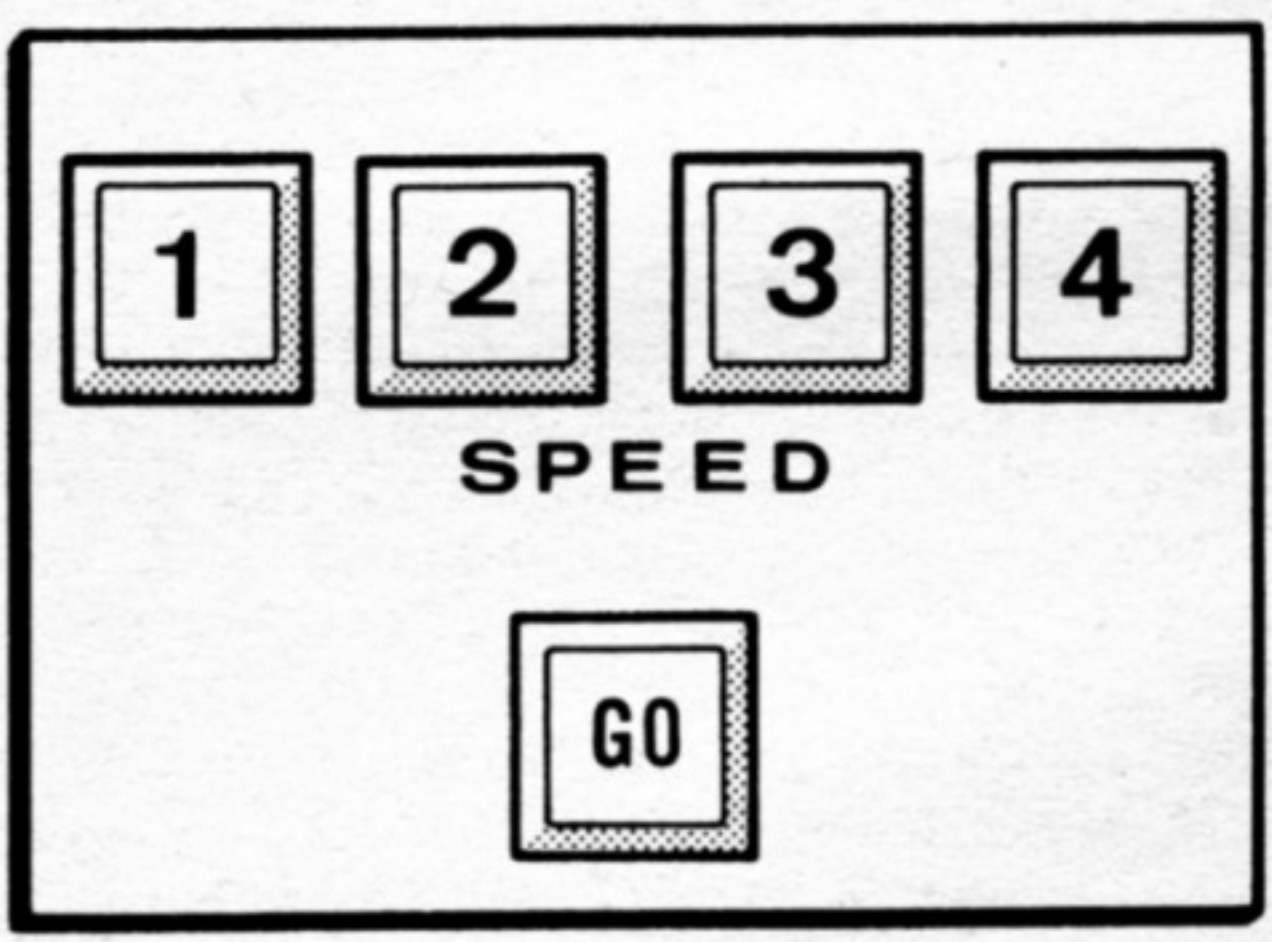
The fifth key is a control key which has several possible functions, depending on how you program the micro. In a games program you can use it to fire the space-gun, or perhaps to indicate "no move". In a program which draws pictures on the screen, under the control of the four "arrow" keys, the fifth key can be used to indicate "pen down" (move the cursor to a different position and draw a line as it moves) or "pen up" (move the cursor to a different position, but do not draw a line).

The keyboard can also be used as a special-purpose controller for a model-railway system or for slot cars (*see* Project 4) with the keys being marked "Go", "Faster", "Slower", and "Reverse" (for locomotives at least!). The fifth key can be "Emergency Stop". A second keyboard can be used for controlling points or signal lamps.

Many two-player games are improved if each player has an individual keyboard. They are especially useful if it is the sort of game in which each player has to "make a move" without letting the other player know which move is being made. Also, the computer can be programmed to read one



(a)



(b)

Fig. 3.1 Two possible layouts for a Five-key Pad; (a) direction arrows for games control, with firing key; (b) slot car speed controller – press "go" and one of the numbered speed buttons then release "go" key

keyboard or the other, depending on whose turn it is to play. Playing out of turn is prevented.

The keyboard can be used as a remote-operating keyboard. You place the micro in one room and the keyboard in an adjacent room and send instructions to the micro along a 6-wire cable.

How It Works

Each key is connected to one input of an exclusive-OR gate (Fig. 3.2). An exclusive-OR gate has two inputs. When both inputs are alike (both "high" or both "low") the output of the gate is "low". The output is "high" only when the inputs differ. The IC used here belongs to the TTL family. The inputs of this family have the property that if they are disconnected they count as a "high" input. Thus, when a button is not pressed the corresponding input acts as if it were a "high". Pressing any button makes its input "low". The effect of pressing various combinations of buttons is shown in Table 3.1.

There are two designs for this project. The simpler design is one that plugs directly into the games port (otherwise known as the controller port) of the VIC-20. It does not require the use of the Decoder. The Commodore 64 can have two Five-key Pads plugged into it, one to each games port. The design consists only of IC1, the outputs from the gates being taken directly to the games port.

The other design includes buffers on each output line (IC2). This circuit is to be plugged into the Decoder in the same way as most other projects in this book. Use this design with the BBC Microcomputer, or with the VIC-20 if you wish to attach a second Five-key Pad to the machine.

The output of each exclusive-OR gate goes to a buffer with three-state outputs. The outputs of these buffers are connected to data lines PB0 to PB3. The buffers are made to put data on the bus by bringing the enable line low. This is connected to an addressed output of the Decoder (*see* Fig. D.0). This enables the buffer, and the data from the buffers is put on to the input lines of port B.

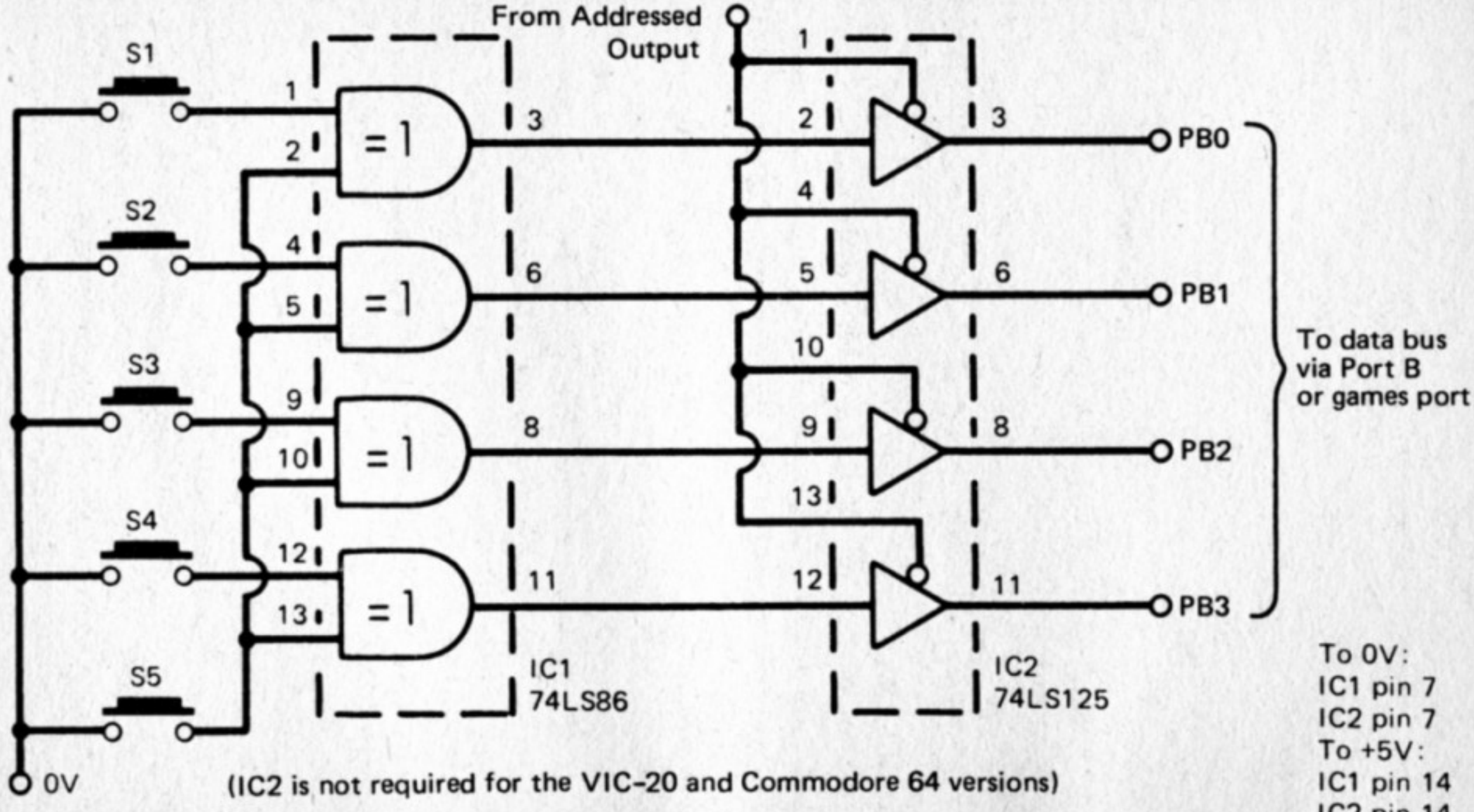


Fig. 3.2 Circuit diagram of the Five-key Pad. S5 is the "control" or "fire" key

Table 3.1 Data output from the Five-key Pad

Keys pressed 5 4 3 2 1	*Data lines* 3 2 1 0	*Decimal equivalent*
− − − − −	0 0 0 0	0
− − − − +	0 0 0 1	1
− − − + −	0 0 1 0	2
− − + − −	0 1 0 0	4
− + − − −	1 0 0 0	8
+ − − − −	1 1 1 1	15
+ − − − +	1 1 1 0	14
+ − − + −	1 1 0 1	13
+ − + − −	1 0 1 1	11
+ + − − −	0 1 1 1	7

− = Key not pressed + = Key pressed 0 = "low" 1 = "high"

Building It

Although it is possible to use any kind of switch for the keys, it is best to employ the type specially designed as keyboard keys. Usually the switch is sold separately from the key top. Key tops fit on to the switch and are of various types. Some makes of keys have tops in a range of bright colours. Others have white tops with transparent covers. You write a letter or symbol on the top (or use rub-down lettering) then clip the transparent cover over the top to protect the lettering from wear. Alternatively, draw the symbol or letter on a small square of card and insert this before clipping on the cover.

The project is best mounted in a small plastic case. Cases can be obtained with a sloping top which makes them very suitable for use as a keyboard enclosure. Alternatively, a plastic box used for packing foods and other commodities is generally obtainable free and may be used for housing this project.

The layout and construction of the circuit presents no

problems. The two ICs and the Decoder IC can easily be accommodated on a small piece of stripboard which will fit into the case. Cut an aperture in the lid for the keys.

There are 6 leads between the Pad and the games port connector. Since the device requires very little current, it may be powered from the VIC-20 or Commodore 64. The connector is a 9-way "D" socket, as shown in Fig. 3.3. Note that the output from pin 11 of IC1 goes to *pin 6* of the "D" socket if the Pad is intended for the VIC-20, but to *pin 4* of the socket if it is intended for the Commodore 64. The leads may be a metre or so in length to allow you to operate the Pad in any convenient position.

The BBC version requires 6 leads (0 V, +5 V, PB0, PB1, PB2, and PB3) running to a 6-way plug, and a single lead (addressed output) running to a 3-way plug.

Testing

When construction is complete, test the circuit to make sure that there are no short-circuits between next-door data lines or between these lines and the power lines. Also make sure that there is no short-circuit between the two power lines 0 V

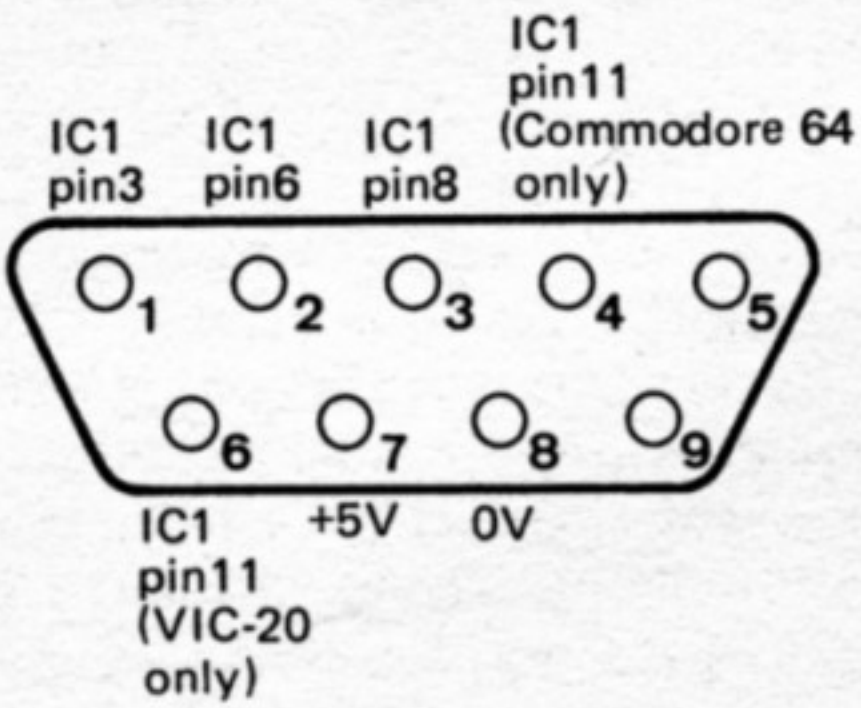


Fig. 3.3 Connections to be made to the VIC-20 and Commodore 64 games port(s)

and +5 V. Connect the circuit to a 5 V power supply and temporarily connect the addressed output to the 0 V line. Then press each button in turn, while measuring the output from the corresponding buffer with a voltmeter. Table 3.1 shows what to expect.

The Pad may now be plugged on to the Decoder which is plugged into the micro. Switch on the micro. If the usual display fails to appear on the screen, *switch off immediately* and repeat your checking of the circuit. Now run a simple test program.

Here is a test program for use with a Five-key Pad plugged, into the games port of the VIC-20:

```
1Ø FOR J=1 TO 8Ø
2Ø PRINTPEEK((37137)AND6Ø)/4;
3Ø NEXT
4Ø GET A$:IF A$="" THEN 4Ø
5Ø GOTO 1Ø
```

This program reads and displays the input from the Pad 80 times. It repeats when any key is pressed. Note that it uses the address of *port A* of VIA no. 1, so there is no problem in having the Pad connected to the computer at the same time as the Decoder with any attached add-ons, which always use port B.

The test program for the Commodore 64 is the same as above except for line 20:

```
2Ø PRINTPEEK(56320)AND15;
```

This is for a Pad plugged into games port 1. For games port 2, use the address 56321 instead.

For testing the BBC version plugged into the Decoder, use the program given on p. 34. In line 20, substitute for "192" the appropriate value from Table 0.2, depending on which addressed output is being used to enable the Pad.

When the pad has been checked and found to be in working order, the next step is to make use of it on one of your own programs. Exactly how you do this depends on what sort of program it is. The principle is simple: read the Pad, and then branch to different parts of the program depending on

the value that is obtained. The program of p. 39 illustrates two points about reading from keys. A micro works fast, and it can read a key hundreds of times while it is being pressed just once! The program on p. 39 is controlled by pressing the "Send" key of the Picture Digitiser. In lines 50 to 70 it waits in a loop until it detects that the key has been pressed. The pause in the loop is intended to overcome the contact, bounce of the keys. When a key is pressed it seldom changes straight from off to on, but is more likely to go on-off-on-off-on-off and so on several times until it finally settles at on. The pause allows it time to settle to fixed state before taking the next reading. The main part of the program deals with the reading so obtained. Then, before taking the next reading, the program must check that the key has been released. It waits in another loop (lines 150 to 170) until it detects that the key is no longer being pressed. Then it jumps back to the beginning of the program to wait for the key to be pressed the next time.

You may need to use routines of this sort in your Key-pad program if you want to interpret a series of key presses as a series of commands. Of course, if the keys are simply to be pressed and held down for as long as a particular action is to continue, you simply read the Pad repeatedly without pauses until a change of input occurs.

The VIC-20 and Commodore 64 normally have single action on all except a few control keys, so holding the key down will not work with these machines. However, it is possible to make these machines operate with repeat action on all keys by the statement:

```
POKE 650,128
```

This applies to both machines.

Parts Required for the Five-key Pad

Integrated Circuits

IC1 74LS86 quad exclusive-OR gate

IC2 74LS125 quad bus buffer gate with three-state outputs (BBC version only)

Miscellaneous

S1–S5 Key switches with tops (5 off)

14-pin IC sockets (1 or 2 off)

6-way socket to fit 6-way plug of Decoder board (BBC only)

3-way socket to fit 3-way plug of Decoder board (BBC only)

9-way "D" socket (for VIC-20 and Commodore 64 versions)

Key-pad case

Connecting wire

PROJECT 4

Model Controller

Putting a micro in command of a model-railway system adds greatly to the realism of its operation. The locomotive can be stopped, started, or reversed and its speed can be varied in a number of stages. The project can also be used with other electrically powered models such as slot cars. If you are keen on building your own robot, this project provides a way to control its actions. In fact, any model which is driven by low-voltage DC motors, or is activated by electromagnets can be controlled through this interface. On the model-railway scene, it may be used for switching points. The interface can also switch lamps, so is ideal for signal switching.

Many of the circuits used for controlling the speed of a motor require that the power source should have a higher output voltage than is actually needed for driving the motor at top speed. This circuit employs relays to do the switching, so it is able to take power from the transformer or power pack that you normally use for your model railways or slot cars. There is no need to obtain or build a special power supply. The relays do not cause loss of voltage, so the motors run at top speed when you want them to do so.

The project as described here is suitable for use with a model railway but you will find it easy to adapt it to slot cars or other models. One point about the circuit is that you do not need to build it all at once. You can start with just one relay and expand it with further relays from time to time. As your model-railway system grows, you can build a second or even a third version of the project, perhaps using one for controlling the locomotive and the other for controlling points and signal lamps.

How It Works

Fig. 4.1 shows the relay-control side of the project. IC1 con-

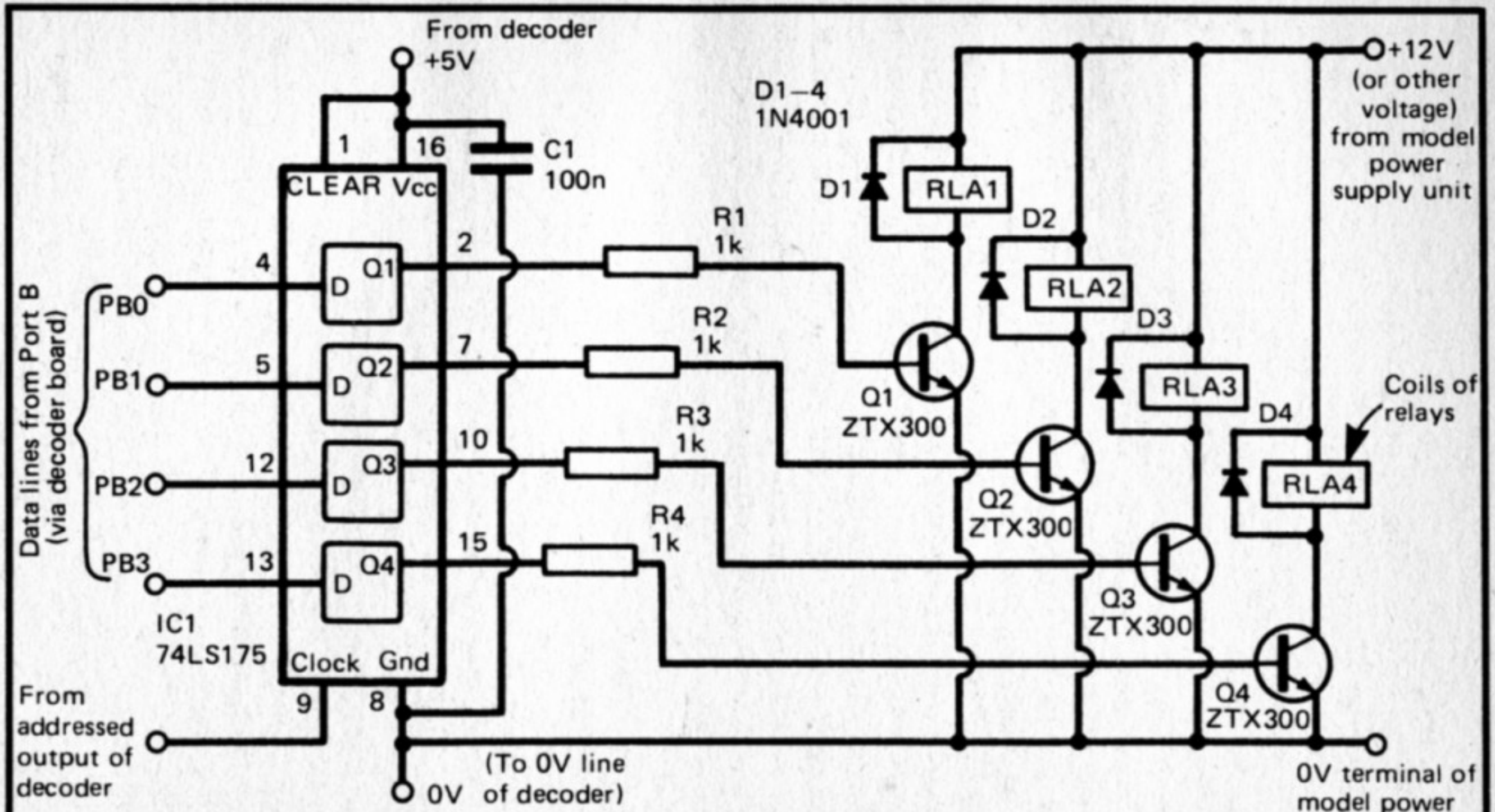


Fig. 4.1 The relay-controlling circuit of the Model Controller

tains four latches. The D (for data) inputs of these are connected to the 4 data lines of port B. The IC has a "clock" input (pin 9). This is connected to one of the addressed outputs of the Decoder (Fig. D.0).

When using this device we define *all* lines of port B as outputs. The lower four lines (PB0 to PB3) send data to the add-on, instead of receiving data from it, as is the case with the other projects in this book. Having defined all lines as outputs, we make a line high when we want the corresponding relay to be energised. At the same time, the addressed output is addressed for an instant. One of the other Outputs (e.g. output 0) is addressed immediately afterwards. This has the effect of producing a brief low pulse at the addressed output. This low pulse triggers the latches of IC1. The output (Q) of each latch becomes the same as the input (D) on the corresponding line of port B. Once the triggering pulse is over, Q does not change even though D may change. D is very likely to change, for the computer will probably redefine the lower four lines as inputs again, and use them to receive data from other add-ons. But the data which was on the lines at the instant the triggering pulse occurred is held unchanged (latched) at Q until the next time that the micro sends data to the Controller.

The output of each latch supplies base current to a transistor. If Q is "low", there is no base current. If Q is "high", base current flows and the transistor is turned on. This causes a collector current to flow, which activates the coil of a relay. What happens then depends on how the relay is wired.

Fig. 4.2 shows a typical relay circuit for controlling a model locomotive. Note that the power supply for the relay coils and for the locomotive all come from the power-supply unit which belongs to the model railway. This *must* be a DC supply and you must not use the circuit with voltages higher than 25 V, for this would damage the transistors. The normal voltage for models is 12 V or less, so this circuit is very likely to be suitable for all your models.

Relay 3 (RLA3) is a simple on-off switch. It is wired so that the circuit is closed (the locomotive moves) when the

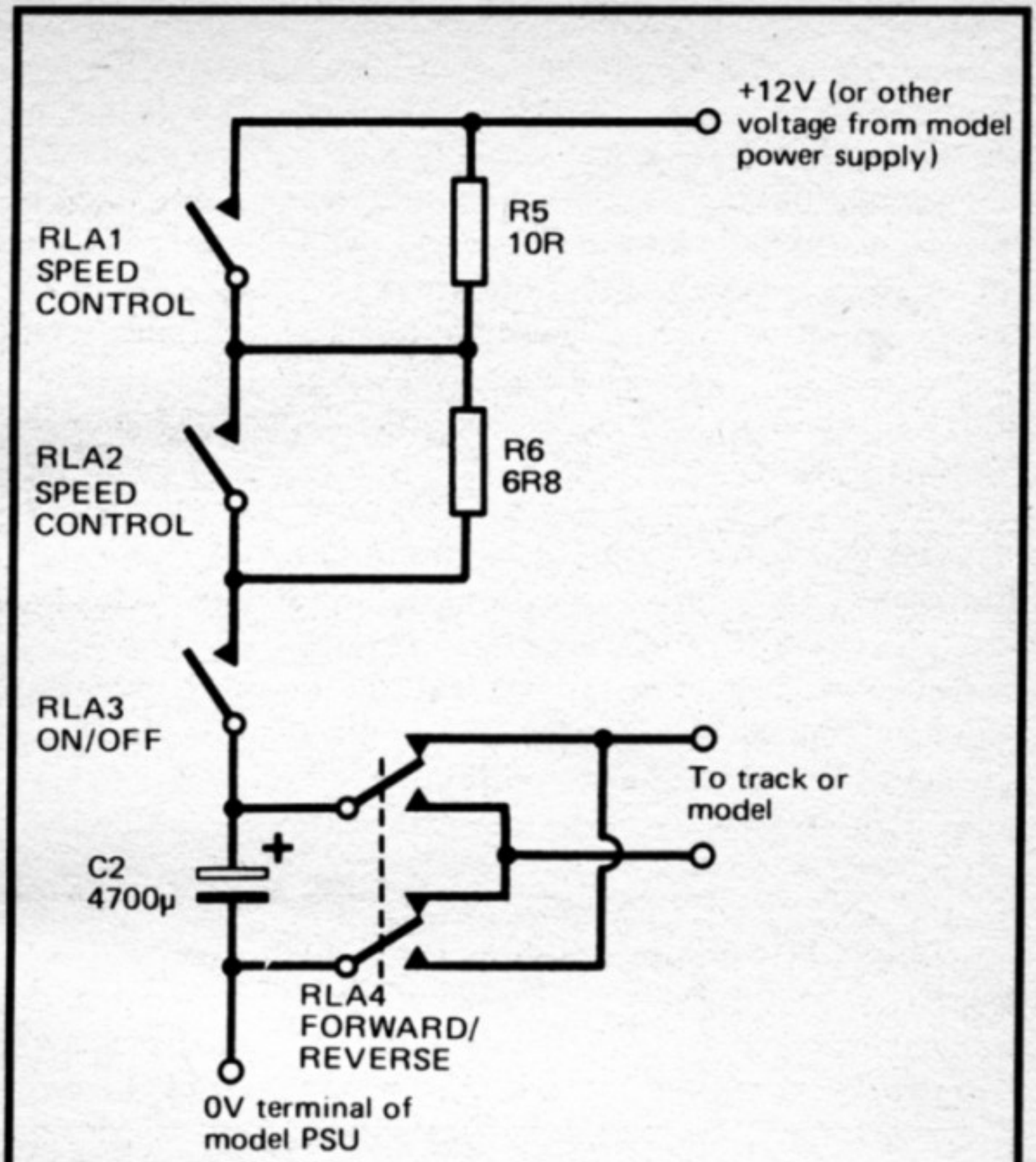


Fig. 4.2 The way the relay contacts are wired for a typical Model Locomotive Controller

output of the latch is "high" and the transistor is turned on. The current to the track may pass through two resistors (R5, R6), but either or both of these may be short-circuited by closing relays 1 and 2. These are wired so that they are open when the output of their latches is "low".

When relays 1 and 2 are both open, the current to the track has to pass through R5 and R6, putting a resistance of 16.8 ohms in series with the motor of the locomotive. This

gives a slow starting speed. Incidentally, the values of resistors shown in Fig. 4.2 are those which were used with a particular model locomotive when this circuit was being designed and tested. You will probably find that you need quite different values in your circuit, though those given should be good as a starting point for your own trials. When one of relays 1 and 2 is closed, the resistance in series with the motor becomes 10 or 6.8 ohms. These reduced resistances give two higher speeds. Finally, both relays may be closed, reducing the resistance to zero and putting the locomotive into top speed.

Relay 4 is a reversing relay. The relay has two pairs of contacts which are changed over simultaneously. As a result, the current flows one way or the other around the circuit and the locomotive runs forwards or in reverse.

Table 4.1 shows the outputs required from the latches in order to produce differing speeds and direction.

Table 4.1 Command values for the Controller

Direction of motion	*Speed*	*Latch outputs* Q4	Q3	Q2	Q1	*Decimal equivalent*
Forward	STOP	0	X	X	0	0, 2, 4, 6
	Slow	0	0	0	1	1
	Medium	0	0	1	1	3
	Fast	0	1	0	1	5
	Top	0	1	1	1	7
Reverse	STOP	1	X	X	0	8, 10, 12, 14
	Slow	1	0	0	1	9
	Medium	1	0	1	1	11
	Fast	1	1	0	1	13
	Top	1	1	1	1	15

0 = "low" 1 = "high" X = "low" or "high"

Building It

The single IC takes up very little room, but allow plenty of space for the relays and for R5 and R6. C2 also takes up a lot of room. It is essential to install this capacitor, for a locomotive produces sparks as it crosses the joins between one rail and the next, or if it encounters a part of the track where the rails are rough or slightly corroded. These sparks generate voltage "spikes" which can be transmitted through the circuit and reach the micro. They are not likely to do any harm to the micro but may cause some of the memory locations to change state. This leads to rather odd error messages appearing which seem to bear no relation to the program. If interference is worse the program may "crash". This capacitor completely eliminated all such interference in the prototype circuit. C2 should be wired as close as possible to the two terminals which supply power to the track. If a reversing relay is fitted, C2 must be as close as possible to this, on the side nearer the other relays (*see* Fig 4.2).

There are similar reasons for C1. Latches have the habit of becomming "unlatched" if there is interference around. They are sensitive even to the voltage surges caused by switching the relays on and off. Position C1 so that its wires are soldered as close as possible to the power terminals of IC1 (pins 8 and 16).

The final protective feature is the diodes. When relays switch off, a large current is generated. This could eventually damage the transistors. The diodes conduct this current safely away to the +5 V line. The diodes should be wired as close as possible to the terminals of the coils of the relays.

The Controller is connected to the Decoder board at two points. One connection is to the data lines, this cable requires 4 wires and may be plugged on to one of the 6-way plugs. The other connection is to one of the 3-way plugs with an addressed output – this needs 3 wires, one for the addressed Output, the others being the 0 V and the +5 V lines.

As mentioned earlier, there is no need to build the whole circuit at once. If you want to try it out on a small scale first,

wire up just IC1, R1, Q1, D1, and RLA1. You also need to include C1 and C2. This will allow you to experiment with controlling one relay which you might use, for example, simply to start and stop the locomotive. Speed control and reversing can be added later. At that stage you may decide to use only one relay for speed control (giving just 3 speeds – slow, medium, top speed), so freeing a relay for use in switching points or operating signal lamps. The main point is that, if you start with a small system, build it on a board large enough to accommodate future expansion.

As mentioned above, the most suitable values for R5 and R6 depends on the characteristics of the motor of the locomotive or other model. The best course is to try connecting the resistors in series with the locomotive *before* wiring them permanently into the circuit. The amount of current taken by the motor may be 1 ampere or more, so it is essential to use resistors rated at 2.5 W at least. Fixed-value wire-bound resistors are available cheaply. It is possible to obtain variable wire-wound resistors (3 watts), but these are relatively expensive so the best course is to buy a selection of fixed-value resistors and experiment with these. If you are following the two-relay/two-resistor scheme of Fig. 4.2, first find out what resistance is enough to let the locomotive run at its lowest steady speed. The speed must be such that the motor does not stall when the locomotive crosses a gap in the track or when running round sharp bends. Also, it must allow a stationary locomotive to begin moving while pulling the heaviest train it is likely to have to pull. Having established the correct value (which is the total value of R5 and R6), divide this value into two parts, one rather larger than the other. In Fig. 4.2, the total value is 16R8 ohms, broken into 10 R and 6R8 ohms. These values are, of course, the nearest standard values obtainable. If you need a value which is non-standard, it is often possible to join two or more resistors in series; for example, you can make up a resistance of 7R5 by joining 1 R, 1 R, 2R2 and 3R3 ohms in series. Often, an easier method is to put two larger resistors in parallel. In this instance, wire two 15 R resistors in parallel to obtain 7R5 ohms. When wired in parallel, and providing the resis-

tors are more or less equal in value, they *share* the current. Resistors of lower rating may be used, such as 1 W or 2 W carbon resistors.

Before connecting the Controller to the micro, test it to make sure that there are no short circuits between adjacent data lines, between the data lines and the power lines, and between the two power lines. The model's +12 V (or other) power line does *not* connect directly to any line going to and from the micro, but its 0 V line *must* be connected to the 0 V line of the project. To test the operation of the circuit, connect the Controller to the track, and to the power lines of the Decoder (plugged into the micro), but do not connect the data lines or the addressed output yet. Use leads with crocodile clips to connect the data line terminals and addressed output terminal of the Controller to the 0 V line.

Switch on the power supply to the Controller and the railway power-supply unit. Connect the lead of terminal PB0 to the +5 V line. Then connect the addressed output lead (i.e. the one which connects to the "clock" terminal of IC1) to the +5 V line. As soon as it makes contact with the +5 V line, the output of latch 1 changes from "low" to "high", because of the "high" on its input. Often it changes *before* you actually touch the lead against the +5 V line, for the slight rise in voltage caused by taking it away from the 0 V line is enough to trigger the latches. Putting the lead back on to the 0 V line has no further effect. As the output of latch 1 goes "high", the locomotive should start moving slowly. Now put the PB0 lead back on to the 0 V line, touch the addressed output ("clock") lead to the +5 V line and the locomotive stops. This is because the output from latch 1 has now returned to "low". In a similar way, check the action of the other latches and relays.

Programming

The sequence required is this:

(i) Define all lines of port B as outputs.
(ii) Send to port B a byte made of of the data to be sent,

plus the address of the addressed output (Table 0.2). This puts the required data on the line and begins the low pulse to the clock input of IC1.

(iii) Send to port B a byte consisting of data only. This holds the data on the lines, and also makes the addressed output go high again. This causes the data to be latched.

(iv) Redefine port B with 4 input lines and 4 output lines, ready for use by other add-ons. (This step is not required if the Controller is the only add-on currently in use.)

Defining all lines as outputs requires the same commands as defining it with 4 input and 4 output lines. Use the programs given on p. 15 but instead of POKEing (or indirecting) the value 240 to DDRB, POKE (or indirect) the value 255.

Fig. 4.3 shows how to make up the byte which is to be sent

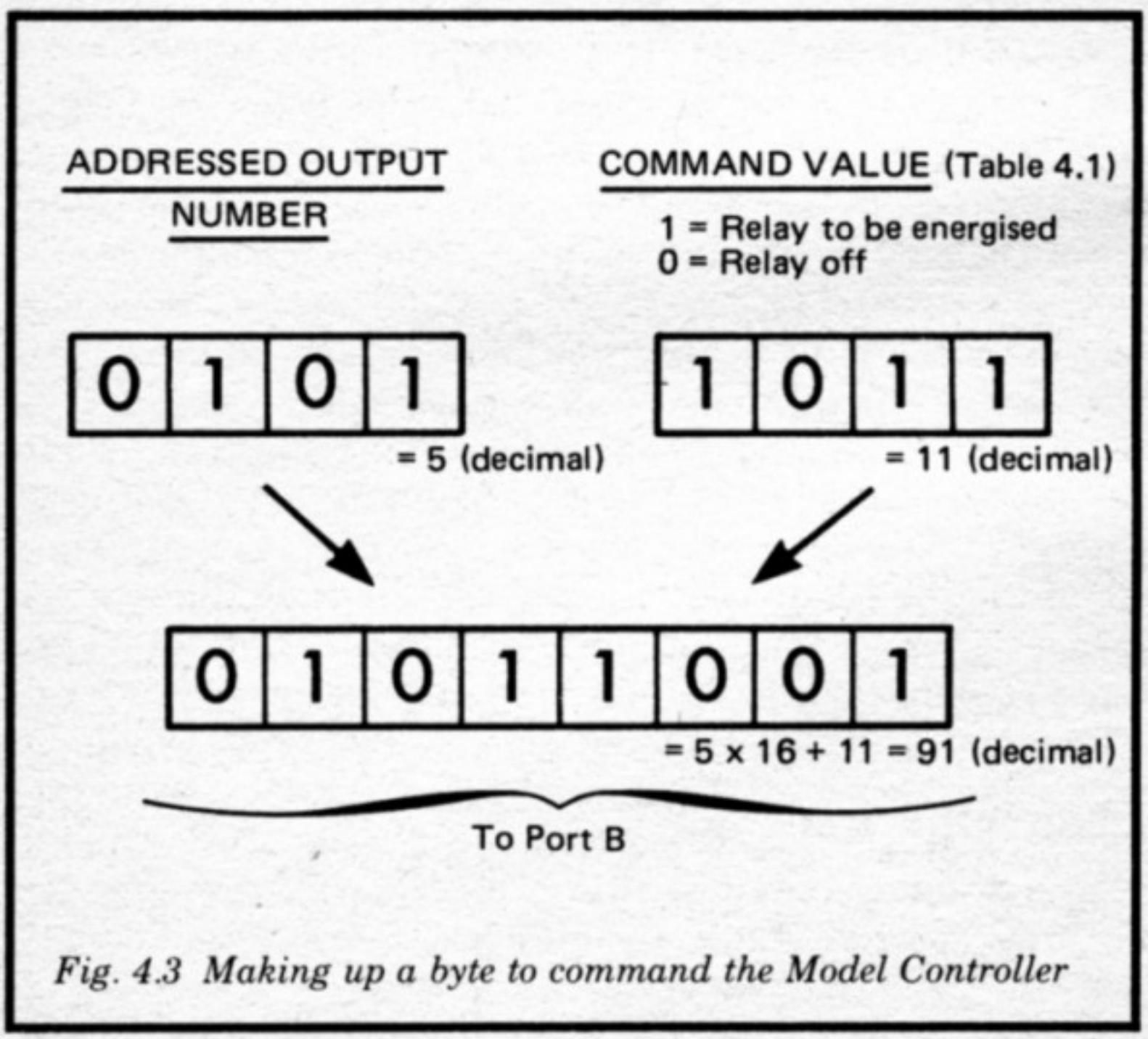


Fig. 4.3 Making up a byte to command the Model Controller

to port B at stage (ii) previously. In the example given in the Figure, the Controller is plugged to addressed output 5, so its Decoder address is 5 times 16, or 80. This number appears in the upper four bits only. The lower four bits indicate which relays are to be energised. In this example we are making the locomotive run in reverse at medium speed (*see* Table 4.1). To calculate the value to be POKEd (or indirected), just add the Decoder address (Table 0.2) to the command value (Table 4.1). Their total is sent to port B using exactly the same program lines as are used to trigger the addressed output. The *effect* of the programming is different because the lower four lines are now defined as outputs. Not only is the addressed output triggered but data is sent to the latch of the Controller. Stage (iii) is also exactly the same as triggering an addressed output. POKE (or indirect) the command value taken from Table 4.1 but *without* adding the addressed output number. If step (iv) is required, this is performed in the normal way.

There is one point to be considered when starting the locomotive (or other motor) from rest. Some model power-supply units have an automatic cut-out which shuts down the supply if it becomes overloaded. If you try to accelerate the locomotive too rapidly by starting it off at top speed (relays 1, 2 and 3 together), the sudden surge of current may trip the cut-out. You will then have to reset the power unit by hand. If your unit is of this type, always start off the locomotive at its slowest speed. Once it has started moving, it can be put into higher speeds almost immediately without risk of triggering the cut-out.

On the other hand, certain types of motor need an initial burst of power to get them running, after which they can be run at relatively low speed. Slot cars are often like this, requiring a quick "kick" on the control lever to start them. If your model is of this kind, you may find that the best procedure for starting it from rest is to begin with relays 2 and 3 closed, so that maximum power is delivered. Follow this *immediately* with a command to open both relays. The micro is programmed to deliver these two commands in very quick succession, so that the initial high acceleration is over before

you have had time to notice it. The car apparently accelerates smoothy away from its starting point.

With railway systems and models of some other kinds, especially robots, it is possible (though quite a challenge!) to write a program by which you can control the model from the keyboard of the micro by pressing certain keys. While this is happening, the computer is storing in its memory a list of all the commands you have issued and the length of time for which each command was in effect. When the sequence is over, the micro repeats all the commands with the same timing, so the model repeats the whole sequence automatically. This is the way in which an industrial robot is "taught" to perform a complicated sequence of actions by an experienced instructor.

One way in which the Controller may be helped to do its job is to let the micro know what the model has actually done. With a railway system, for example, it is helpful if the micro can be told exactly which part of the track the train has reached, when it is approaching a station, or when the end of the train has moved fully into the siding. One way of locating the train is to arrange beams of light across the track to be broken by the train as it passes. The Lap Sensor described in Project 10 is ideal for this purpose. One or more of these placed at strategic positions on the railway system will make practicable many kinds of automatic manoeuvre. A sophisticated program would allow the operator to type in the names of the departure and destination stations, whereupon the micro would work out the route, set the points and signal lamps and drive the train from the one station to the other, without any further guidance for the operation.

Parts Required for the Model Controller

Resistors

R1–R4	1 k carbon, 0.25 W, 5% tolerance (4 off)
R5, R6	Wire-wound resistors, 2.5 W, of suitable values (*see* p.56)

Capacitors

C1	100 n polyester
C2	4700 μ electrolytic, working voltage greater than that of the model power-supply unit

Semiconductors

D1–D4	1N4001 (4 off)
Q1–Q4	ZTX300 or similar npn transistors

Integrated Circuit

IC1	74LS175 quad D-type flip-flop

Miscellaneous

RLA1–RLA3	Miniature relays; coils rated to voltage of model power-supply unit; single-pole single-throw contacts or change-over contacts; contacts rated to voltage of power supply to 2 A minimum (3 off)
RLA4–	Miniature relay, specification as above, except that contacts are to be double-pole change-over

Circuit board
14-pin IC socket
6-way socket to fit 6-way plug of Decoder board
3-way socket to fit 3-way plug of Decoder board
Plug to fit outlet of model power-supply unit
Connecting wire

PROJECT 5

Bleeper

A device which emits a short "bleep" when triggered by the micro has many uses in connection with games programs and in various applications around the home. Although the BBC Microcomputer has a built-in loudspeaker, and the other two computers make use of the TV loudspeaker, the source of sound is necessarily inside or close to the micro. With this project, the micro can be in one room and the Bleeper in another. For example, with the Bleeper in the kitchen, you could use the micro as an egg timer "bleeping" at the end of the cooking times which each member of the family prefers.

How It Works

The note emitted by the Bleeper is produed by an oscillator built from a single NAND gate (IC2, Fig. 5.1). This gate has Schmitt trigger inputs, which means that the output of the gate swings sharply when the input voltage reaches a certain value. The inputs of the gate are wired together (except for one, which we shall discuss later), so that it acts as an *inverter*. When the inputs are "high" the output is "low", and when the inputs are "low" the output is "high". Suppose the input is low to begin with, and the output is high. Current flows through R2 and gradually charges the capacitor C2. When the voltage across C2 reaches a certain level, this counts as a "high" input. The output swings low immediately. Now current flows from C2 towards the output, which is at 0 V. The capacitor gradually discharges and the voltage across it falls. When it falls below a certain level it counts as a "low" input and so the output swings "high" again. In this way, the output swings "low" and "high" continuously.

The rate at which the circuit oscillates depends on the values of C2 and R2. For the circuit to work, the value of R2

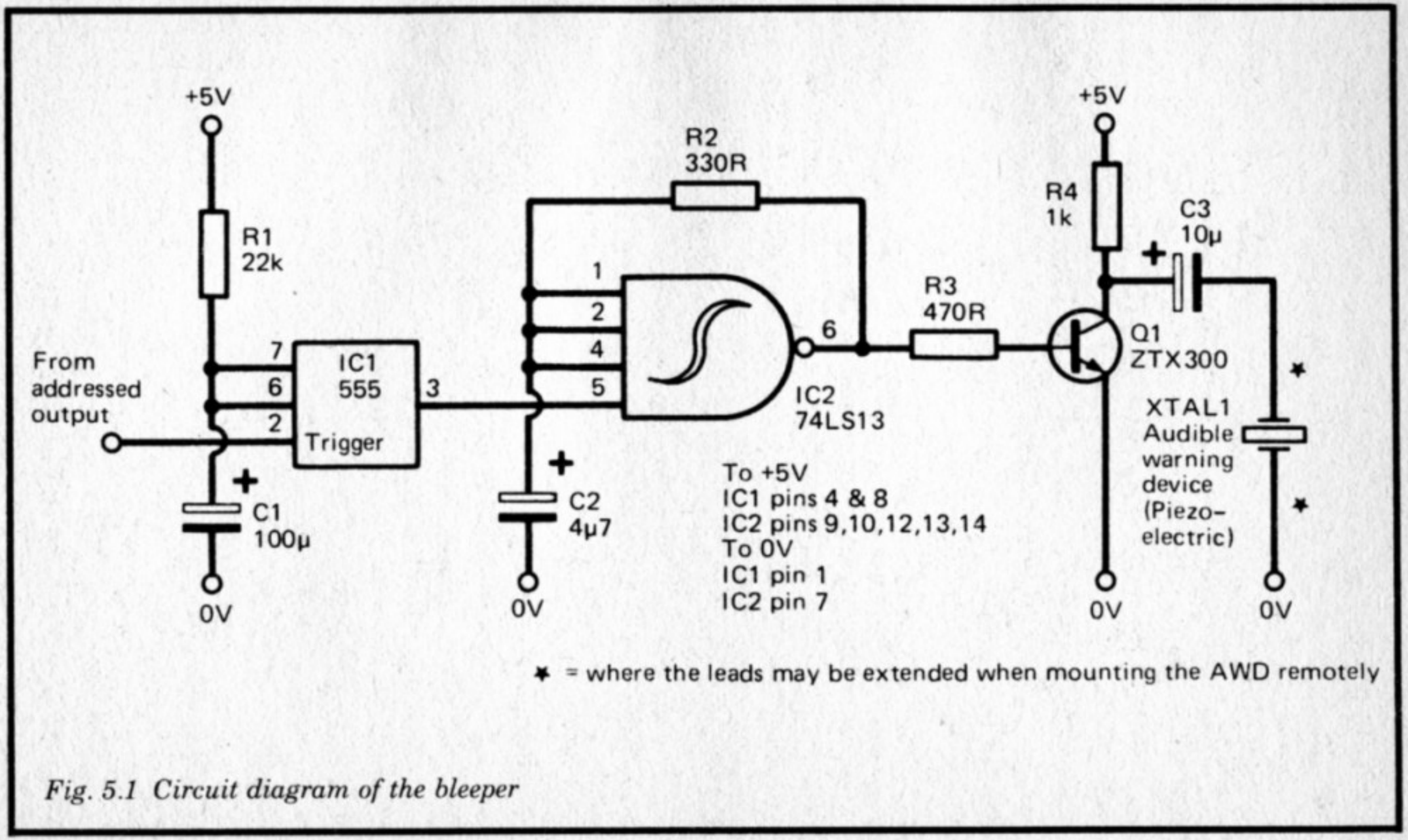


Fig. 5.1 Circuit diagram of the bleeper

must lie between 330 ohms and 470 ohms, but we can alter the value of C2 over a reasonably wide range to give a note of the chosen pitch. With the values shown in Fig. 5.1, the pitch is about 500 Hz.

The duration of the "bleep" is controlled by a 555 timer IC, (IC1). This is wired so as to produce a single "high" pulse (at its output, pin 3), whenever a short "low" pulse is delivered to its trigger input (pin 2). The input pulse may be very short indeed and in this circuit we use the pulse from an addressed output of the Decoder (Fig. D.0). The length of the output pulse can be anything we choose, within reason. Its length depends on the values of C1 and R1. The equation for calculating the duration is:

$$t = 1.1\ RC$$

where t is the time in seconds, R is the resistance of R1 in ohms and C is the capacitance of C1 in farads. With the values given in Fig. 5.1, the duration is about 2.4 seconds.

The output from the timer is normally "low". One input of the NAND gate is held "low", so the gate is prevented from changing state. It cannot oscillate and no sound is heard. When the timer is triggered, its output goes high for 2.4 seconds, during which time the NAND gate is able to oscillate and the "bleep" is heard.

The sound is made by a piezo-electric audible warning device. This is a thin slice of crystalline material which vibrates when a pulsing signal is passed through it. It is rather like a crystal microphone or record-player cartridge, but working in reverse. The output from the oscillator is insufficient to drive the crystal directly, so we use a transistor (Q1) which is switched on and off by the output from the oscillator. This provides enough power to make a suitably loud noise come from the crystal. If you prefer, a small loudspeaker may be wired in place of the crystal.

Building It

The project may be housed in any plastic case big enough to hold the scrap of circuit board on which it is assembled. It

needs only three wires from the Decoder: 0 V, +5 V and one of the addressed outputs. If you intend to use the Bleeper in another room at some distance from the micro, it is best for the main part of the circuit to be in its case close to the micro with the crystal (or loudspeaker) on the end of a long pair of wires leading to the other room.

One thing to think about before beginning construction is the mounting of the audible warning device. The volume of sound obtained is much greater if it is mounted on a firm (but not too firm) surface. The surface acts as a sounding board, helping to transfer the energy from the crystal to the air around. Most crystals are already in a light metal case with metal lugs attached (Fig. 5.2). These lugs are pushed through holes bored in the plastic case containing the circuit board (or a separate case, if the crystal is to be located in another room). Then the lugs may be bent flat to hold the device firmly against the wall of the case. If you are using a loudspeaker instead of a crystal, mount it on the inside of the wall of the case, with a few holes bored in the case to allow the sound to escape. An old transistor radio set could be adapted for this purpose, providing that its loudspeaker is in working order. There is no need to remove the "works" for there is sure to be enough room to spare for the Bleeper board. Disconnect the loudspeaker from the radio circuit and connect it to the Bleeper instead.

Apart from the points mentioned above, building the circuit is very straightforward. The device needs no address decoding other than that already done by the Decoder. Simply run a 3-way cable from one of the addressed output plugs of the board.

The circuit may be tested by connecting it to a +5 V or +6 V supply. Usually it "bleeps" as soon as it is switched on. Temporarily connecting pin 3 of IC1 to the 0 V line will make it "bleep" again. If it does not, check the wiring carefully. If the pitch of the note is not as required, substitute a different capacitor for C2. The larger its value, the lower the pitch. If the length of the "bleep" is not correct, alter C1 or R1. Changing either or both of these to greater values, lengthens the "bleep". It is possible to have the note sound-

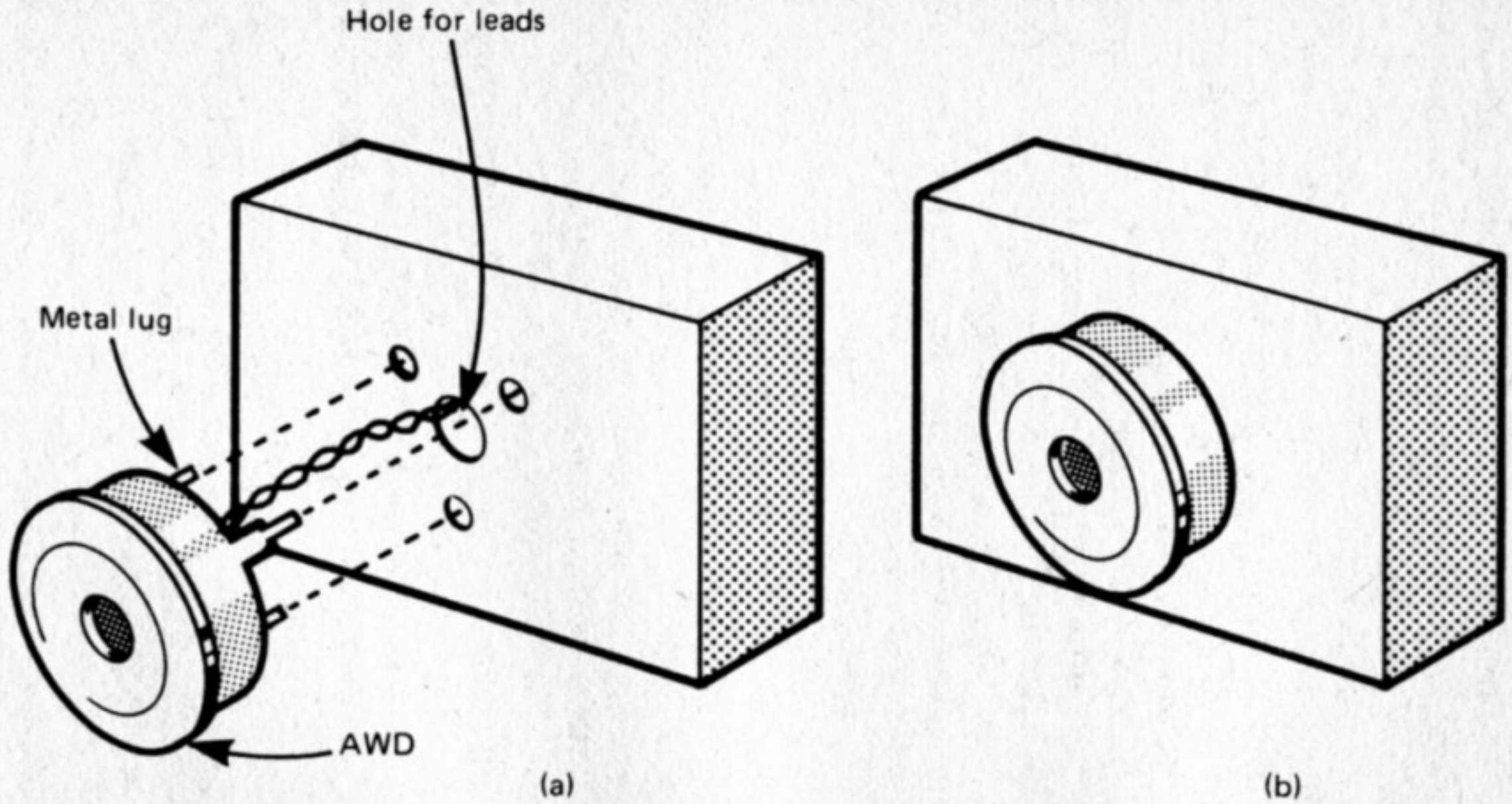


Fig. 5.2 Mounting the audible warning device on the case; (a) showing holes required; (b) lugs inserted in holes and bent around the panel

ing for several tens of minutes.

Programming

This could not be simpler. All you need to do is to trigger the addressed output as described on p. 15.

Parts Required for the Bleeper

Resistors, carbon, 0.25 W, 5% tolerance

R1	22 k
R2	330 R
R3	470 R
R4	1 k

Capacitors, electrolytic

C1	100 μ
C2	4μ7
C3	10 μ

Semiconductor

Q1	ZTX300 or similar npn transistor

Integrated Circuits

IC1	555 timer
IC2	74LS13 dual 4-input Schmitt trigger NAND gate

Miscellaneous

XTAL1	Piezo-electric audible warning device, PCB mounting (loudspeaker, 3 ohm to 15 ohm may be substituted)

Circuit board
8-pin IC socket
14-pin IC socket
1 mm terminal pins (5 off)
3-way socket to fit 3-way plug of Decoder board
connecting wire

PROJECT 6

Lamp Flasher

This circuit flashes one or two lamps (possibly more) on and off, whenever commanded to do so by the micro. It has lots of applications in models, from flashing the "eyes" of a robot, to controlling signal lamps in model-railway systems. It has uses in games and more serious uses around the house for operating warning or other indicating lamps. The lamps used are light-emitting diodes, which are obtainable in several colours – red, orange, yellow or green. There is also the possibility of using infra-red LEDs, whih have applications for remote control of devices not actually wired to the computer. It is also possible to modify the circuit to control small filament lamps.

How It Works

The basic circuit consists of a J-K flip-flop. LEDs are connected to its two outputs, Q and $\overline{Q}$ (Fig. 6.1). These outputs have opposite states, so one LED is on when the other is off. This action can be made good use of for a pair of warning lamps. For a model railway, with red LEDs placed side by side, it imitates the typical pair of flashing lamps found beside the road at the approach to a level crossing. If you want to flash a signal lamp on and off, you need connect only one to the flip-flop.

The J and K inputs are wired to the +5 V line. The effect of this is to make the flip-flop change state whenever a pulse is applied to the "clock" input. This is known as a toggle action. If a given LED is off and it is to be switched on, the micro addresses the flasher. This makes the addressed output of the Decoder go low for an instant, causing the LED to come on. The LED stays on until the next time the Lamp Flasher is addressed, when it goes off. Note that it is when an output from the flip-flop goes *low* that the LED lights.

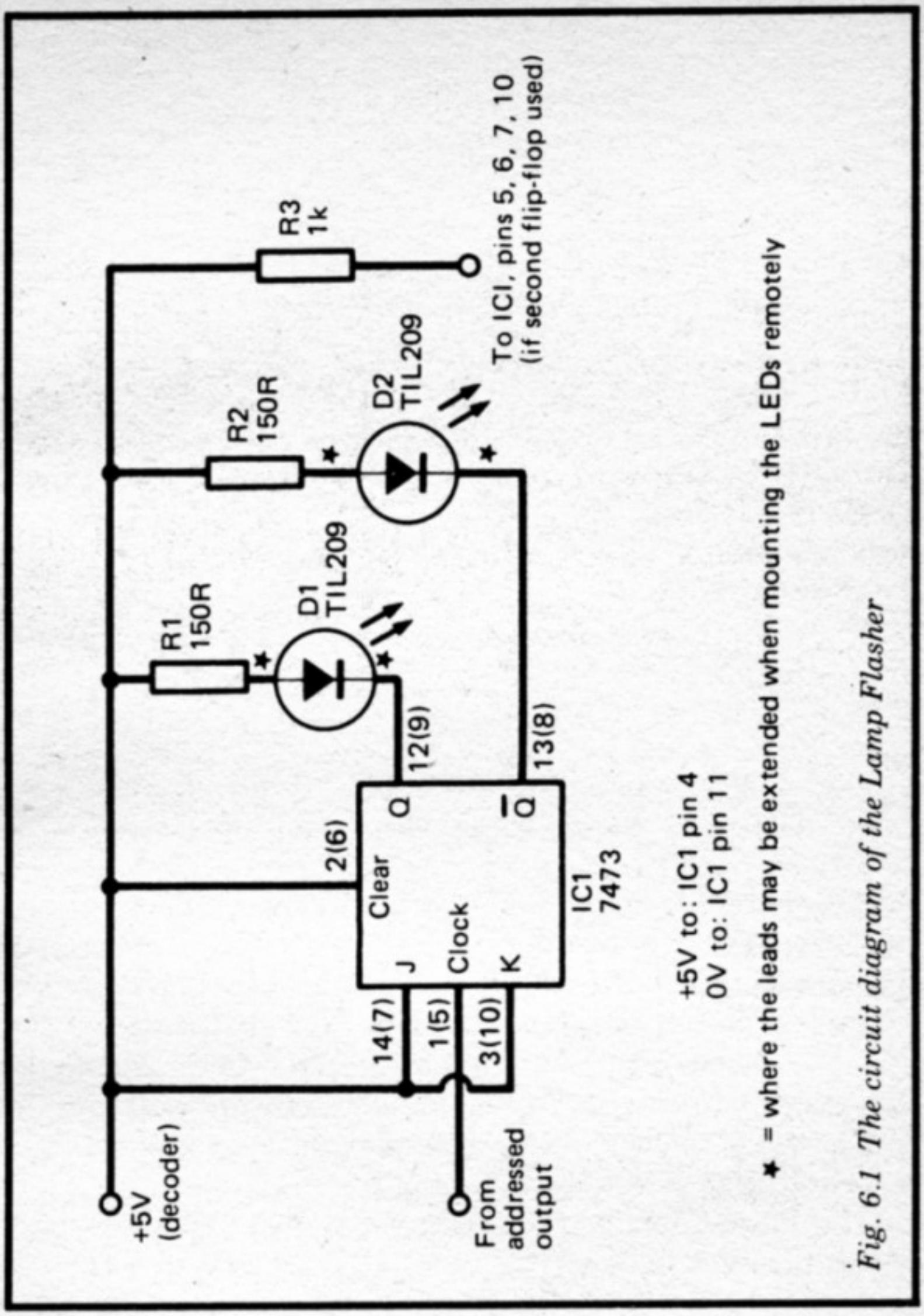


Fig. 6.1 The circuit diagram of the Lamp Flasher

Several modifications of the circuit are possible. The outputs from the flip-flop are insufficient to drive a filament lamp directly, but with the addition of a transistor (Fig. 6.2), it is easy to switch such a lamp. Here, the lamp is on when the output from the flip-flop is *high*. If the lamp requires a

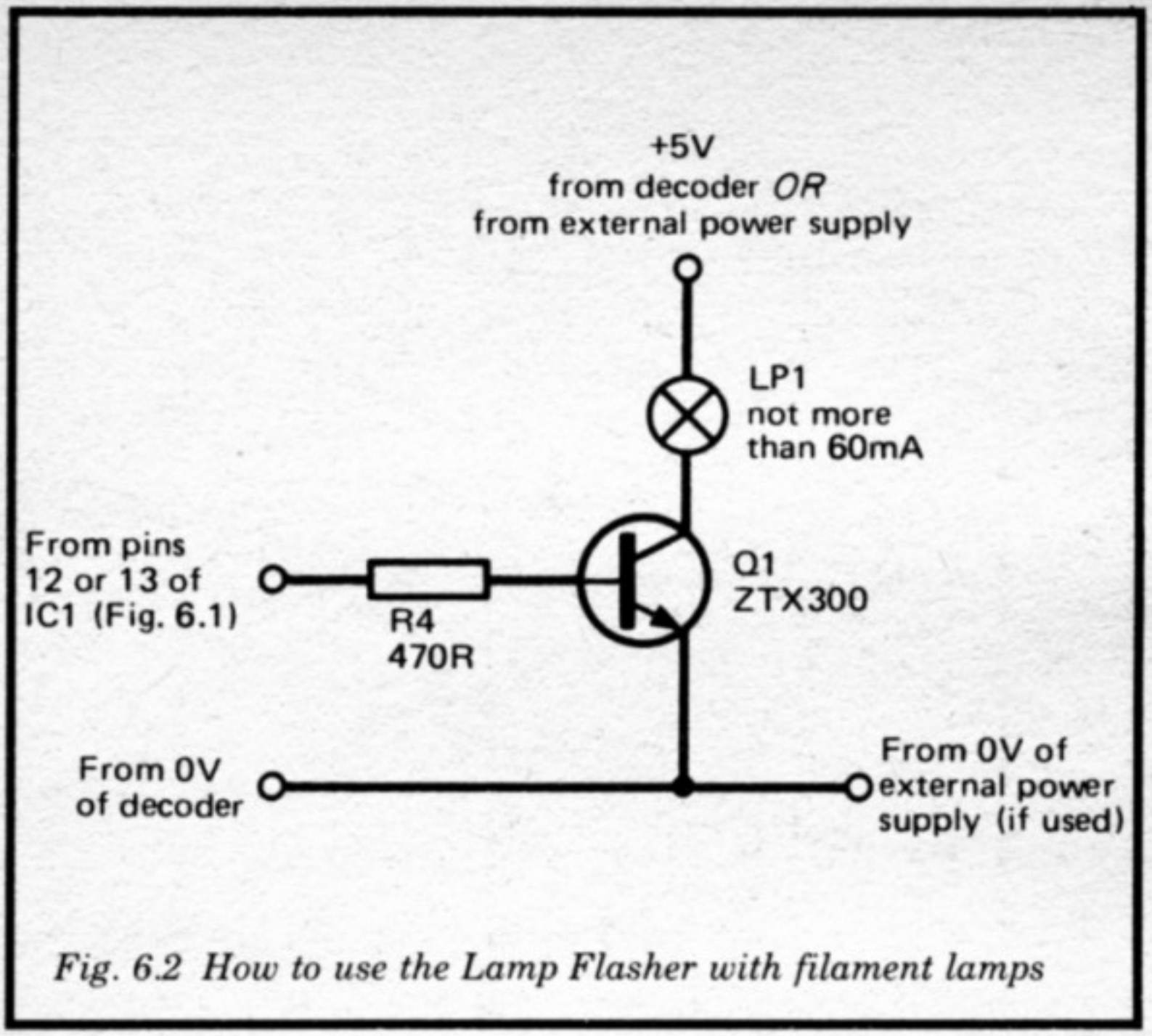


Fig. 6.2 How to use the Lamp Flasher with filament lamps

voltage higher than the 5 V provided by the Decoder, it must be powered from some external source, such as a battery or a power supply. The power supply *must* be direct current. The voltage required depends on the rating of the lamp, but should not exceed 25 V; voltages greater than this would damage the transistor. The gain of the transistor limits the lamp current to about 40 mA, so a 60 mA lamp may not light at full brilliance.

The IC contains two flip-flops, only one of which is used. The pin connections for the second IC are shown in parenthesis in Fig. 6.1. Both "clock" inputs can be connected to the same addressed output allowing two pairs of lamps to be controlled. Alternatively, you can use a separate addressed output for each, controlling the flip-flops independently.

Building It

The circuit takes up so little room that, when used in connection with a model, it may be hidden away inside the model itself. Otherwise, it needs a small plastic box to contain it. Wiring up the circuit presents no problems. The only connections to the Decoder board are the two power lines and the addressed output line. If you intend to place the lamps at some distance from the micro (for example, in another room, or on a mobile model), it is better to have the circuit itself close to the micro and run pairs of wires to the LEDs. The asterisks in Fig. 6.1 show where the leads should be extended.

When you have assembled the circuit, check that there are no short-circuits between the three wires which connect it to the Decoder. Then plug it into the Decoder at one of the addressed output plugs, and switch on the power. One of the LEDs comes on, though it is not possible to say in advance which one this will be.

Test it by triggering an addressed output, as described on pp. 15–18. The lamp changes state each time the output is addressed.

Programming

This program for the VIC-20 makes the lamp flash on and off 10 times:

```
1Ø POKE 37138,24Ø
2Ø FOR J=1 TO 2Ø
3Ø POKE 37136,16
4Ø FOR K=1 TO 1ØØØ:NEXT
5Ø NEXT
```

This program operates by way of addressed output no. 1. Alter the value in line 30, if the device is attached to another output. Pages 15–18 list the corresponding commands for the other two micros.

Of course, there is a lot more to programming than simply making the LEDs flash. Commands are inserted in a wide

variety of programs whenever the flip-flop is to change state. When using the LEDs as indicators, the rate of flashing can be varied according to what information it is intended to convey. An interesting programming project would be to devise a Morse Code sender. This would be a helpful practice device. The user types in a message at the keyboard of the micro. When "Return" is pressed, the micro converts the message into the equivalent symbols of the Morse Code and then flashes a single LED to transmit the message.

Parts Required for the Lamp Flasher

Resistors

R1, R2 150 R (2 off)

Semiconductors

D1, D2 TIL209 or similar light-emitting diodes (any colour, 2 off)

Integrated Circuit

IC1 7473 dual J-K flip-flop, with clear. (Note: this project requires the standard TTL version not Low-power Schottky (LS) version.)

Miscellaneous

Circuit board
14-pin IC socket
3-way socket to fit 3-way plug on Decoder
1 mm terminal pins (3 off; 7 off if LEDs are mounted remotely)

PROJECT 7

Light Pen

You may already have seen a Light Pen being used with a computer and know some of the ways in which it can be used. Often it is used to "draw" on the screen. The user moves the pen across the screen, just as if drawing, and a line appears on the screen, following the tip of the pen.

There are several other ways in which the Light Pen is used. For example, it is used in selecting choices from a "menu" displayed on the screen.

You point the pen at the item of your choice and the computer then does as requested. The same idea can be applied to learning programs that have multiple-choice questions. You point the pen at the answer which you think is correct. Many people are unfamiliar with the layout of a typewriter keyboard and perhaps slightly apprehensive about trying to use one. For such people it is easier to point with a Light Pen than to look at a keyboard and try to find and press one of several possible keys.

There are applications for the Light Pen in games too. Instead of making your moves by using the keyboard or a joystick, use the Light Pen.

How It Works

A Light Pen works by sensing light coming from the screen. In this design, the sensor is a phototransistor.

The potential at A is compared by an operational amplifier (IC1) with a reference potential which is set by adjusting the variable resistor RV1. The amplifier is very sensitive to small differences of potential. If the potential at A is a little higher than that at the wiper of RV1, the output of the amplifier falls very sharply, almost to 0 V. On the other hand, if the potential at A falls slightly so that it becomes just a little lower than that at the wiper of RV1, the output voltage

swings sharply upwards, almost to +5 V. In this way, the circuit gives a low output in darkness or dim light, and high output in brighter light. There is a rapid swing at a fixed level of brightness. We can choose this *threshold* level by the position on which RV1 is set.

The output from the amplifier is a clear-cut signal which is entirely suitable for feeding to a TTL logic gate. The output is taken directly to one of the data input buffers on the Decoder (Fig. D.0).

Building It

The circuit is connected to the Decoder board by only three wires: the two power lines, and a wire to one of the data

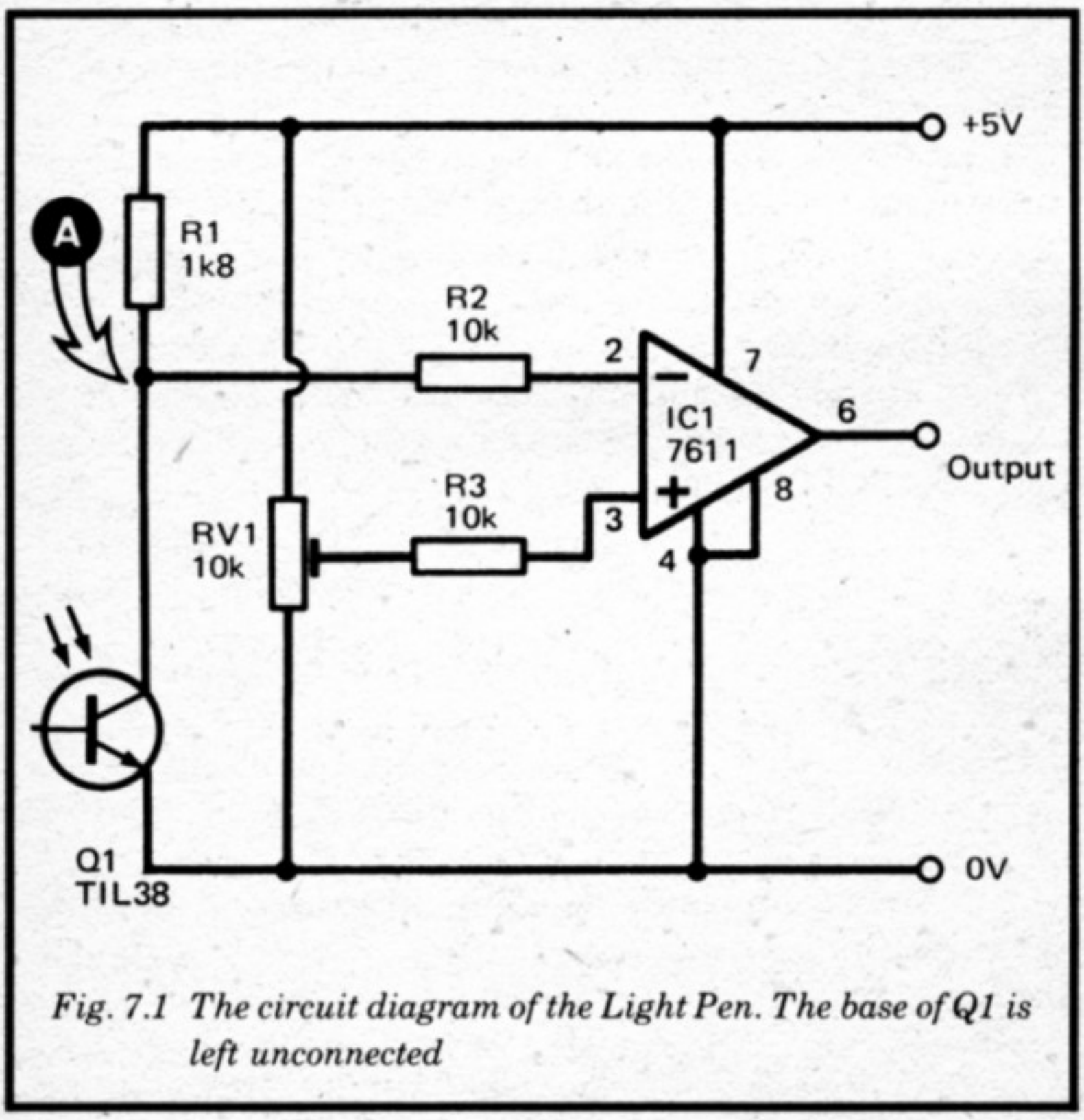


Fig. 7.1 The circuit diagram of the Light Pen. The base of Q1 is left unconnected

input plugs. There are two wires going to the phototransistor.

All of the circuit except for Q1 is mounted on a small piece of stripboard, enclosed in a case. The potentiometer is of the preset type. It is worthwhile to make a small hole in the case so that a screwdriver may be inserted to adjust the potentiometer occasionally.

The pen itself consists of the phototransistor mounted inside a tube of light-proof material (Fig. 7.2). The barrel of an old felt-tip pen can usually be adapted for this. The prototype used the barrel of a "painting stick" felt-tip pen. The internal diameter of the barrel was about 7 mm, narrowing to about 4 mm at the tip. The phototransistor was wedged inside the barrel, after first wrapping plentiful amounts of insulating tape around its leads. Its lens was 30 mm from the open tip of the pen. Holes were bored in the plug at the rear of the pen to take the twin leads. The leads were knotted as shown in Fig. 7.2 to prevent an unintentional pull on the leads from dislodging the phototransistor.

The barrel of the pen was made of black plastic, with the advantage that internal reflection was reduced. This made the pen more directional. If you are using a barrel made of material of another colour, paint the inside of the tube a dull black, or line it with matt-black paper.

If you are using a metal tube, take care to cover its end with a softer or more flexible material so that there is no danger of scratching the front surface of the TV tube.

The lead between the pen and the main circuit should be about 1 metre long to allow freedom of movement of the pen.

When used with the VIC-20, the device is connected to the games port (controller port), using a 9-way "D"-type socket. The +5 V and 0 V leads are soldered to pins 7 and 8 as in Fig. 3.3, while the output lead goes to pin 6. The Light Pen may then be used entirely independently of the Decoder and the devices attached to it. The video interface chip of the VIC-20 keeps track of the position of the electron beam as it scans the face of the TV tube. At the instant the pen detects the beam and sends a low pulse to pin 6 of the games con-

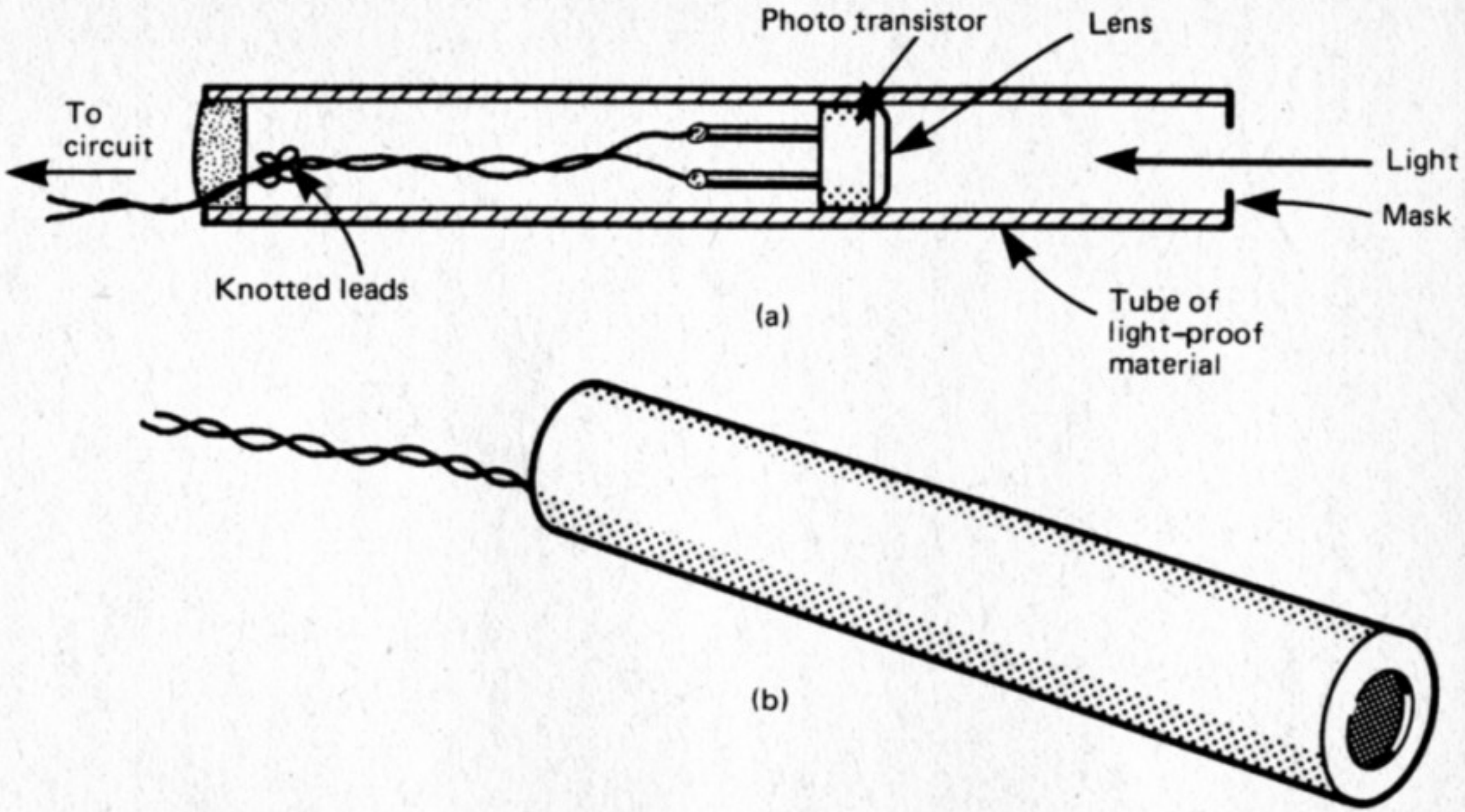


Fig. 7.2 A simple version of the Light Pen, (a) in section, (b) external view

nector, the video interface chip puts two numbers into its two Light Pen registers to record the position of the electron beam. One register (address 36870) holds the X-position (left-right) and the other (address 36871) holds the Y-position (up-down). These values are held latched until they are read. The registers retain these values until the next time a low pulse is detected from the pen. The values in the registers thus indicate the location on the screen to which the pen is currently being pointed.

This is the test program for the VIC-20:

```
10 PRINT "■"
20 PRINTPEEK(36870);PEEK(36871)
30 FOR J=1 TO 100:NEXT
40 GOTO 10
```

The program repeatedly reads the Light Pen registers and displays their contents on the screen. The delay loop at line 30 allows you just long enough to read these before the screen is cleared and a new value is displayed. Plug the Light Pen into the games port and then RUN the program. Point the Light Pen at the screen. The tip of the pen should be within 5 cm of the screen, but there is no need for the pen to touch the screen. You will see a pair of numbers flashing at the top-left corner of the screen. Wave the pen about pointing first at the top of the screen, then at the bottom. The second number (Y-position) should change in value each time it is displayed, ranging from about 40 when the pen is pointing at the top of the screen (excluding the coloured border), to about 125 when pointing at the bottom. If the number does not change, the sensitivity of the pen needs adjusting; vary the setting of RV1 until the number displayed changes as described above.

Move the pen from left to right across the screen. This time the first pair of figures (X-position) should change. The range is from about 60 at the left to about 140 at the right. X-direction scanning is much faster than Y-direction scanning, because the beam has to scan across the tube 50 times while scanning only twice from top to bottom. Consequently, delays in the system (both in the hardware and the software)

are likely to be more serious when measuring the X-position. The result is that when the pen is pointed to the extreme right of the screen, the beam has been flicked back to the left side before its position is registered. This is why you may obtain low values (less than 20) for the X-position when the pen is pointed to the extreme right). There are two solutions to this problem:

(i) Do not use the extreme right of the screen.
(ii) Adapt the BASIC program to add a constant value to all values less than, say, 50.

For many applications, such as selection from menus or indicating the answers to a set of questions, only the Y-position need be read, so this complication does not arise.

The Light Pen may be attached to the Commodore 64 and used in the same way. The 9-way "D" socket is wired as for the VIC-20, but it may only be used with port 1. Test it as described above, but reading X-position from address 53267 and Y-position from 53268.

The BBC Microcomputer Model B accepts a Light Pen input at its analogue input port. This takes a 15-way "D"-type plug, the connections to the Light Pen being wired as in Fig. 7.4. The procedure for testing the pen is a little more complicated than for the other two computers. The 6845 CRT Controller chip of the BBC micro stores the position of the beam as a single value in two registers, R16 (hex $10) and R17 (hex &10). The contents of R16 multiplied by 256 and added to the contents of R17 give the position of the scan. In order to read from one of these registers, we first have to put the number of the register in the address register of the chip. This register resides at address &FE00. As soon as this has been done, the value required may be read from the register file of the chip at address &FE01. Both of these addresses lie in the SHEILA range, so are best accessed by using an OS-BYTE call. Here is a procedure for performing the operation described above:

```
5ØØ DEF PROCpen
51Ø A%=&97:X%=Ø:Y%=&1Ø
```

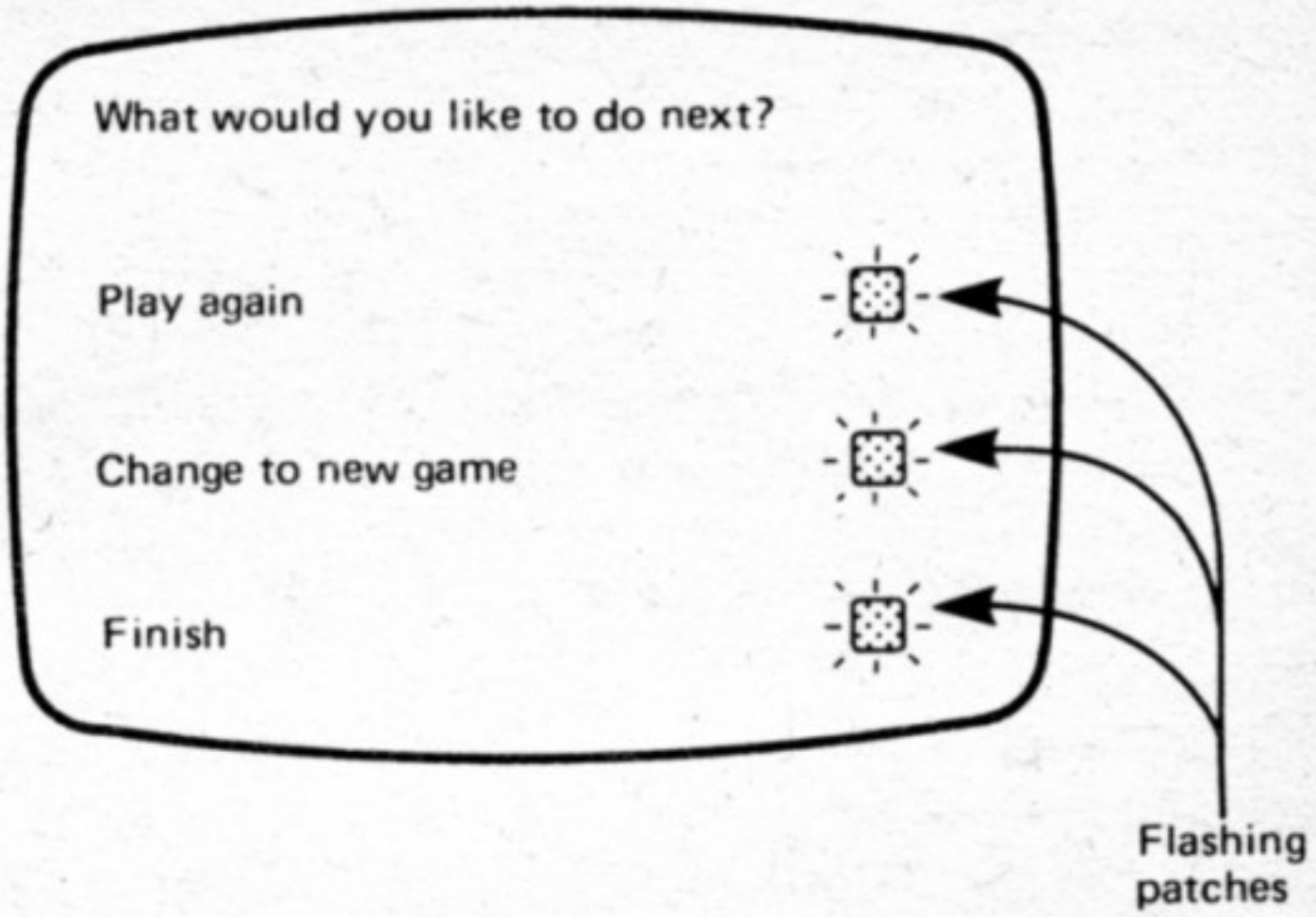


Fig. 7.3 The menu displayed with flashing patches for use with the Light Pen (BBC Microcomputer, Model A)

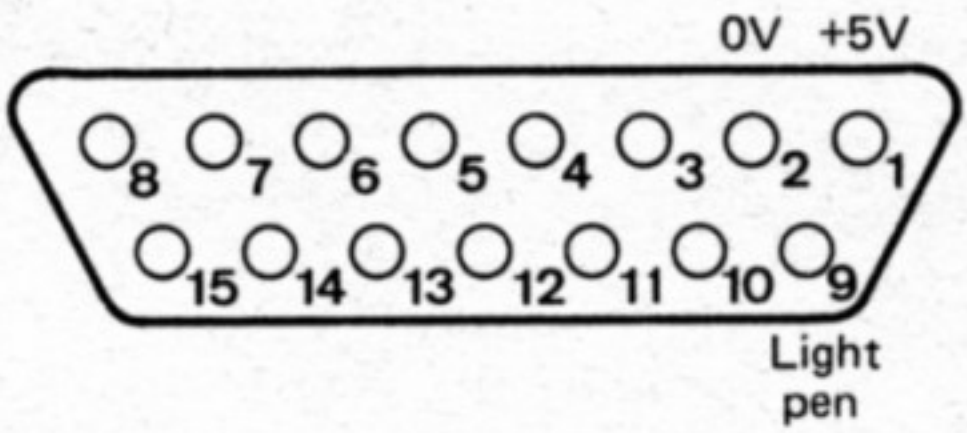


Fig. 7.4 Connections to be made to the BBC Microcomputer analogue input port

```
520 CALL&FFF4
530 A%=&96:X%=1
540 X=(USR(&FFF4)AND&FF0000)/&100
550 A%=&97:X%=0:Y%=&11
560 CALL&FFF4
570 A%=&96:X%=1
580 X=X+(USR(&FFF4)AND&FF0000)/&100000
590 ENDPROC
```

Multiplication by 256 is taken care of by dividing by &100 in line 540 instead of dividing by &10000.

This procedure may now be used in a testing program which is essentially the same as those given above:

```
10 CLS
20 PROCpen
30 PRINT X
40 FOR J= TO 200:NEXT
50 GOTO 10
```

If you have the Model A machine with the user port upgrade but not the analogue port upgrade, you can still use the Light Pen. It is plugged into a data input of the Decoder and its state is read through port B, using the program lines or procedure given on pp. 17–18. The Model A has no provision for finding out the position of the electron beam by reading registers. You will have to do this by software instead. The general principle is to decide on certain locations on the screen to which the pen may be pointed. You write the program so that each patch of the screen flashes alternately white and black (Fig. 7.3). After each command to flash white, use a procedure to read the Light Pen. If the Light Pen shows a low output, it must be pointing at that particular patch of screen. You now know which menu item is being selected. The patches are flashed in order, one after another and the output of the Light Pen is read after each patch has been flashed white (and before it is made black again).

For Acorn Electron *see* note on page 187.

Programming

With the VIC-20 and Commodore 64, the chief problem in programming is that the X-position reading fluctuates appreciably, even while the pen is being held still. Programs which involve choices of menu or selecting answers to questions need refer only to the Y-position which remains reasonably constant when the pen is held still.

The program segment below shows a typical menu, the items being displayed on screen lines 3, 7, 11 and 15. Pen readings previously taken on these lines have given readings centred on values of 52, 66, 83 and 96. The program allows for readings one less than or one greater than these values.

```
200 Y=PEEK(36871)
210 PRINT"■";"POINT TO ITEM REQUIRED":
220 PRINT"Q":PRINTTAB(5)"1) READ FILE"
230 PRINT"QQ":PRINTTAB(5)"2) CREATE FILE"
240 PRINT"QQ":PRINTTAB(5)"3) DELETE FILE"
250 PRINT"QQ":PRINTTAB(5)"4) SAVE FILE"
260 IF ABS(PEEK(36871)-Y)<5 THEN 260
270 Y=PEEK(36871)
280 IF Y>50 AND Y<54 THEN 1000
290 IF Y>64 AND Y<68 THEN 2000
300 IF Y>81 AND Y<85 THEN 3000
310 IF Y>94 AND Y<98 THEN 3000
320 GOTO 270
```

The four routines called by this menu begin at lines 1000, 2000, 3000 and 4000 respectively. The first PEEK establishes the value in the Y register when the pen is *not* pointing at the screen. This could be the last value read during a previous use of the pen. After displaying the menu, the program waits in line 260 until there is a significant change in Y. This is taken to indicate that the pen has now been pointed at some part of the screen. Lines 260–320 repeately read Y until it is found to lie close to one of the values which correspond to an option on the menu. The program then branches to the appropriate routine. One or more of these routines may begin with a further menu selected by the

Light Pen; for example, a menu could be used to determine which file is to be erased.

If many options are available, it is better to divide them into groups with similar features. The first menu allows a choice between these groups, then the menu for the chosen group is displayed, allowing the final choice of an option. Used in this way, the Light Pen makes it easy for the inexperienced user to get to the required part of the program. It also has applications for disabled persons. There are many people who are unable to operate a keyboard but who can more easily direct a Light Pen. A program in which the user points to displayed letters of the alphabet or numerals, as well as the names of certain computer keys (e.g., "Return", "Shift") would allow a disabled person to write programs on the computer, to use it for word processing, or simply to communicate with other people, either directly or by way of the telephone (using a modem).

The program segment outlined above can readily be adapted for the Commodore 64 and the BBC Microcomputer. With the latter, it is necessary to allow for a greater range of values for a given screen line.

Light Pens are frequently used with games including a number of "drawing" and "painting" programs. With these, knowing the X-position is essential. As has already been mentioned, consecutive readings at the same pen position are liable to vary appreciably. The effects of such variations may be "ironed out" by repeated sampling and by combining *part* of the current reading with the average of previous readings. This is what is known as a "digital-filter" technique. A description of the theory and method is given in *Real Time Programming – Neglected Topics* by Caxton C. Foster (Addison-Wesley). The program segment below shows how this may be done on the VIC-20.

```
10 XX=PEEK(36870)
20 DX=XX-X
30 X=X+INT(DX/5)
40 PRINT X
50 GOTO10
```

Here the program is given as a simple loop, though it is more likely to be used as a subroutine which is called repeatedly. It can be called repeatedly from the same part of the program (as when waiting for input) or from different parts of the same program. Line 10 reads the X-position register, Line 20 finds DX, the same amount by which X has changed since the previous pass through the routine. A fraction of this difference is then added to the previous value of X before the new value of X is displayed. Thus, any change in X is reduced to one-fifth of its magnitude before being merged with the previous reading. Since an erratic change in one direction is likely to be followed by an erratic change in the opposite direction, such changes tend to cancel out, giving a steady reading. However, as you move the pen across the screen, there is cumulative effect. The value of X increases or decreases steadily. The effect of the routine may be adjusted by altering the value of the divisor in line 30. The greater its value, the more the fluctuations in X are filtered out and the steadier the reading becomes. The drawback of this is that the program cannot cope with rapid movement of the pen across the screen. Displayed values of X lag behind the actual position of the pen. It is necessary to choose a value which compromises between steadiness and speed of action. You need to try the program segment in any given application to find the most suitable value for the divisor.

When using the X-position register for the VIC-20 and Commodore 64, and for all readings taken with the BBC Microcomputer, it is usual to correct the reading by subtracting a constant value from it to allow for time delays. Allocate a variable in your program for this purpose. The exact value of this can then be altered until the program gives the intended result. Having done this and, if necessary, filtered out rapid fluctuations, the result can be scaled to give screen co-ordinates. These can be in terms of lines and columns, of "TAB" values, or of the screen co-ordinates used in graphics displays.

Parts Required for the Light Pen

Resistors (carbon, 0.25 W, 5% tolerance)

R1	1k8 (depending on average resistance of R4)
R2, R3	10 k (2 off)
RV1	10 k variable preset resistor

Semiconductor

TIL38 phototransistor

Integrated Circuit

IC1	7611 CMOS operational amplifier

Miscellaneous

Circuit board
8-pin IC socket
"D" connector (9-way *socket* for VIC-20 or Commodore 64; 15-way *plug* for BBC Model B; or 3-way socket to fit 3-way pin on Decoder for BBC Model A)
1 mm terminal pins (5 off)
Materials for making the body of the pen
Connecting wire, including light flexible wire for the lead to the pen

PROJECT 8

Magnetic Catch

This circuit has a once-for-all action. It is reset manually and remains reset until triggered by the micro. Triggering leads to some kind of mechanical action. After this, it must be reset manually again. Exactly what action occurs when the catch is triggered depends on the application to which it is put. It could be triggered to release the bolt on a door, so, for example, letting the cat out early in the morning. Or maybe the bolt is on the door of a cupboard which is to be opened only when the correct secret password is typed into the computer. It could be used to discharge a quantity of food into an aquarium, so overcoming the problem of feeding the fish on Saturday evening when you are away for the weekend. If you are away for the day, and if the thermometer of Project 15 is used to monitor the temperature in your greenhouse, the magnetic catch can be triggered by the micro when the temperature gets too high. It opens the window or door or lets the sun blinds roll down over the roof.

How It Works

The action of the catch depends upon briefly energising a coil or solenoid. There is a sliding soft-iron core partly inside the solenoid. When a burst of current flows, the magnetic field so created pulls the core swiftly into the solenoid. This motion is used to perform whatever mechanical action is required. One of the problems with solenoids is that the more powerful ones have coils with a relatively low resistance. They would draw too much current from the micro. If a battery supply was provided, the batteries would soon be exhausted if the coil were to remain energised for periods of a few hours. This circuit uses capacitors to store the energy needed to activate the solenoid (Fig. 8.1). When the circuit is being reset, button S1 is pressed and charges the capacitor up to maximum

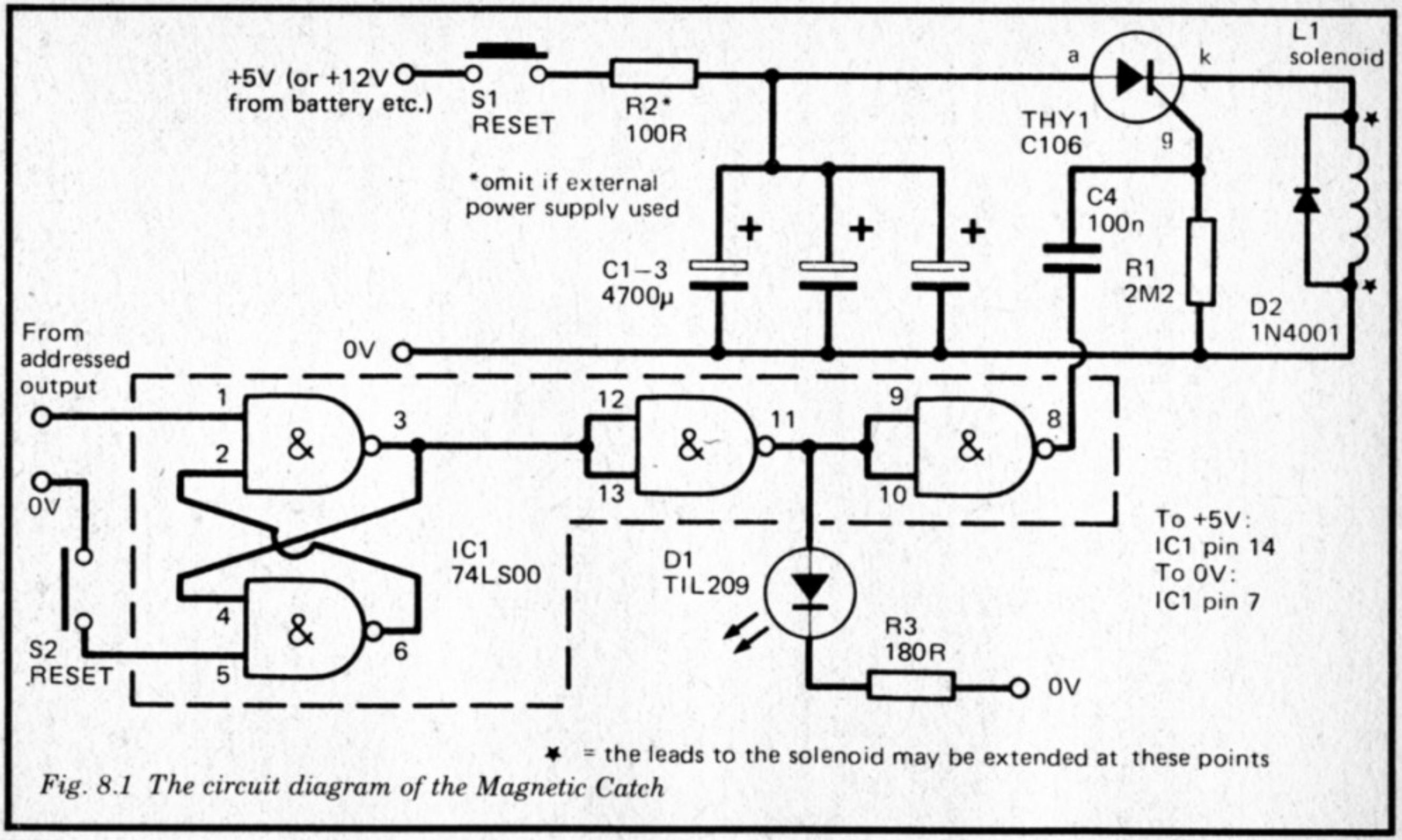


Fig. 8.1 The circuit diagram of the Magnetic Catch

voltage. If the supply is taken from the micro or Decoder this is +5 V. If the solenoid requires a higher voltage for satisfactory operation, it is charged from a battery or other external supply.

The capacitors are prevented from discharging through the solenoid immediately because there is a thyristor in the circuit. This does not conduct until a high pulse is sent to its gate electrode (g). Otherwise the capacitors remain charged for many hours, and during this time the only power required to operate the catch is that needed by IC1 (about 8 mA).

The gate electrode is held at low voltage by resistor R1. The gate is also connected, through a coupling capacitor C4 and two inverting gates, to the output of a flip-flop. When this flip-flop is reset by pressing S2, its output goes low. The inverted output goes "high", putting the LED on to indicate that the circuit is now reset. The doubly inverted output goes "low".

The flip-flop is triggered by a low-going pulse from an addressed output of the Decoder (p. 166). When the micro addresses the catch, the flip-flop changes state. The LED is turned off and a high pulse passes through C4 to the gate of the thyristor. This causes the thyristor to begin conducting. It conducts very easily and the capacitors are discharged almost instantaneously through the solenoid, This creates the strong magnetic field required to operate the catch. Once the capacitors have been discharged, no further action occurs until the flip-flop has been reset and the capacitors have been recharged.

If the capacitors are to be charged from the +5 V supply of the micro, it is essential to include a resistor (R2) in the charging circuit. Otherwise, the sudden drain on the power supply of the micro is too great and its program is lost. The resistor reduces the charging current to a manageable amount, though it then takes about 10 seconds to recharge the capacitors.

The mechanical side of the project is left to the ingenuity of the reader, for so much depends upon the application for which it is intended, the strength of the particular solenoid

used, and the skill of the reader in constructing mechanical devices. Fig. 8.2 shows how a pull from the core can be made to release a prop, so causing a window (e.g. of a greenhouse) to fall shut under the action of gravity. Fig. 8.3 shows how a platform carrying the required daily quantity of fish food may be released, so scattering the food into the aquarium below.

As an alternative to a solenoid with a sliding core, the reader may prefer to use a solenoid with a fixed core which attracts a moving armature. An old electric bell or buzzer could be adapted for this purpose. Fig. 8.4 shows how the motion of an armature can release a spring lever, so initiating a relatively more forceful operation.

Building It

The catch requires three wires to connect it to the Decoder: the two power lines and the wire connecting it to an addressed output. The circuit should be connected to the micro by fairly short wires. If the solenoid is to be located at some distance from the micro its leads may be extended at the points marked * in Fig. 8.1. The circuit is best housed in a case, on which are mounted the two push buttons and the LED. If you are using a battery for charging, the case needs to be large enough to hold this too.

It is best to design, build and test the mechnical side of the catch before constructing the electronic side. It is essential to make certain that the solenoid you intend to use will develop enough power to actuate the mechanism. You can connect the capacitors together on a breadboard or by leads ending in crocodile clips. Then charge them from a battery and try discharging them through the solenoid. If the force developed is insufficient, there are several ways of increasing it:

(i) Begin with the moving core further inside the coil (though this also reduces its length of travel).
(ii) Use a solenoid with more turns of wire.
(iii) Add more capacitors in series with C1–C3.

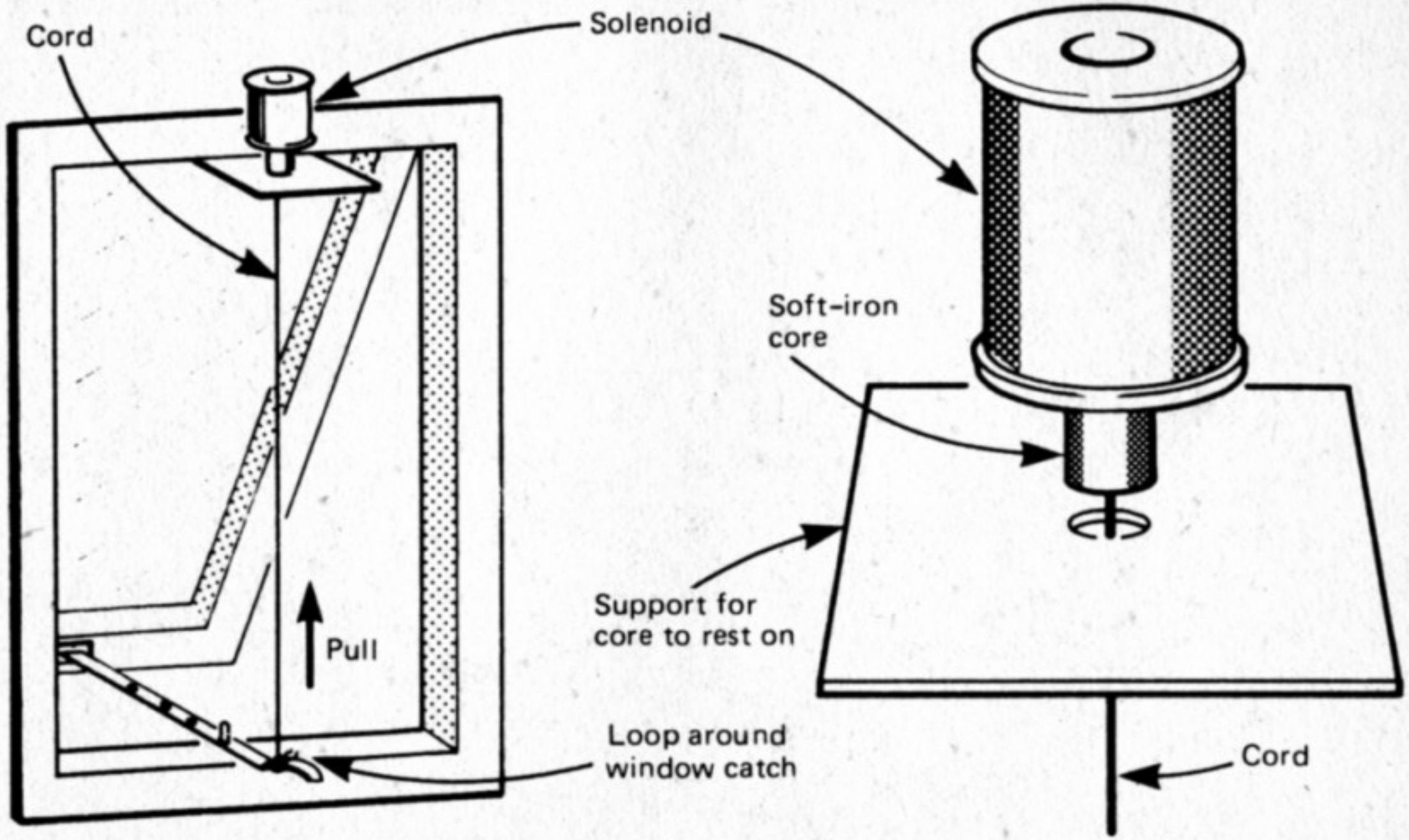


Fig. 8.2 A simple mechanism to close a window

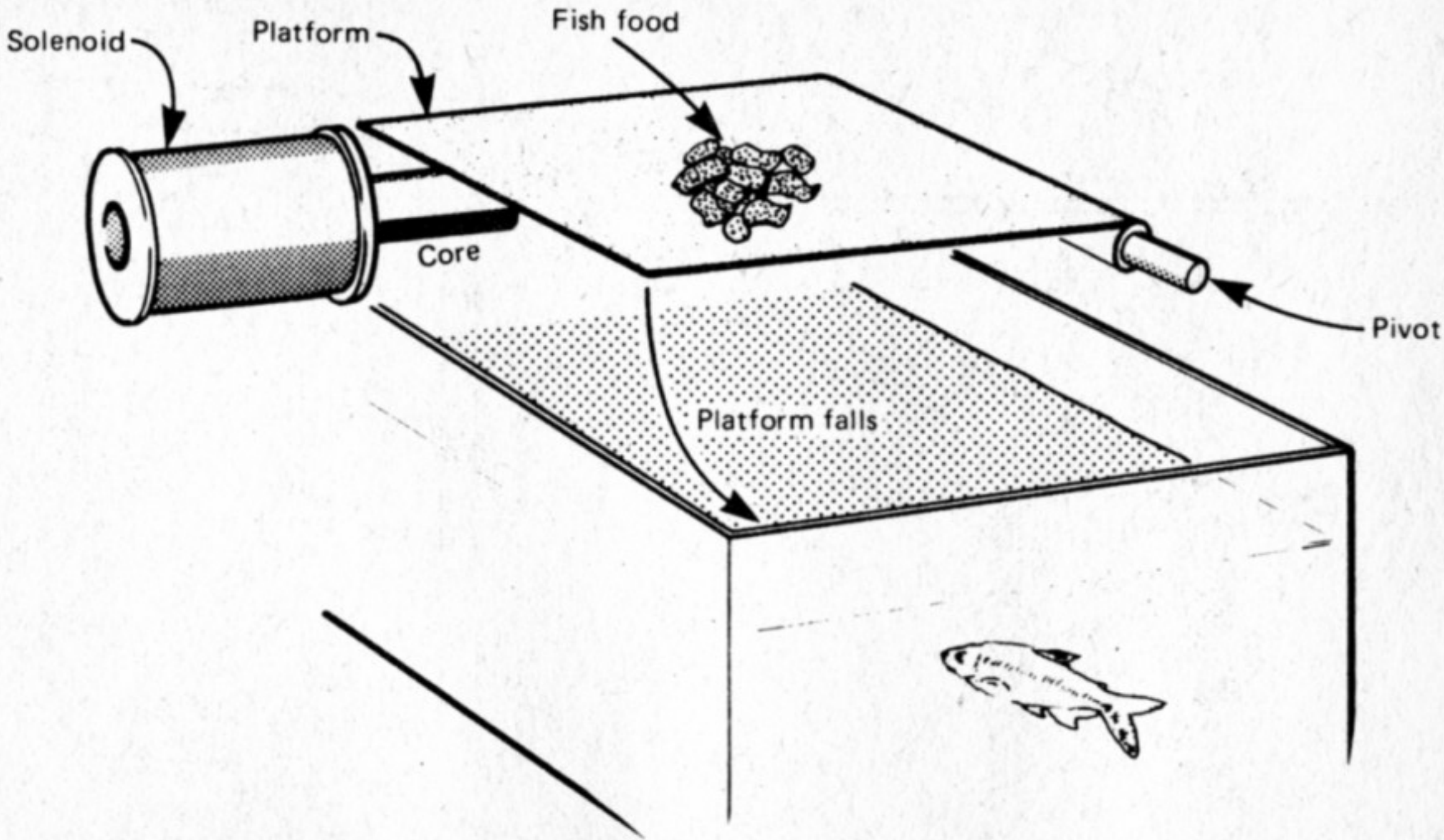


Fig. 8.3 A mechanism for feeding aquarium fish

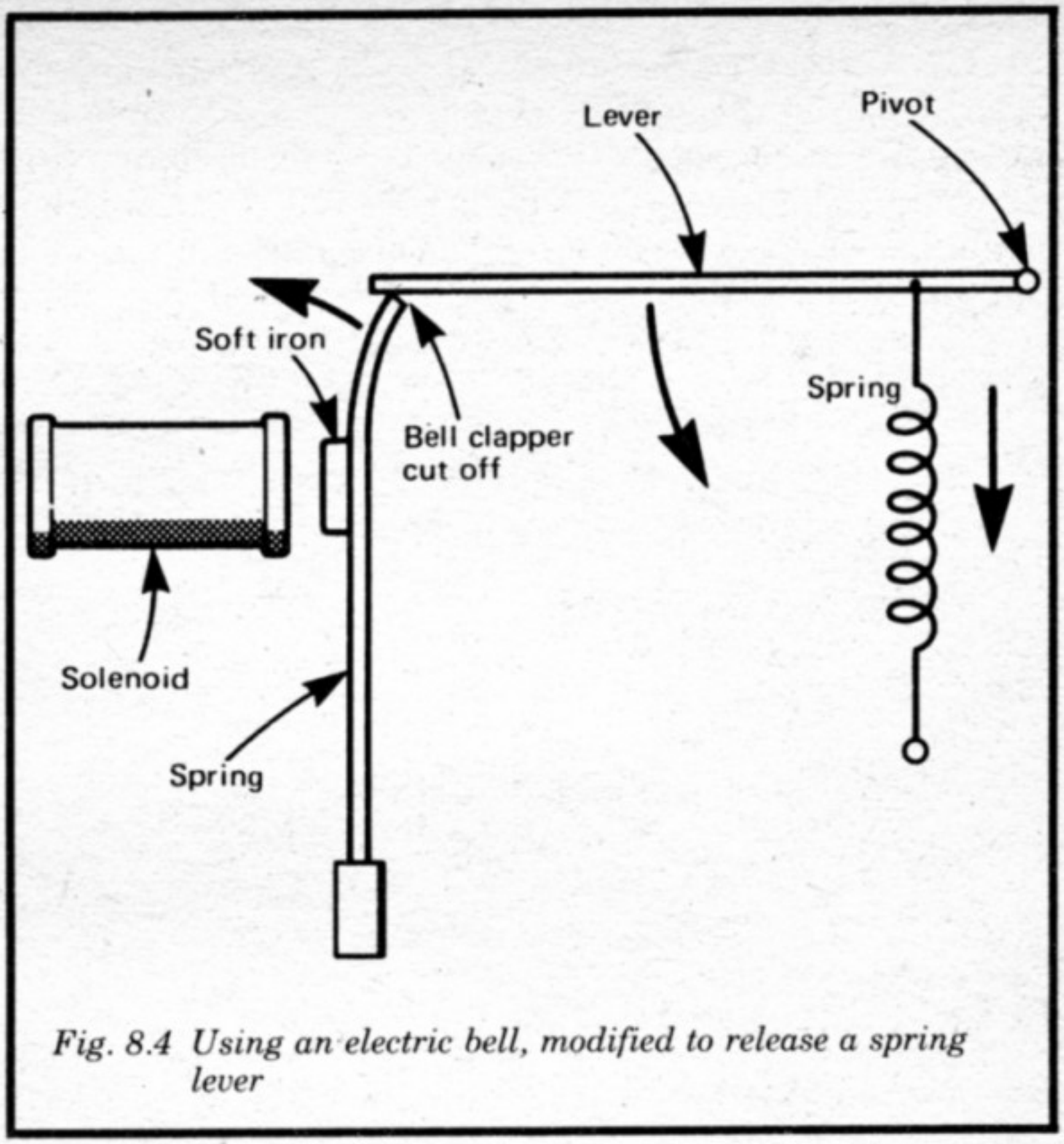


Fig. 8.4 Using an electric bell, modified to release a spring lever

(iv) Charge the capacitors to a higher voltage (assuming this is within the rating of the solenoid).

It is also worthwhile checking that the mechanical design is sound. For example, parts that are supposed to slide should do so smoothly without undue friction. A transverse force applied to the core by the weight of an attached lever may prevent it from sliding easily into the solenoid.

Once assembled, the electronics are tested for short circuits between the lines which connect it to the micro. It is then ready for testing on the micro. Plug it on to the Decoder board and trigger its addressed output as described on pp. 15–18.

Press S2 to reset the flip-flop; the LED comes on. Next press S1 to recharge the capacitors. Hold the button pressed down for about 10 seconds if you are charging from the micro's own power supply. Reset the mechanism so that it is ready to be triggered. When all is reset, RUN the program again. The solenoid is energised, the mechanism is operated and the LED goes out.

Programming

The essential program lines are those on pp. 15–18. Apart from these, the rest depend on the application.

Parts Required for the Magnetic Catch

Resistors (carbon 0.25 W, 20% tolerance)

R1	2M2
R2	100 R (required if micro power supply used for charging)
R3	180 R

Capacitors

C1–C3	4700 μ electrolytic (3 off)
C4	100 nF polyester

Semiconductors

D1	TIL209 or similar light-emitting diode
D2	1N4001
THY1	C106 or similar thyristor

Integrated Circuit

IC1	74LS00 quadruple 2-input NAND gate

Miscellaneous

L1	Solenoid with soft-iron moving core, rated to operate at 6 V, 12 V etc. (alternatively, an old electric bell, buzzer or relay)
S1, S2	Push-to-make push-button switches (2 off)

Circuit board
14-pin IC socket
3-way socket to fit 3-way plug of Decoder
1 mm terminal pins (9 off)
Parts for making mechanism
Connecting wire

PROJECT 9

Lap Sensor

Although this project is called "Lap Sensor", it has many other applications. In its original application as a lap sensor for slot-car racing, it detects when a car passes a given point on the track. Depending on how the micro is programmed, it can keep count of the number of laps, it can measure lap time, or it can do both things at once. From this information it can calculate and display the lap speed and the race speed. Since the device works by the breaking of a beam of light passing across the track, it does not matter what it is that breaks the beam. It could equally well be used for bicycle races, horse races or in many kinds of athletic track events.

Likewise, it does not have to count how many times the same object breaks the light beam, but could be counting how many times different objects break the beam. It can count people going into a room, cars going into a car park, or objects on a conveyor belt. It works so fast that it can count objects (or other causes of interruption of the beam) which come so quickly in succession that it would be difficult, if not impossible, to count them by eye.

Another class of application is that in which we want the computer to wait for and act on a single interruption of the beam. An obvious example is in connection with intruder detection. A beam of light across a corridor is broken by the intruder. The micro detects this and sounds the alarm. It could even be programmed to detect the setting of the Sun and then switch on the porch light!

How It Works

The light is detected by a photodiode (D1, Fig. 9.1). This receives light from a source some distance away. A special lamp may be provided to focus a well-defined beam across the track or corridor, but this is not essential. If a table lamp

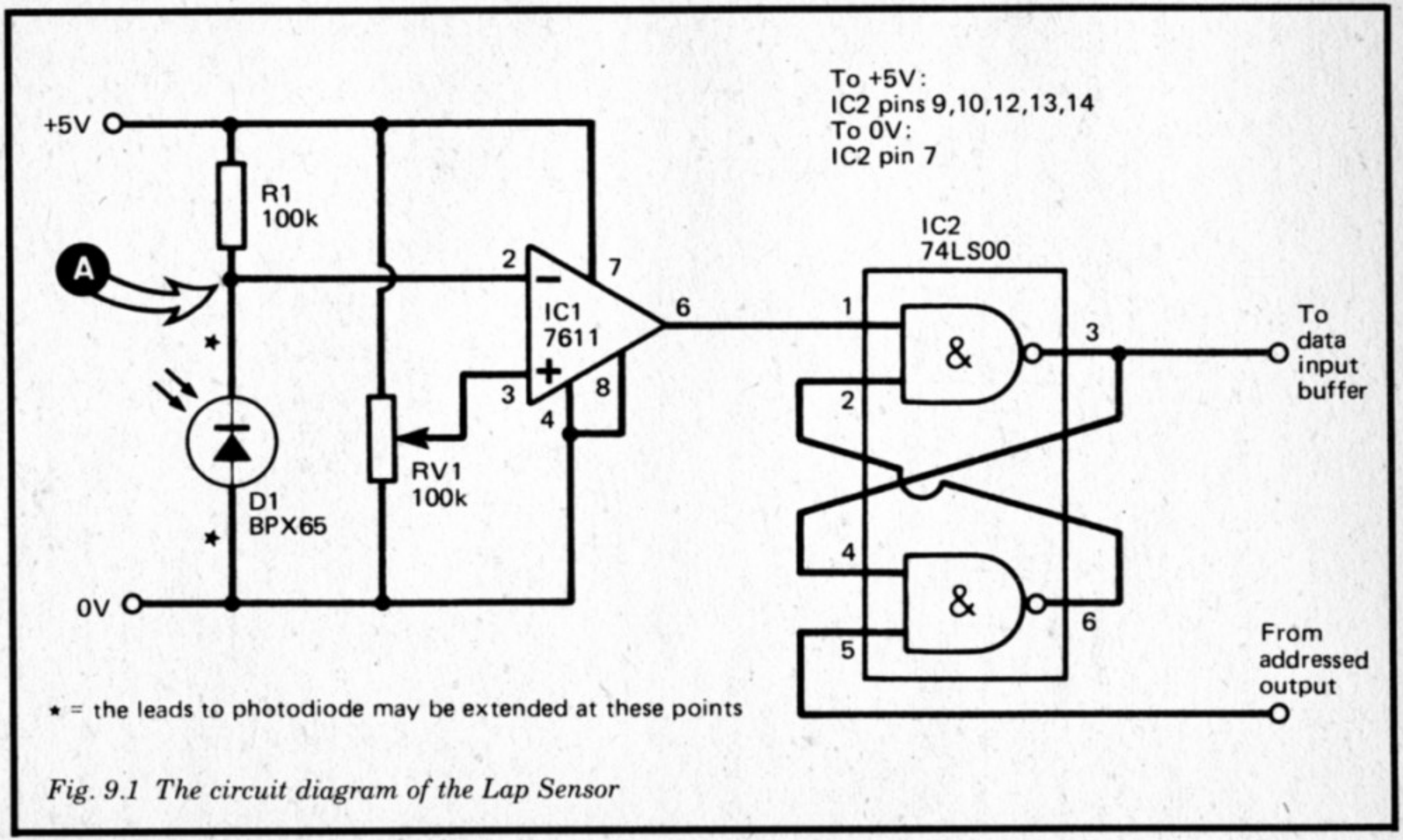


Fig. 9.1 The circuit diagram of the Lap Sensor

with a 60 W bulb is placed on one side of a corridor and the photodiode is on the opposite side pointing at the lamp, the shadow of a person passing between them is sufficient to trigger the sensor. On the smaller scale (for example, on the slot car track or model railway), an unshaded 6 V 60 mA torch bulb placed 10 cm or slightly more from the lamp is entirely suitable as a source of light. The only point to look out for is that other lamps in the vicinity do not shine on the photodiode when the light from the source lamp is supposed to be interrupted.

The photodiode works best with light from filament lamps but is relatively insensitive to light from fluorescent tubes. It works with daylight but, since the intensity of daylight varies so much with cloudiness and time of day, this is not a reliable source.

The photodiode is connected so that it is reverse biassed. The amount of leakage current increases in proportion to the amount of light falling on the photodiode. As the current increases, the PD across R1 increases so that the potential at point A falls. The operational amplifier IC1 compares the potential at point A with a standard potential at the wiper of the variable resistor RV1. We adjust the standard potential to be very slightly greater than the potential found at A when the beam of light is unbroken. The tiny difference of potential is amplified and the output of IC1 swings sharply to +5 V ("high"). When the beam is broken, the current through R1 decreases, the potential across R1 decreases, and the potential at A rises above that present at the wiper of RV1. The difference between these potentials is now in the opposite direction. It may still be a small difference, but it is amplified by IC1. Consequently, the output of IC1 swings sharply to 0 V ("low"). We can adjust the setting of RV1 to allow for light sources of differing brightness and at various distances.

The output of IC1 goes to a flip-flop (IC2). The flip-flop is reset by a low pulse from an addressed output of the Decoder. In this state its own output goes "low". The output of the flip-flop is connected to a data input of the Decoder. In the "reset" state the computer reads this output as a 0 on the data line.

When the beam of light is broken, even for a few milliseconds, the flip-flop is triggered to change state. Its output goes "high". When the micro next reads from the Lap Sensor's address, it finds a 1 on the line. The flip-flop remains in this state indefinitely, so it does not matter if the micro does not read the Lap Sensor *immediately* after the flip-flop has been triggered. This means that the micro does not miss brief interruption of the beam if it happens to be engaged in some other part of its program at the time it occurs.

When the micro has read the Lap Sensor and found that it has been triggered, it resets it by writing to its address. This causes a low pulse to appear at the addressed output and this triggers the flip-flop to reset. Now it is ready to detect the next interruption of the beam.

Building It

The circuit is housed in a small case with the potentiometer RV1 mounted on it. The photodiode can be hidden inside the case with an aperture cut in the wall to allow light to enter. This arrangement helps screen off unwanted light from other sources. In certain applications it may be more convenient to have the light source and photodiode at some distance from the micro, perhaps in another room. In this event, the circuit should be close to the micro, but the leads to the photodiode may be extended where marked * in Fig. 9.1.

The circuit requires only 4 wires to connect it to the Decoder board: the two power lines, and connections to an addressed output and a data input.

There are no difficulties in wiring up the circuit. When it is complete, connect it to a power supply (+5 V or +6 V) and connect a voltmeter to point A. Point the photodiode towards a nearby light source (e.g. a table lamp). As you place your hand between the lamp and the photodiode, the voltage at A should rise. Then connect the voltmeter to the output of IC1. Adjust RV1 until the output *just* swings to +5 V. Interrupting the beam makes it swing sharply to 0 V. Finally, connect the voltmeter to the output of the flip-flop (pin 3, IC2). Briefly connect pin 5 to the 0 V line to reset the flip-

flop. Its output goes high (+2.5 V or slightly more). When the light beam is interrupted, the output falls to "low" (close to 0 V9. The circuit is now ready to be tested on the micro.

To test the device, first reset it by triggering its addressed output. The required program lines appear on pp. 15–18. Then read the state of its output using the lines on pp. 15–18. The result should be "0". Now place your hand between the light source and the sensor. Read the state of its output again. This time the result should be "1", "2", "4" or "8", depending on which line of port B it is connected to. Finally, reset it by triggering its addressed output again. Read its output to check that this has returned to "0".

Programming

The commands to read and to reset the Lap Sensor are easy enough to understand but there are one or two points to consider when using them. Take, as an example, this program for the Commodore 64:

```
1Ø POKE 56579,24Ø
2Ø POKE 56577,32
3Ø POKE 56577,224
4Ø R=PEEK(56577)AND2
5Ø IF R=Ø THEN 4Ø
6Ø N=N+1:PRINT N
7Ø POKE 56577,32
8Ø POKE 56577,224
9Ø R=PEEK(56577)AND2
1ØØ IF R=2 THEN 7Ø
11Ø GOTO 3Ø
```

Where the same address is POKEd and PEEKed repeatedly in a program it saves memory, makes the program RUN faster and also saves typing if you first define a variable to hold the value of the address. If the statement `PB=56577` is put at the beginning of the program, all you need to type later is `POKE PB,n` and `PEEK(PB)`.

In the program above, line 10 initialises port B, using DDRB. Line 20 triggers the addressed output to reset the

flip-flop of the device. This is done in case the flip-flop has gone into the "set" state when the circuit is first switched on. It is assumed that the device is plugged into output 2. Line 40 reads the data from port B (group 1, on PB1). If the result is "0", the beam has not been broken. The program loops back to read the device repeatedly. When the beam is broken the reading is not "0" and the computer then drops through to line 60. Here it increments a counter variable, N, recording how many times the beam has been broken.

Having done this it resets the flip-flop (line 70). At least, it *attempts* to do so. Micros work so fast that it is likely that the object which interrupted the beam has not yet had time to move out of the beam. Lines 80 and 90 read the output to find out if the attempt to reset the flip-flop has been successful. If not successful, the object must still be breaking the beam. The program loops back to line 70, continually repeating its attempt to reset the flip-flop. When the object eventually moves clear of the beam, line 70 will be able to reset the flip-flop. The reading will then change to "0". The program continues to line 90 and then line 30. This example demonstrates that it is important to reset the sensor at the beginning of the operation, and one that must allow the beam to be restored and to *confirm* that the flip-flop has been reset after every interruption. A program such as this can be part of a larger program designed to count objects on a conveyor belt, the number of people passing into a room, or the number of times a slot-car has travelled around the track.

Lap-timing programs need two routines similar to the above (excluding lines 60 and 110). The first routine is to detect an interruption and then set the micro's internal clock. In the VIC-20 or Commodore 64, we might replace line 60 by:

```
6Ø TI$="ØØØØØØ"
```

In the BBC Microcomputer, the equivalent line is:

```
6Ø TIME=Ø
```

Timing commences from the instant the beam is broken as the car begins the lap. The program requires a second routine like that above to detect when the car completes the

lap and breaks the beam again. In this second routine the elapsed time is recorded on the VIC-20 or Commodore by:

```
15Ø PRINT TI/6Ø;"SECONDS"
```

The equivalent program line for the BBC machine is:

```
15Ø PRINT TIME/1ØØ;" seconds"
```

Parts Required for the Lap Sensor

Resistors

R1	100 k carbon, 0.25 W
RV1	100 k variable potentiometer

Semiconductor

BPX65 photodiode (or almost any type)

Integrated Circuits

IC1	7611 CMOS operational amplifier
IC2	74LS00 quadruple 2-input NAND gate

Miscellaneous

Circuit board and case
Knob for RV1
8-pin IC socket
14-pin IC socket
3-way sockets to fit 3-way plugs on decoder board (2 off)
1 mm terminal pins (9 off)
Connecting wire
A source of light

PROJECT 10

Photo-flash

There are really two sections to the project, and you may use either one of them or both together. One section allows the micro to fire a photographic flash gun. The other section is a sound sensor. The two are intended to be used together to fire the flash gun when a sound is heard. One application of this is in high-speed photography. The camera is aimed at an inflated balloon (or even a glass bottle). The lights are turned out and the shutter is opened. Prick the balloon with a pin (or strike the bottle with a hammer) and the resulting noise triggers the flash. The result is a photograph of the bursting balloon or breaking bottle. Of course, a sound sensor could be connected *directly* to a circuit to fire a flash gun, but having these circuits interfaced to a micro means that you can program the micro to delay the firing until a fixed period of time after the sound occurs. In this way you can obtain photographs taken at different intervals after the initial impact. With more than one flash circuit, the micro could fire a succession of flashes, producing multiple images on one frame.

Another application is to take photographs of nocturnal animals. A noise made by the animal triggers the flash. The sound-detection circuit can be used on its own in many ways. It is particularly sensitive to sharp noises (clapping, snapping fingers) and to the higher-pitched whistles, so it provides a way of controlling the micro by sound. This section of the project could be used in conjunction with Project 4 to control the action of a model locomotive or slot car by blowing a whistle, by clapping your hands, or by calling out "Go!" or "Stop".

Project 8, the Magnetic Catch is another one which can be controlled by sound, through the agency of the micro.

The photoflash section can also be used on its own, perhaps simply to give a timed delay in taking a photograph of a

group when you want to be included in the picture. Project 9 can be useful with a photoflash. Instead of triggering the flash by sound, why not trigger it when a beam of light is broken?

How It Works

The photoflash section is controlled by an Addressed Output which is connected to a flip-flop (Fig. 10.1). The flip-flop is reset by pressing S1. Its output is then "low". When the device is addressed, the low pulse from the addressed output sets the flip-flop. Its output goes "high" switching on transistor Q1. The coil of RLA1 becomes energised and the contacts of the relay are closed. The relay contacts are connected across the terminals of a flash gun, which then fires. Note that power to operate the flash comes from the flash gun, not from this circuit. The relay contacts do the same job as the contacts inside the camera which are normally used for firing the flash. Once the flash has been fired, the flip-flop must be reset manually before the flash can be fired again. It is possible to dispense with S1 and wire pin 5 of IC2 directly to another addressed output. This would allow the micro to reset the flip-flop ready for another photograph. If you are using the kind of flash gun which requires flashbulbs to be renewed there is little point in doing this. If you have an electronic flash gun, it is possible to program the micro to make multiple exposures by firing the gun again after a short interval.

The sound sensor takes the signal from a crystal microphone and amplifies it by the operational amplifier IC1. In the absence of sound the output from the amplifier is about 2.5 V. When sound is detected it oscillates above and below this level. Low-going voltage changes act as a low pulse to the other flip-flop, causing it to set. Its output, which is normally "low", changes to "high". This can then be read by the micro. Note that the flip-flop stays high once it has been set, so it does not matter if the micro is not actually reading data from the sensor at the instant the sound occurs.

The flip-flop is reset by pressing S2. For the original

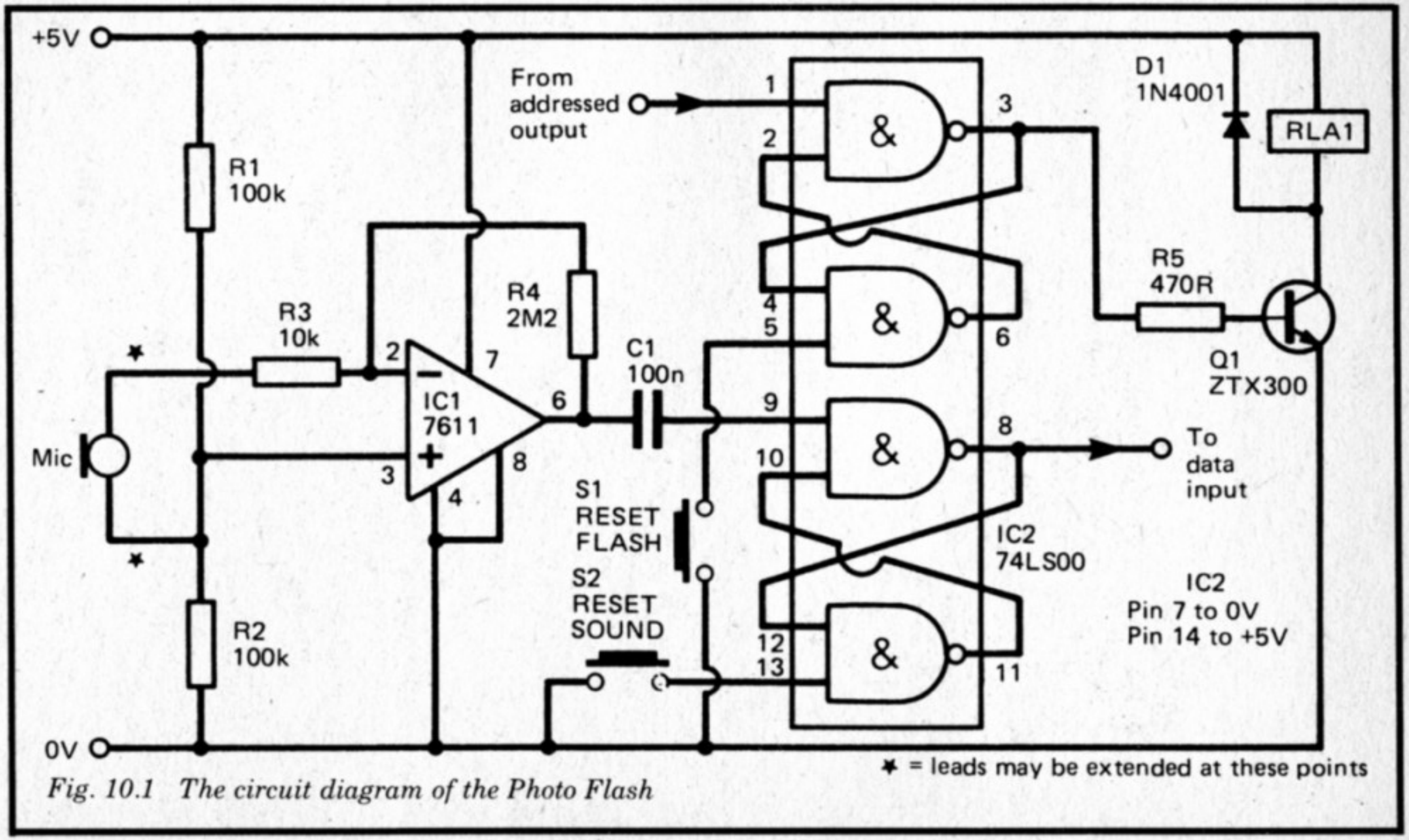


Fig. 10.1 The circuit diagram of the Photo Flash

application of this project, manual resetting is all that is required. However, in some applications it might be convenient to have the micro do the resetting. This is done by replacing S2 with a wire from pin 13 of IC2 direct to an addressed output of the Decoder. This allows the micro to reset the flip-flop after a sound has been detected, ready to detect the next sound.

The device needs a minimum of four lines to connect it to the micro: the power lines, plus connections to an addressed output and a data input. It needs more lines if either of the flip-flops are to be reset by the micro, instead of being reset manually.

The photoflash should be housed in a case, on the outside of which the microphone can be mounted. The two push buttons are mounted on the panel of the case. The switch contacts of the relay are wired to a socket, on the panel of the case. You need a plug to fit this socket, with wires leading to the flash gun.

As explained earlier, this project consists of two parts, and there is no need to build both if you are interested in only one aspect of it. If you want to operate several flash guns, it is easy to add a second 74LS00 IC from which two more flip-flops may be constructed.

When assembly is complete and the project has been tested for short circuits between the various lines which link it to the micro, it may be given its preliminary test. Connect it to a +5 V or +6 V supply. If you alternately press S1, and touch a grounded (0 V) wire to pin 1 of IC2, you should be able to see or hear the relay switching on and off. Then connect a voltmeter to the data output (pin 8 of IC2). Press S2 and the output goes low. Make a sound near the microphone and the output abruptly goes "high" (+2.5 V or more). It responds best to sharp clicking or clapping sounds. Try snapping your fingers about 50 cm from the microphone. Even a gentle "snap" should be enough to trigger it. If this fails, try tapping the microphone gently. If this fails, the circuit is at fault. Failure to respond to sound of reasonable loudness may be due to lack of output from your microphone. To improve amplification, try substituting a resistor of higher

value for R4. A resistor of 3M3 or 4M7 will increase the gain.

The Photo Flash may now be tested on the micro. Plug its leads in the Decoder and switch on. Press both reset buttons. If you have an electronic flash, you can use this while testing. If you need to use expendable flashbulbs, this would be too expensive a session. Instead, wire up an ordinary low-voltage bulb, in place of the flashbulb shown in Fig. 10.2. Trigger the addressed output as described on pp. 15–18. Then read port B as described on pp. 15–18. When the sound-sensor flip-flop is reset its output is low, so it reads "0". When a sound has been detected, the reading changes to "1", "2", "4" or "8" depending on which buffer it is connected to.

Programming

Controlling the Photo Flash is relatively simple, and several examples of possible uses for it have already been described.

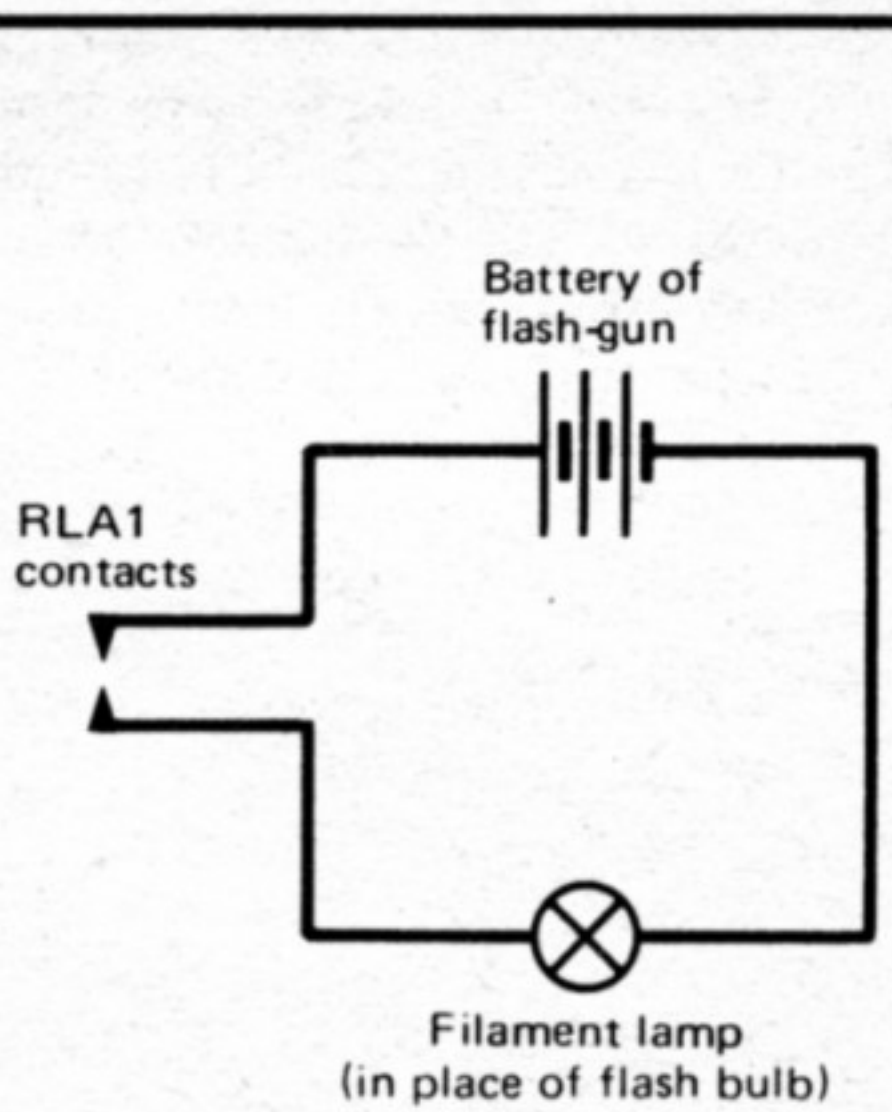


Fig. 10.2 How to use a filament lamp while testing the circuit

Here are programs which fire the flash a short period after a sound has been detected.

This one runs on the VIC-20:

```
1Ø POKE 37138,24Ø
2Ø POKE 37136,224
3Ø R=PEEK(37136)AND4
4Ø IF R=Ø THEN 3Ø
5Ø TI$="ØØØØØØ"
6Ø IF TI<6 THEN 6Ø
7Ø POKE 37136,48
8Ø PRINT"PHOTO TAKEN"
9Ø STOP
```

The computer waits in a loop (lines 30–40) until it detects that the sound has caused the flip-flop to be "set". Then it puts the system clock to zero. It waits in a second loop (line 60) until 6 "jiffies" have elapsed (i.e. a tenth of a second). Then it triggers the addressed output, firing the flash. It also prints a message to indicate this fact. The program assumes the use of addressed output no. 3, and receives data from the device on PB2. The length of the delay can be adjusted by altering the value in line 60. The Commodore 64 is programmed in an identical way, except for the addresses.

The program for the BBC Microcomputer follows the same pattern as above:

```
1Ø DDRB=65122:?DDRB=24Ø
2Ø PORTB=6521Ø:?PORTB=224
3Ø REPEAT R=?PORTB AND 4
4Ø UNTIL R<>FALSE
5Ø TIME=Ø
6Ø REPEAT UNTIL TIME=1Ø
7Ø ?PORTB=48
8Ø PRINT"Photo taken"
9Ø END
```

In the program above the micro waits in a loop until it finds that the sensor has been triggered. Then, after a pause, the length of which you can adjust, it fires the flash. It would be a useful precaution to begin the program with a routine

which reads the sensor to make sure that you have remembered to reset it first. If it finds that it is not reset, it displays a message to that effect. Then when it detects that you have reset it, it goes on to the routine listed above, waiting for the sound to occur. Something of this sort will eliminate the risk of the micro firing the flash as soon as you run the program.

Parts Required for the Photo Flash

Resistors (carbon, 0.25 W, 5% tolerance)

R1, R2	100 k (2 off)
R3	10 k
R4	2M2
R5	470 R

Capacitor

C1	100 n polyester

Semiconductors

Q1	ZTX300 or similar npn transistor
D1	1N4001

Integrated Circuits

IC1	7611 CMOS operational amplifier
IC2	74LS00 quadruple 2-input NAND gate

Miscellaneous

RLA1	Reed relay or other single-pole single-throw relay operating on 6 V
S1, S2	Push-to-make push-button switches (2 off)
MIC	Crystal microphone or microphone insert

8-pin IC socket
14-pin IC socket
Socket for connection to flash gun
3-way plugs to fit 3-way sockets on Decoder board (2 off)
Circuit board
1 mm terminal pins(8 off)
Connecting wire

Another refinement, especially useful if you are using the device to photograph any animals which might come close to the camera, is to insert a long pause in the program to give you time to get away before it becomes able to respond to sound. In this event it would be a good idea to have the sensor reset by the micro, so that the sounds of your departure do not permanently trigger the sensor.

If you are using the micro to reset the sensor after a sound is detected, the program must test that it has reset successfully. If it attempts to reset whilst the sound is still continuing, it will be unsuccessful.

PROJECT 11

Games Control

A Games Control adds a whole new aspect to computer fun. It consists of a small box with a control knob on top. As you turn the knob one way or the other, you move a "bat" round the screen, aim a "laser gun", steer a "racing car", or in some other way take a very active part in the game. You can build and operate 2 or more of these if you wish. Another possibility is to house this project in the same case as the Five-key Pad (Project 3).

How It Works

There are three slightly different versions of this add-on, one for each computer (Fig. 11.1). The circuit for the VIC-20 is the simplest, consisting only of a potentiometer. It sends a variable current to the "POT X" (pin 9) or "POT Y" (pin 5) terminals of the games port (or controller port). This current charges a capacitor which is inside the computer. The rate of charge of the capacitor depends on the amount of current flowing which, in turn depends on the setting of RV1. The computer measures the time taken for the capacitor to charge and then discharges it again automatically. The charging time is recorded as a value which is stored in one of two registers.

The addresses of the registers are 36872 (POT X) and 36873 (POT Y). The value found in a register ranges between 0 (input = +5 V) to 255 (input = +0 V). You can have two separate games controls, one wired to each of POT X and POT Y (each as in Fig. 11.2a). These are suitable for two-player games, each player having a control. As explained in the programming section, each control could be used for moving a "bat" or similar object across the screen, or for aiming a gun or other weapon. If the project is contained in the same case as the Five-key Pad, the keys can be used as firing but-

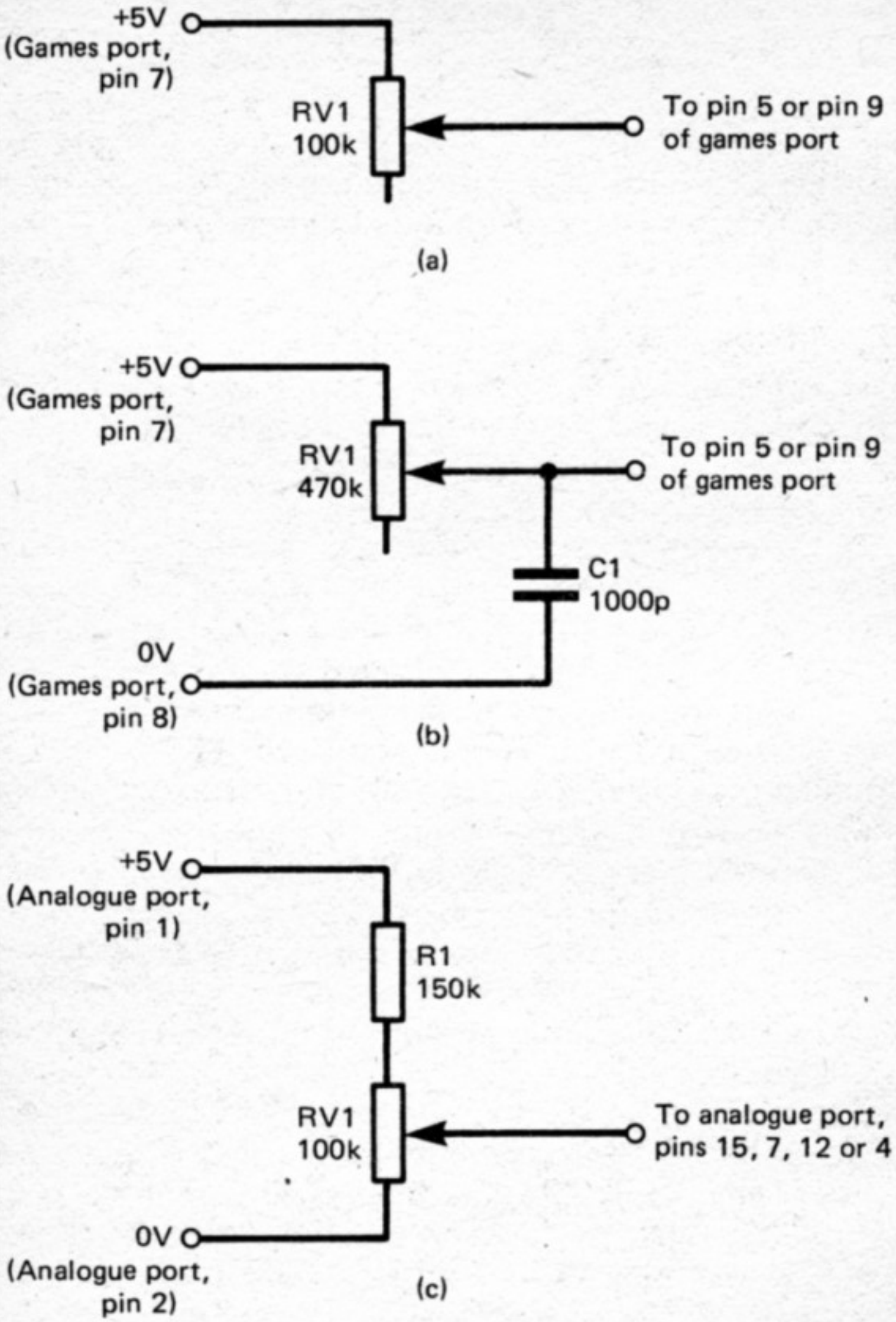
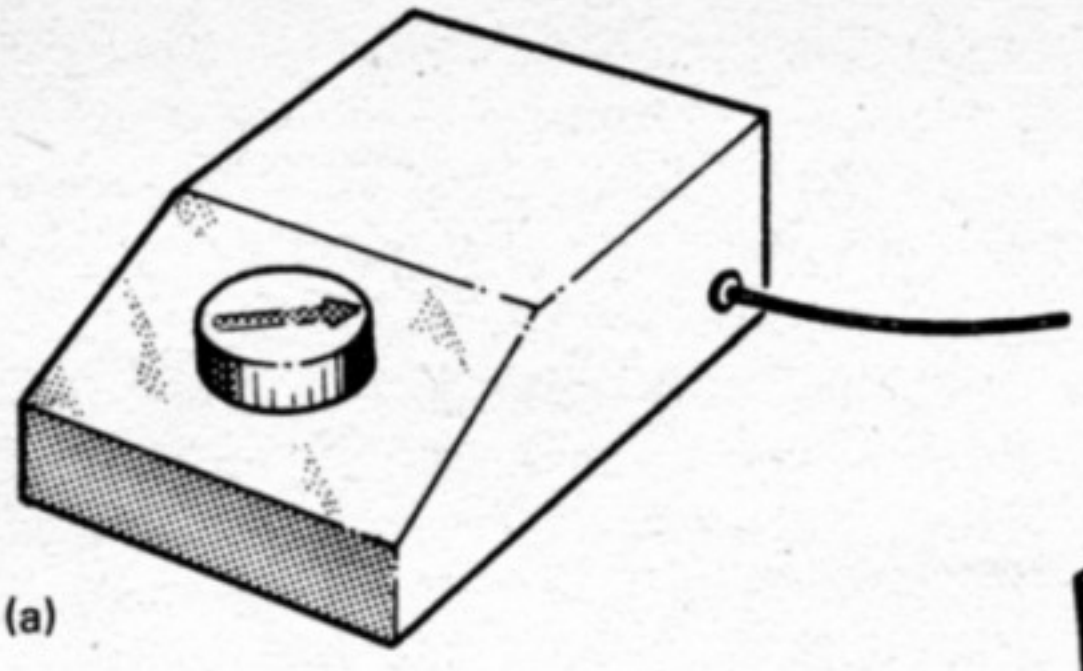
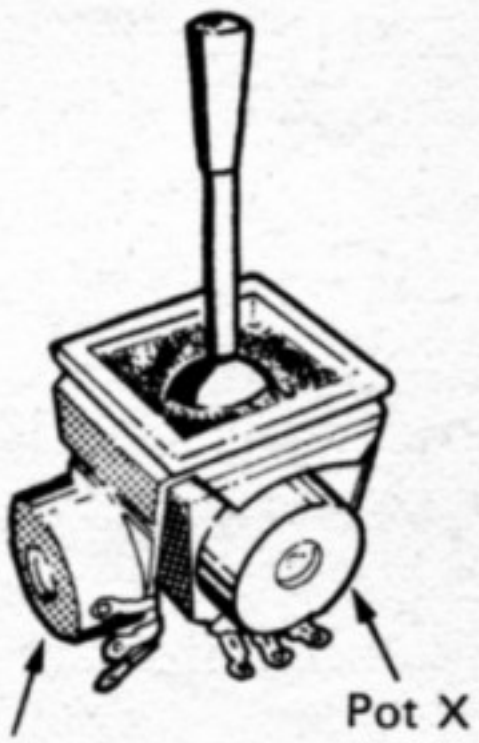


Fig. 11.1 The circuit of the games control for (a) VIC-20, (b) Commodore 64 and (c) BBC Microcomputer

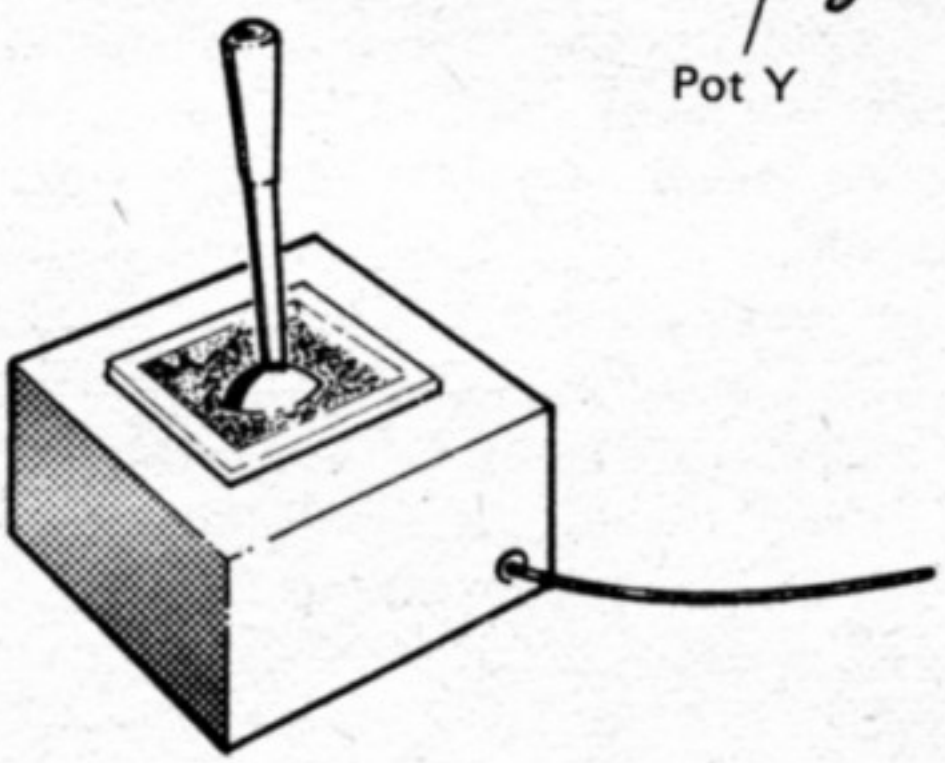


(a)



Pot Y

Pot X



(b)

Fig. 11.2 Types of controller; (a) single potentiometer; (b) joystick. Inset: a commercially-made joystick unit, ready for mounting

tons and in other ways. An alternative approach is to wire up the two potentiometers of a joystick, connecting one to POT X and the other to POT Y, as in Fig. 11.2b. This gives one player control in two dimensions. An object can be moved or steered both from left to right *and* up and down.

The device works in a similar manner on the Commodore 64, except that no capacitor is provided inside the computer, so we have to include one (C1) in the add-on itself. This computer has two games ports so that 4 potentiometers can be attached in total. You can have 4 simple controllers as in Fig. 11.2a, or in a pair of joysticks, as in Fig. 11.2b. The addresses to be PEEKed in order to read the position of each potentiometer are:

POT X (pin 9) – address 54297
POT Y (pin 5) – address 54298

To read the pair of potentiometers plugged into games port 1, bit 7 and 6 of address 56320 must be set to "0" and "1" respectively. To read the pair on games port 2, the bits must be "1" and "0". The way to program this is described later.

The circuit for the BBC Microcomputer Model B consists of a resistor and potentiometer wired in series as a potential divider network. As the wiper of RV1 is swept from one end of the track to the other, the voltage at the wiper ranges between 0 V and about +1.8 V. This voltage is fed to any one of the four analogue-to-digital converters in the computer. It is converted to a digital value in the range 0 to 65520 (in steps of 16). These are read by using the ADVAL statement. BBC Model A will need the analogue port upgrade.

For Acorn Electron *see* note on page 187.

Building It

All that is required in a suitable case or console on which to mount RV1 (Fig. 11.2a). C1 or R1, if required, may be soldered directly to the appropriate terminal of RV1. Leads are taken to a "D" connector. Pin numbers are given in Fig. 11.1. A simple test program for the VIC-20 is:

```
10 PRINT PEEK(36872)
20 FOR J=1 TO 200:NEXT
30 GOTO 10
```

RUN this program and watch the displayed numbers change as you turn the knob of the Controller. The numbers should range from 0 to just over 100. Values in the range above 100 are not usually obtained with this Controller. The reason for avoiding high values is that towards the higher end of the scale, the values increase rapidly for a relatively small change in the setting of the knob. They also tend to be erratic. A range of 0–100 gives more than enough resolution for moving objects across or up and down the VIC-20 screen.

A similar program is used to test the paddle when plugged into games port 1 of the Commodore 64:

```
10 POKE 56320, (PEEK(56320)AND63)+64
20 PRINT PEEK(54297)
30 FOR J=1 TO 200:NEXT
40 GOTO 20
```

Line 10 sets bit 7 of 56320 to 0 and bit 6 to 1, without altering the other bits at that address. If the Controller is plugged into games port 2, then change line 10 to:

```
10 POKE 56320, (PEEK(56320)AND64)
```

The BBC Microcomputer is simpler to organise for testing. The Controller may be wired to any one of the 4 analogue inputs, which are numbered from 1 to 4. If it is connected to input 2, for example, the program is:

```
10 PRINT ADVAL(2)
20 FOR J=1 to 200:NEXT
30 GOTO 10
```

With this computer, the displayed value changes much more smoothly or steadily over its range, giving excellent resolution.

Programming

This is extremely easy, for all that has to be done is to read the position of the Controller knob, using program lines like those given in the previous section. The value obtained is then used in the plotting of the moving "bat", "ball", "rocket ship" or whatever other object on the screen is to be controlled. One might also use such values for the control of external devices attached to the Controller, for example, the Model Controller (Project 4) or a robot.

The program below, for the VIC-20, illustrates how to use Controller values:

```
10 PRINT" ":G=7680:H=3840
20 FOR J=0 TO 21:POKE G+J,81:NEXT
30 FOR J=0 TO 21:POKE H+J,1:NEXT
40 POKE H+PEEK(36873)/4,2
50 GOTO 50
```

As the knob is turned, a red "ball" moves back and forth across the top of the screen. Line 20 "prints" a row of balls (or disk symbols) across the top of the screen. Since line 30 makes the printing colour white, they are being printed in white on a white background, and are invisible. The program alternately makes all balls white, then displays one of them in red (line 40). The effect is as if a single ball moves across the screen. Note that the value obtained by PEEKing is scaled down in line 40, by dividing it by 4. This is because tests showed that the readings varied from 0 to just over 80 when the knob was turned. Scaling by this factor makes the ball move across the full width of the screen.

In this program we are POKEing addresses in the video RAM, depending on the value read from the Controller. In certain applications it is important to ensure that these readings will not result in POKEing an address beyond the range of video RAM. If an address in some other part of memory is POKEd accidentally, it will do no harm to the computer but could possibly result in a "crashed" program.

The Commodore 64 program would be similar to that given above, and the program for the BBC Microcomputer

would operate on similar lines, depending on the graphics mode employed.

Parts Required for the Games Control

VIC-20 Version

Resistor

RV1 100 k variable potentiometer

Miscellaneous

Case suitable for mounting potentiometer
Knob for RV1, suitable for use in games
9-way "D" socket, to fit games port of VIC-20

Commodore 64 Version

Resistor

RV1 470 k variable potentiometer

Capacitor

C1 1000 p polystyrene

Miscellaneous

Case suitable for mounting potentiometer
Knob for RV1, suitable for use in games
9-way "D" socket, to fit games port of Commodore 64

BBC Microcomputer Version

Resistors

R1 150 k carbon, 0.25 W, 5% tolerance
RV1 100 k variable potentiometer

Miscellaneous

Case suitable for mounting potentiometer
Knob for RV1, suitable for use in games
15-way "D" socket, to fit analogue port of BBC Microcomputer

Joysticks

Several commercially available joysticks have 220 K potentiometers. These should be quite suitable for all 3 machines. On the Commodore 64, increase C1 to 2200 p. On the BBC Microcomputer increase R1 to 390 K.

PROJECT 12

Rain Detector

The remaining projects in the book make up a weather station series. They can all be plugged into the Decoder and operated together to record certain aspects of the weather. These are easy projects and are not precision instruments, but they give enough information to allow the micro to try to predict what the future weather is likely to be. Although these last six projects are concerned with meteorology, many of them have several other useful applications.

This project tells you when it is raining. Not only does this allow the micro to keep a record of the number of rainy days, or the number and duration of rain showers, but it has other uses around the home. On washdays it is handy for detecting when it starts to rain. If the Bleeper (Project 5) is connected at the same time, and is located in the kitchen (or beside the TV set?), the micro can sound a warning that the washing must be taken in. The project also detects when water rises above a given level. In conjunction with the micro it can warn you when the bath is overflowing, or when the well is running dry.

How It Works

The base current to the transistor (Q1, Fig. 12.1) comes by way of a probe. This consists of strips of conducting material placed close together. Their distance apart is such that raindrops are able to bridge the gap. A single raindrop will allow sufficient conduction to occur to turn on the transistor. The result of this is that the level at the data input falls from "high" to "low". Fig. 12.2 shows an easy way to make the probe from a scrap of stripboard.

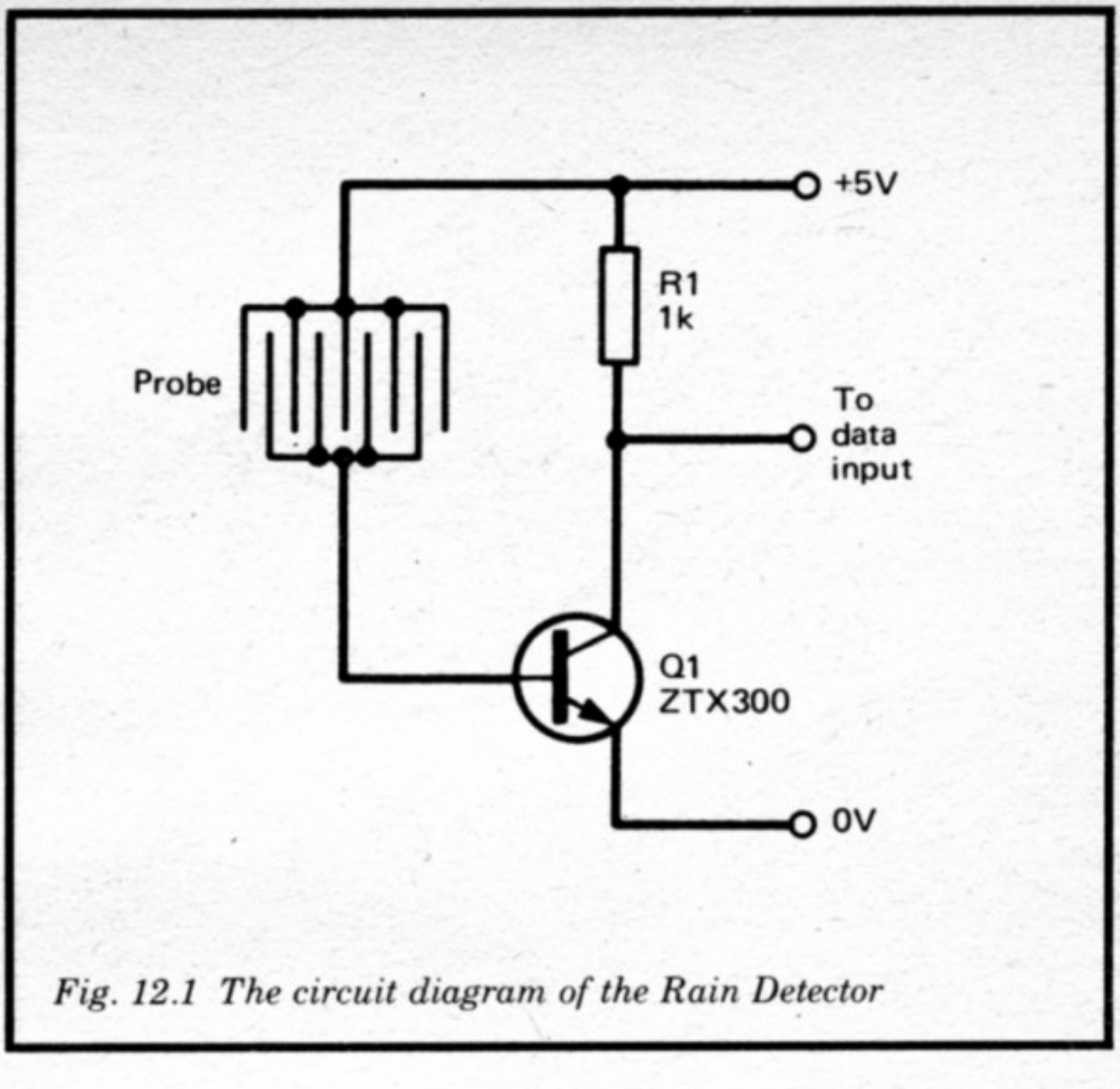


Fig. 12.1 The circuit diagram of the Rain Detector

Building It

Two small scraps of stripboard are required, one for the circuit and one for the probe. It is in fact possible to build the whole device on one piece of board, covering the area on which the components are placed and leaving the rain-detecting area exposed. This circuit gives a DC output, so it is in order to have a long wire connecting the output to the data input of the Decoder board.

Test the circuit by connecting a voltmeter to its output. It should read +5 V. When you let a drop of water fall on the probe, the voltage should fall close to 0 V. The same effect can be obtained by touching the probe with a moistened finger, so this device could also have applications as a touch-control switch.

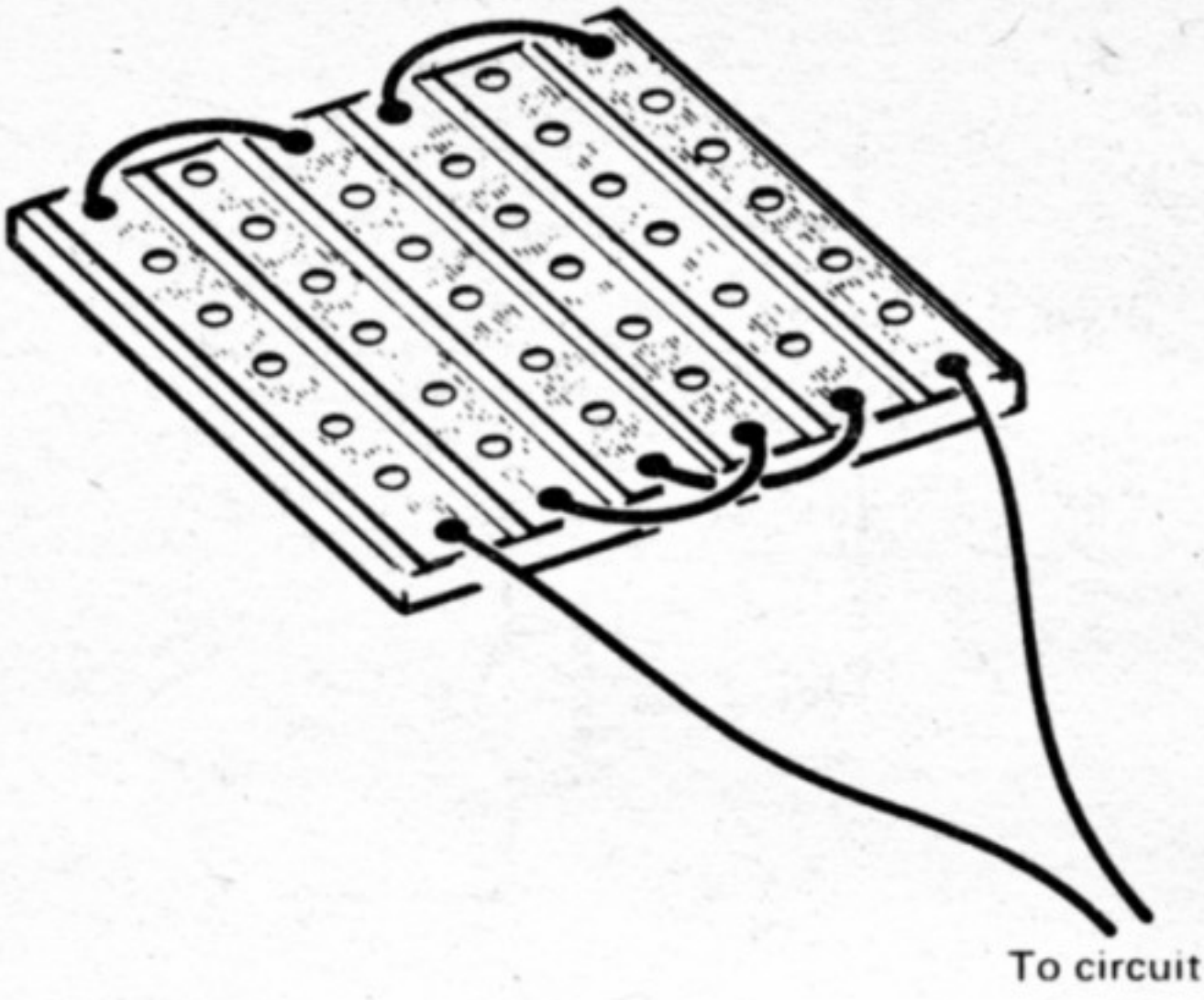


Fig. 12.2 The probe for the Rain Detector

The probe should be mounted in an exposed position outdoors, so that buildings or trees do not protect it from driving rain coming from certain directions. It is best to slope it slightly, so that rain drains away when the shower is finished, and the probe dries quickly.

Programming

All that is needed is a program line which reads the input from the project at regular intervals (*see* pp. 15–18).

Parts Required for the Rain Detector

Resistor

R1 1 k

Semiconductor

Q1 ZTX300 or almost any npn transistor

Miscellaneous

Stripboard for circuit and probe (small scrap)
3-way socket to fit 3-way plug on Decoder board
Connecting wire

PROJECT 13

Weathercock

Knowing the direction of the wind is an essential requirement for weather forecasting. This project reads wind direction and passes the information to the micro. It is also interesting as an example of the way in which the micro can be informed of a *position* which is being measured as an *angle*.

How It Works

The position of the weather vane is sensed by three light-dependent resistors (LDRs). Fig. 13.1 shows the circuit for

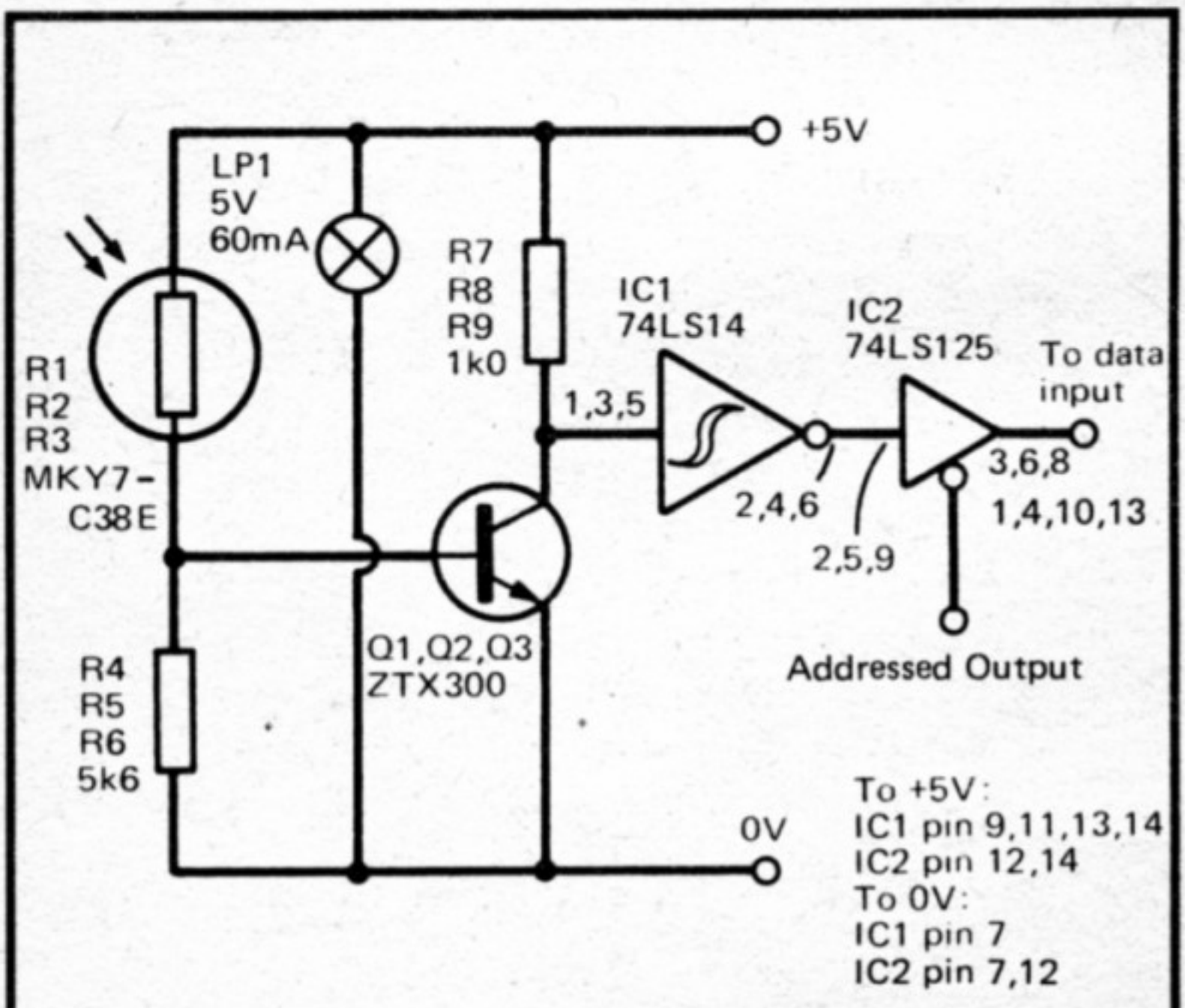


Fig. 13.1 The circuit diagram of the Weathercock

one of these. The LDRs receive light which comes from a lamp and passes through an encoder disk (Figs. 13.2 and 13.3). This disk is fixed to the shaft of the weather vane. The disk is transparent but has a pattern of black sectors on it. In any given position of the vane (and hence of the disk), each one of the LDRs is either shaded or unshaded. As the wind direction changes, the vane and the disk rotate. Some LDRs become exposed to the light from the lamp and some become cut off.

A closer look at the disk (Fig. 13.2) shows that the pattern of the black sectors does *NOT* represent a sequence of binary numbers. Reading clockwise and converting to decimal, we find: 0, 1, 3, 2, 6, 7, 5, 4. This is what is known as "Gray Code". The feature of this sequence is that as we go from one number to the next (or return to the first number from the last number) only one digit changes each time. To see why

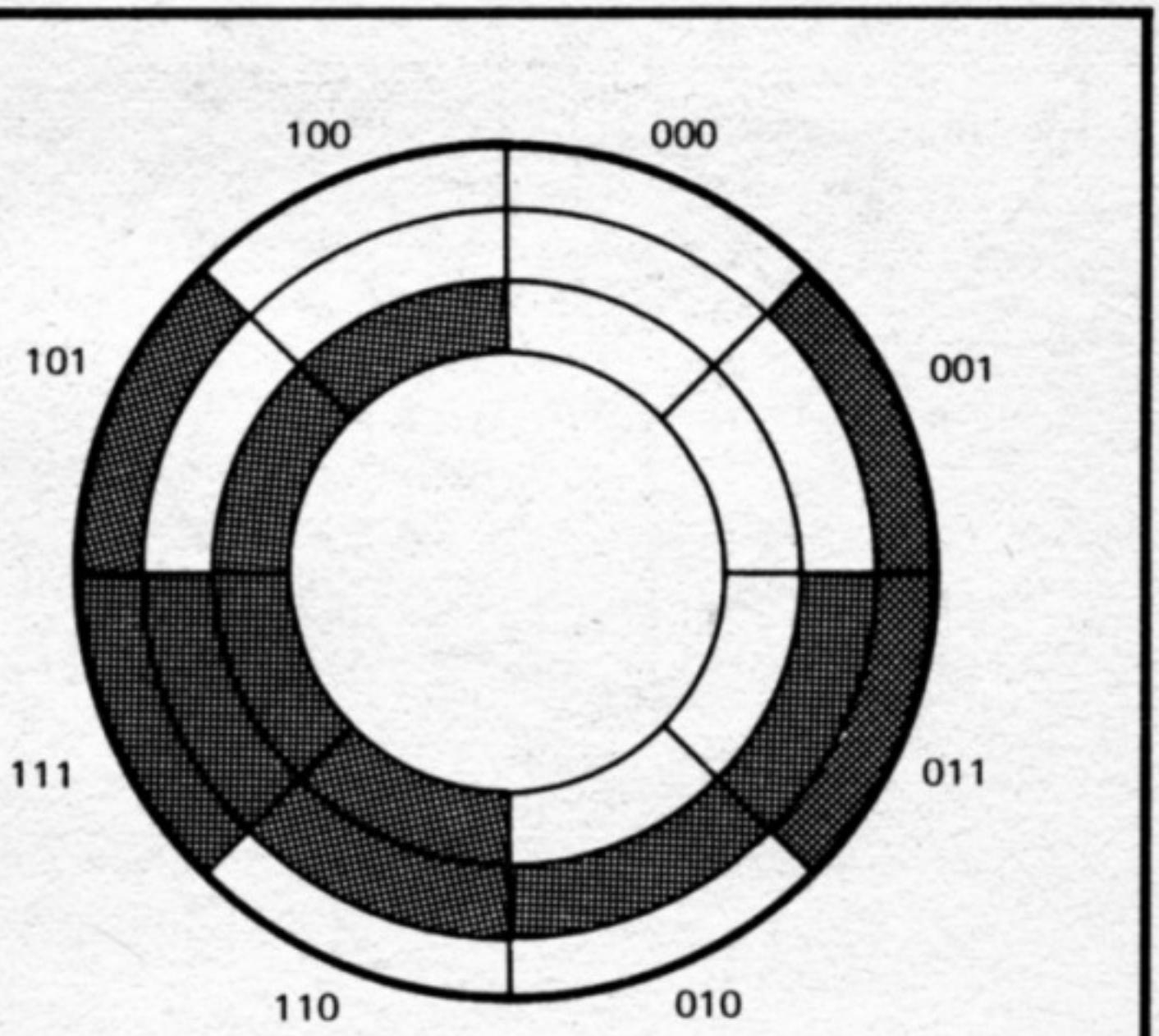


Fig. 13.2 Marking out the encoding disk for the Weathercock

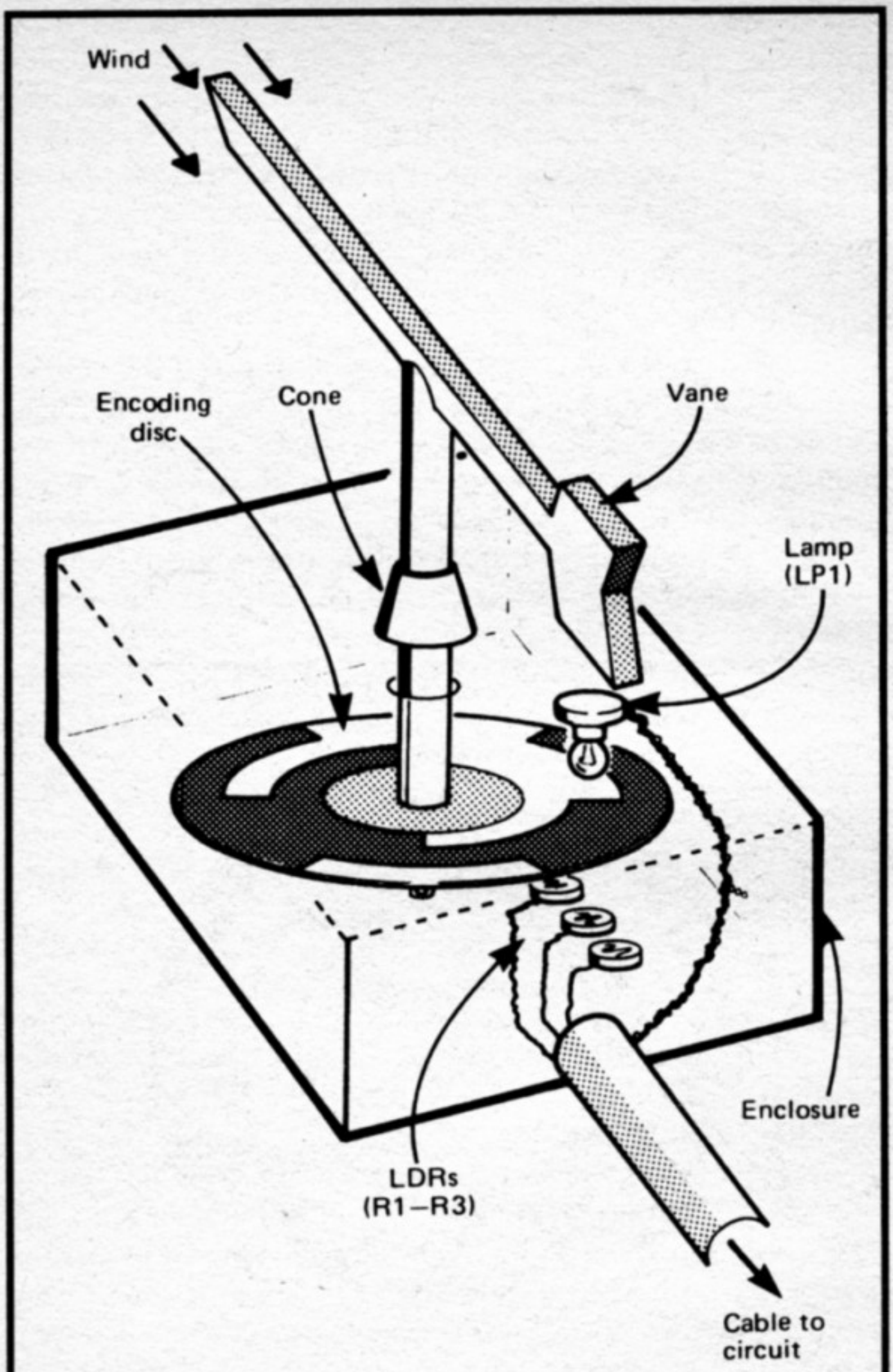


Fig. 13.3 The main features of the mechanical construction of the Weathercock

this is important let us look at what would happen if we began the sequence with 0, 1, 2, ... The first change would be:

```
          Ø Ø Ø
changing to Ø Ø 1
```

This creates no problems and this change is found in the Gray sequence. But the second change could be troublesome:

```
          Ø Ø 1
changing to Ø 1 Ø
```

This would be all right if both digits changed at exactly the same instant. In a piece of equipment such as this it is very unlikely that the LDRs are so precisely arranged and their resistances are so exactly matched that each will respond in an identical manner. They will *not* all change at the same instant. If the middle digit changes before the right-hand digit, we get an intermediate stage showing 0 1 1. We get the same if the right-hand digit changes first. Instead of getting 0, 1, 2, ... as the disk rotates, we get 0, 1, 3, 2, ... The 3 is only there briefly, but there would be time for the micro to take a reading and obtain a false result. The Gray Code overcomes this problem, since only one digit changes each time.

Building the Vane

Figure 13.3 shows the general features of a typical Weathercock. Exactly how you construct yours depends on what materials and tools you have available and on your skill in making such a device. It is often possible to adapt a ready-made weathercock. Such instruments are obtainable cheaply from shops selling educational equipment. Often they are made from plastic and while cheaper models are generally less durable than those intended for serious use, they are usually easier to adapt. The disk is made from clear transparent plastic. A sheet of thick acetate film can be used, and may be painted by using a black marker pen (felt-tip spirit pen). Alternatively, the sectors may be cut out from black insulating tape and stuck on the disk. In order that the read-

ings indicate the major points of the compass (N, NE, E, SE, etc.), the LDRs are placed so that the *centre* of each sector lies over the line of LDRs when the vane is aligned with the main compass points.

For clarity, Fig. 13.3 shows the disk several centimetres above the LDRs, but it is much better if the gap is small and the disk almost touches them. The lamp should be mounted a few centimetres above the disk, so that its light spreads fairly equally to all three LDRs. Leads are taken from each LDR back to the main circuit which is to be housed in a small case near to the micro.

Fig. 13.3 shows a way of preventing rainwater from running down the shaft of the mechanism. The cone deflects it on to the top surface of the light-proof box. It is not essential for the hole where the shaft enters the enclosure to be absolutely light-proofed, though a collar around the shaft helps to prevent light from scattering towards the LDRs.

The circuit uses three lines of the data bus (lines PB0 to PB2). IC1 has three-state outputs which are enabled by a low from the address Decoder circuit.

Building The Circuit

The device is to be plugged on to one of the 6-way plugs of the Decoder (Fig. D.0). It requires 7 lines: the two power lines, four data lines, and one addressed output. Although only three lines are used for the transmission of data, it is convenient to connect the fourth line to PB3 and hold this low. As indicated in Fig. 13.1, the input to the fourth buffer (IC2, pin 12) is connected to the 0 V line. This means that it gives a low output whenever the buffer is enabled. The circuit board also requires terminal pins for the connections to the LDRs. The type of LDR recommended is small, but any LDR of small diameter can be used instead. Larger types *can* be used if these are the only ones available, but to use these requires that the disk must also be made larger. If the LDRs are not of the type specified and have a higher or lower resistance range, it may be necessary to substitute resistors of a different value for R4–R6.

It is best to build and test the circuit for one LDR first. It is then easy to check that the correct resistance is being used for R4–R6, before proceeding with the remainder of the construction. When all is finished, temporarily wire the buffer enable inputs (IC2, pins 1, 4, 10 and 13) to 0 V to enable the outputs. Slowly rotate the disk to measure the output of each buffer with a voltmeter. Fig. 13.2 shows what results to expect, where 0 represents a "low" output (less than 0.8 V) and 1 represents a "high" output (more than 2 V). Assuming that all is correct, test all wires to the Decoder board to check that there are no short circuits between any pair of them.

Programming

All that is required is to read the data input, using the program lines from pp. 15–18. You then have to write 8 program lines to interpret this reading. This is one of them:

```
3Ø IF X=8 THEN WIND$="North"
```

You may be able to use the wind direction to predict the weather. What the prediction will be depends very much on your locality. It may also depend on other features of the current weather measured by some of the other projects. As well as reading direction, you can use the weathercock to detect changes in the wind direction over a period of several hours. If the wind is veering, that is to say, changing direction in a clockwise direction, it possibly indicates an approaching depression, with the likelihood of low cloud and rain.

Another aspect is the steadiness of wind direction. If the Weathercock is read every minute and it is found that the direction is changing frequently, this can be another clue to what future weather is likely to be. There are several books on weather which will help you work out your predictions. You may also be able to program the micro to work these out for you.

Parts Required for the Weathercock

Resistors (carbon, 0.25 W, 5% tolerance, except R1–R3)

R1–R3	MKY7C38E or similar light-dependent resistor (preferably small) (3 off)
R4–R6	5k6 (but *see* text) (3 off)
R7–R9	1k0 (3 off)

Semiconductors

Q1–Q3	ZTX300 or similar npn transistor (3 off)

Integrated Circuit

IC1	74LS14 Hex Schmitt trigger
IC2	74LS125 quadruple bus buffer gate with three-state output

Miscellaneous

LP1	5 V, 60 mA filament lamp

Socket for LP1
14-pin IC sockets (2 off, if on-board Decoder used)
6-way socket to fit 6-way plug on Decoder
3-way socket to fit 3-way plug on Decoder
Circuit board
1 mm terminal pins (10 off)
Materials for the mechanical parts

PROJECT 14

Anemometer

The anemometer measures wind speed. It has a set of three or four cups arranged on top of a shaft (Fig. 14.1) so that they spin round when the wind blows. It is assumed that the cups travel at a speed close to that of the wind itself so, knowing the radius of the assembly and the rate of rotation, we esti-

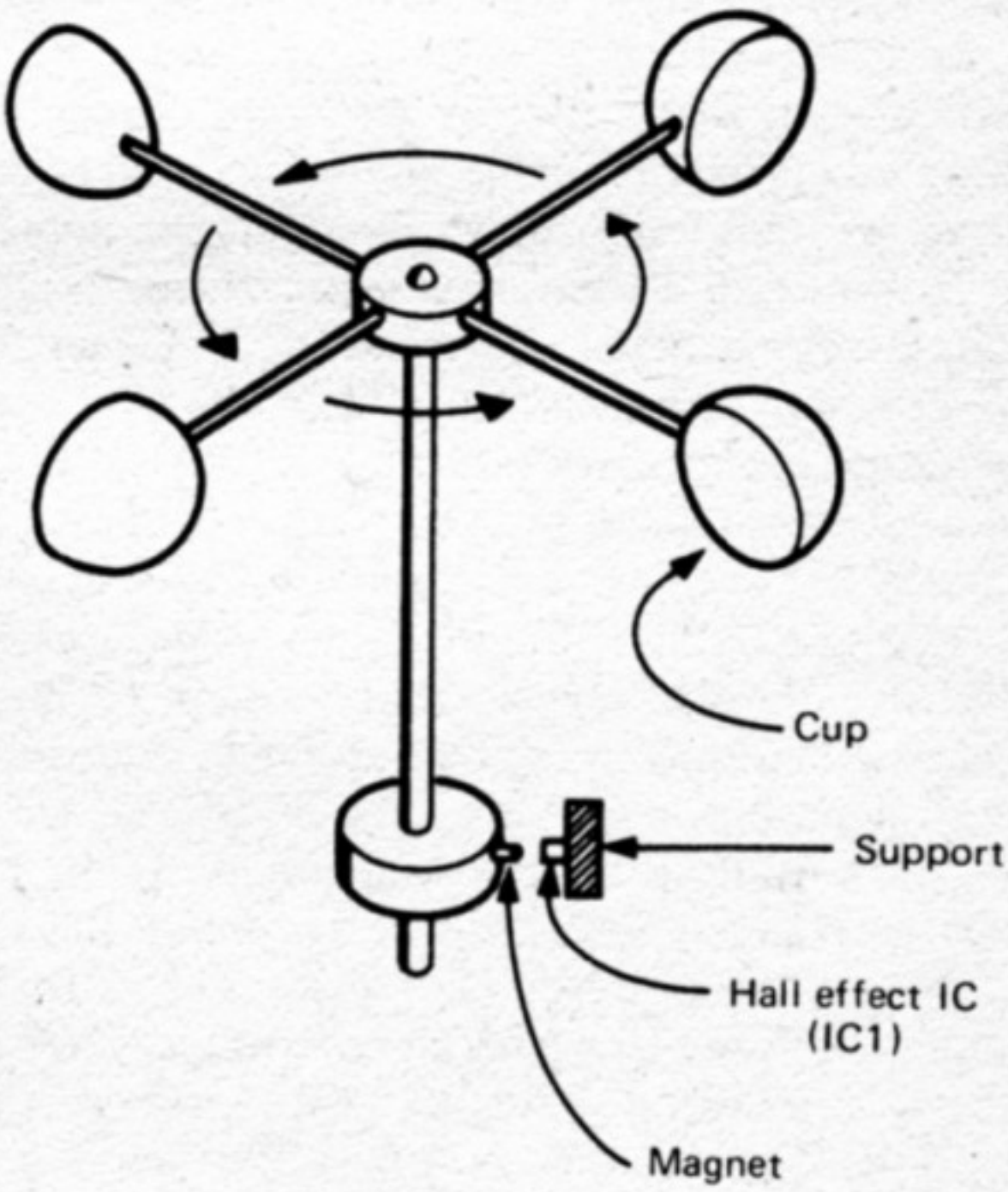


Fig. 14.1 The main features of the mechanical construction of the Anemometer

mate wind speed. In effect, the circuit of this project is a tachometer, a circuit to measure *rate of rotation*. The circuit could be adapted to measure the rate of rotation of other machinery.

How It Works

Fig. 14.1 shows that there is a wheel at the base of the shaft of the anemometer. This rotates at the same rate as the cups. The wheel is made from non-organic material such as plastic or wood. It carries a small permanent bar magnet at one point on its circumference. The magnet is arranged with one of its poles protruding slightly. As the wheel rotates, the magnet is carried past an IC which is mounted to one side of the rotating assembly. This is a Hall Effect IC. The Hall Effect occurs when a piece of semiconducting material is placed in a magnetic field. Those who remember learning Fleming's Left-hand Rule in physics lessons will know that there is an interaction between a current and a magnetic field. In Fig. 14.3 a current is passed along a slice of semiconductor. The moving electrons are deflected to one side when a strong magnetic field is present. The result is a potential difference between the two sides of the conductor. This is detected and amplified by circuits within the IC. The outputs of IC1 are normally at about 2 V. When a magnet is held close to IC1 the voltage at one output rises slightly and the other falls. The operational amplifier compares these voltages. When they are equal (or nearly equal) its output is "low". When the voltages are unequal (i.e., when a magnet is near), the output of IC2 rises sharply to +5 V. This quickly switches on Q1. The voltage at the input of the counter (IC3) falls sharply, triggering the counter. Thus the counter is incremented every time the magnet passes close to IC1.

IC3 is wired as a divide-by-16 counter, so its D output changes at 1/16 of the rate at which the anemometer is rotating. This rate is measured by the micro, which then calculates the wind speed.

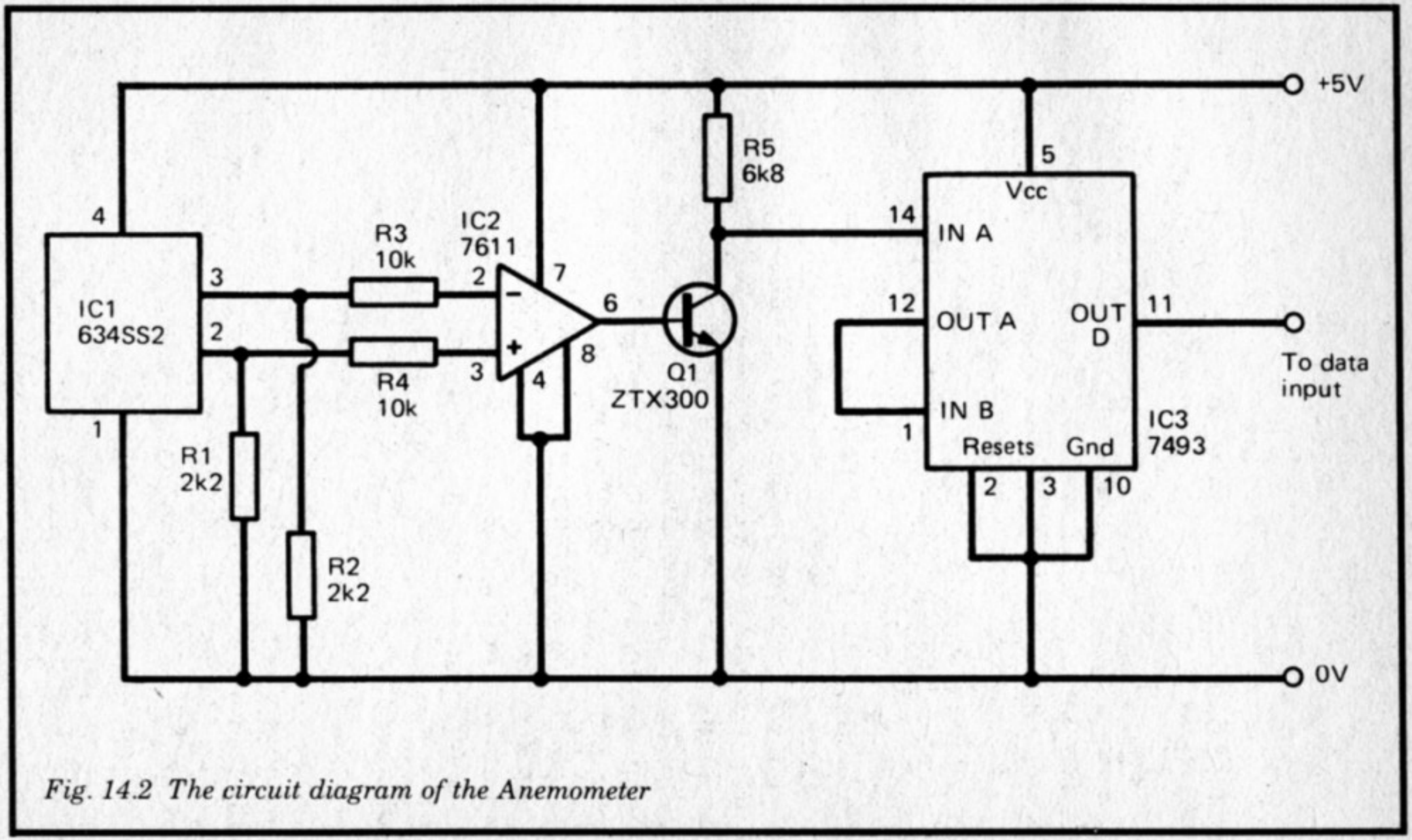


Fig. 14.2 The circuit diagram of the Anemometer

Building It

Unlike most of the interfaces in this book, the whole circuit is best located close to the sensor. The output from IC3 is a slowly changing one, making it suitable for transmission along a relatively lengthy line to a data input of the Decoder. The circuit requires three connections to the Decoder: the two power lines and a wire to a data input.

The circuit is to be housed in a weatherproof case. It may be convenient to mount IC1 on the outside of the case. Leads to this should be covered with melted paraffin wax or a weatherproofing or waterproofing compound to eliminate the possibility of short circuits.

The anemometer assembly is easy to build. Empty food cartons may be used for the cups and the cross-pieces can be made from stiff wire. The main essential is that the assembly rotates freely. Its diameter should be 30 cm or possibly a little more. The wheel which carries the magnet can be cut from wood, plastic or a large cork stopper. A suitable magnet is usually sold with the Hall Effect IC. Take care when mounting the magnet and the IC that there is no danger of the two coming into contact as the assembly rotates, particularly in high winds. On the other hand, the end of the magnet needs to pass within about 1 mm of the centre of the IC in order to generate a sufficiently great voltage difference.

Build the entire circuit, then connect it to a +5 V power supply for testing. Connect a voltmeter to one of the outputs of IC1. Rotate the anemometer slowly and note the rise or fall of voltage as the magnet passes the IC. Now transfer the voltmeter to the output of IC2. The output should be 0 V, but rises sharply to +5 V whenever the magnet passes IC1. If this does not happen, the polarity of the magnet is wrong; remove it from the wheel and replace it the other way round. Connect the voltmeter to the D output of IC3. The output should rise to "high", fall to "low" and rise to "high" once for every 16 rotations of the anemometer.

The output from D changes at a slow rate so the cable between the interface and the Decoder board can be several metres long. This allows the device to be mounted as high as

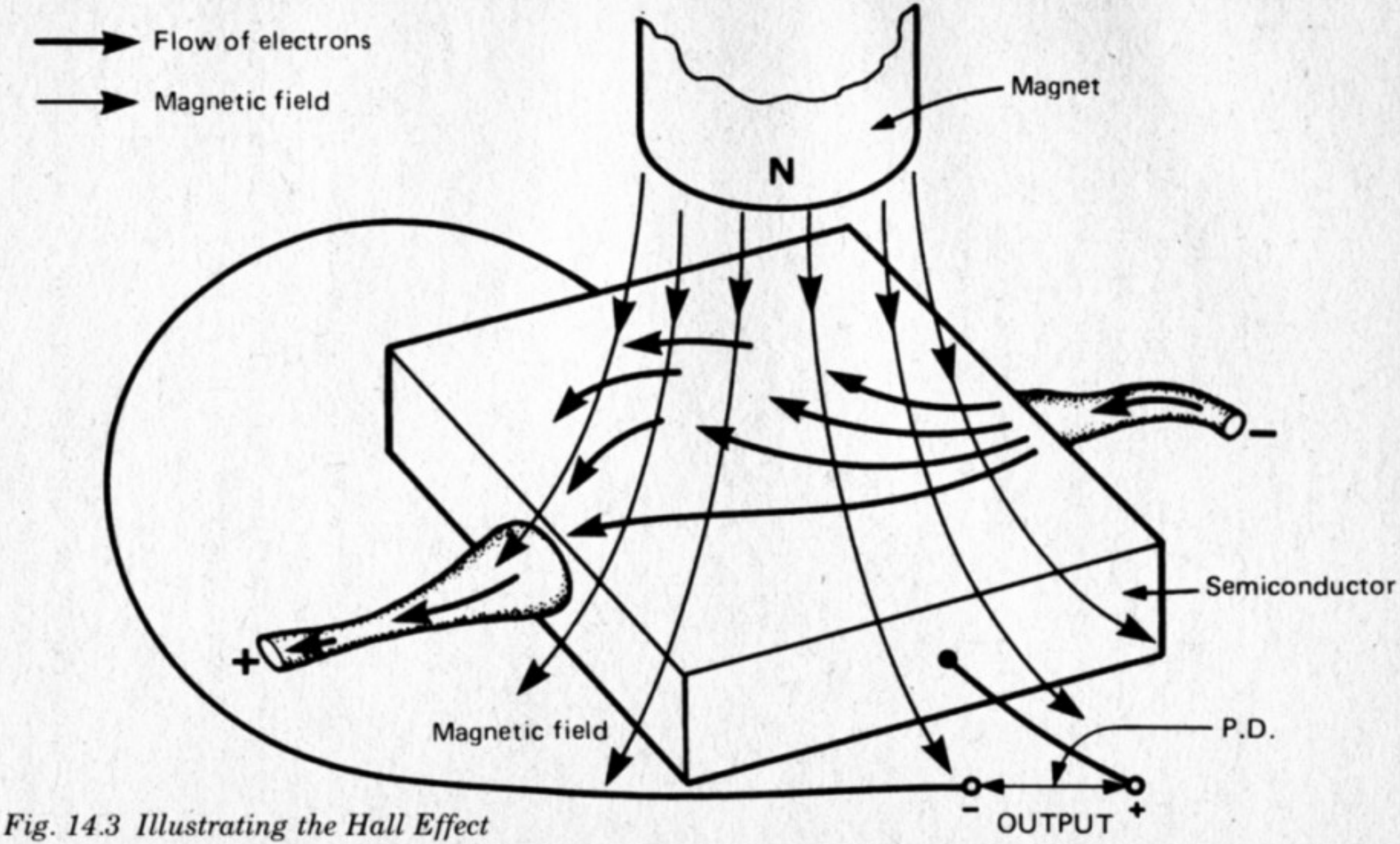


Fig. 14.3 Illustrating the Hall Effect

possible above ground so as to expose it to the full speed of the wind.

Programming

We require a program which measures the length of pulses coming from the output. The technique is to wait for the output from the device to become high, then to wait until it changes from high to low. Timing begins at that instant. The program then waits for the output to go low, and to go high again. The next time it changes from high to low, the elapsed time is recorded. This is the time for 1 pulse, i.e., 16 revolutions of the Anemometer. Here is the program for the VIC-20 with the Anemometer connected to line PB0, in group 1:

```
10 POKE 37138,240
20 POKE 37136,224
30 S=0:GOSUB 1000
40 S=1:GOSUB 1000
50 TI$="000000
60 S=0:GOSUB 1000
70 S=1:GOSUB 1000
80 T=TI/60
100 END
1000 R=PEEK(37136)AND1
1010 IF R=S THEN 1000
1020 RETURN
```

Subroutine 1000 is a loop which waits for the output of the Anemometer to take the value of variable S (0 or 1).

The program measures the time T in seconds for 16 revolutions. Assuming that the cups of the Anemometer are moving with the same speed as the wind, the wind speed is:

$$S = 3.619 \times r - T$$

where S is in kilometres per hour, r is the radius of the arms of the Anemometer in centimetres, and T is the time of 16 revolutions, obtained as above.

If the radius is 15 cm, the factor in the equation becomes

3.169 × 15, = 54.3. The program line to calculate wind speed is:

```
9Ø PRINTINT(54.3/T);"KM/H"
```

The corresponding equation for British units is:

$$S = 5.712 \times r - T$$

where S is in miles per hour, and r is the radius in inches. If the radius is 6 inches, the factor becomes 5.712 × 6, or 34.3. The program line is then:

```
9Ø PRINTINT(34.3/T);"MPH"
```

The Anemometer can easily cope with rotation rates of double the above, so can register winds of 100 km/h or more.

The program above may readily be adapted for the other two computers. With the Commodore 64, all that needs to be done is to substitute the correct addresses in lines 10, 20, and 1000.

The program for the BBC Microcomputer is:

```
1Ø A%=&97:X%=&62:Y%=24Ø:CALL&FFF4
2Ø X%=&6Ø:Y%=&224:CALL&FFF4:A%=&96
3Ø PROCwait (Ø)
4Ø PROCwait (1)
5Ø TIME=Ø
6Ø PROCwait (Ø)
7Ø PROCwait (1)
8Ø T=TIME/1ØØ
9Ø PRINT INT(54.3/T);" km/h"
1ØØ END
1ØØØ DEF PROCwait(S)
1Ø1Ø REPEAT R=(USR(&FFF4)AND&1ØØØØ)
/&1ØØØØ
1Ø2Ø UNTIL R=S
1Ø3Ø ENDPROC
```

Parts Required for the Anemometer

Resistor (carbon, 0.25 W, 5% tolerance)

R1, R2	2k2 (2 off)
R3, R4	10 k (2 off)
R5	6k8

Semiconductor

Q1	ZTX300 or similar npn transistor

Integrated Circuits

IC1	634SS2 Hall Effect IC (complete with magnet)
IC2	7611 CMOS operational amplifier
IC3	7493 4-bit binary counter/divider

Miscellaneous

Circuitboard
8-pin IC sockets (2 off, including one for IC1)
14-pin IC socket
1 mmterminal pins (7 off)
3-way socket to fit 3-way plug of Decoder board
Materials for making Anemometer assembly
Connecting wire

PROJECT 15

Thermometer

Although this project is part of the weather station series, it has many other applications. It may be used indoors to measure room temperatures, or perhaps the temperature of photographic solutions. It can used to detect excessive temperature as part of a domestic fire alarm system.

How It Works

The sensor in this device is a thermistor, the resistance of which varies significantly with temperature. The thermistor used in this circuit (Fig. 15.1) has what is known as negative temperature coefficient, which means that, over its usual working range, its resistance decreases as its temperature increases. An increasing temperature causes the resistance of the thermister to fall, resulting in a falling potential at point A and a falling analogue voltage at the input of IC2. The voltage varies *smoothly* over this range. We say it is an *analogue* quantity. But micros are not able to accept such an input. A micro understands only two kinds of input, "high" and "low". So, IC2 converts the smoothly varying analogue voltage into one which alternates between "high" and "low". This IC is driven by a clock, built up from two NAND gates of IC1. The clock pulses cause a voltage (V_{ramp}) inside IC2 to ramp down from about 3.75 V to 1.25 V. This actually happens in a series of 128 steps, but for all practical purposes it is a smooth ramp, as shown in Fig. 15.2. Whenever the input analogue voltage is higher than V_{ramp}, the output of the IC is "high". Otherwise it is "low". The result is that we get a waveform which has a fixed frequency ($\frac{1}{128}$ of the clock frequency), but has a varying mark-space ratio. As input voltage rises the "mark" (or "high" level) decreases, and the "space" (or "low" level) decreases. The micro is programmed to find out how much time is spent in the "high" or "low"

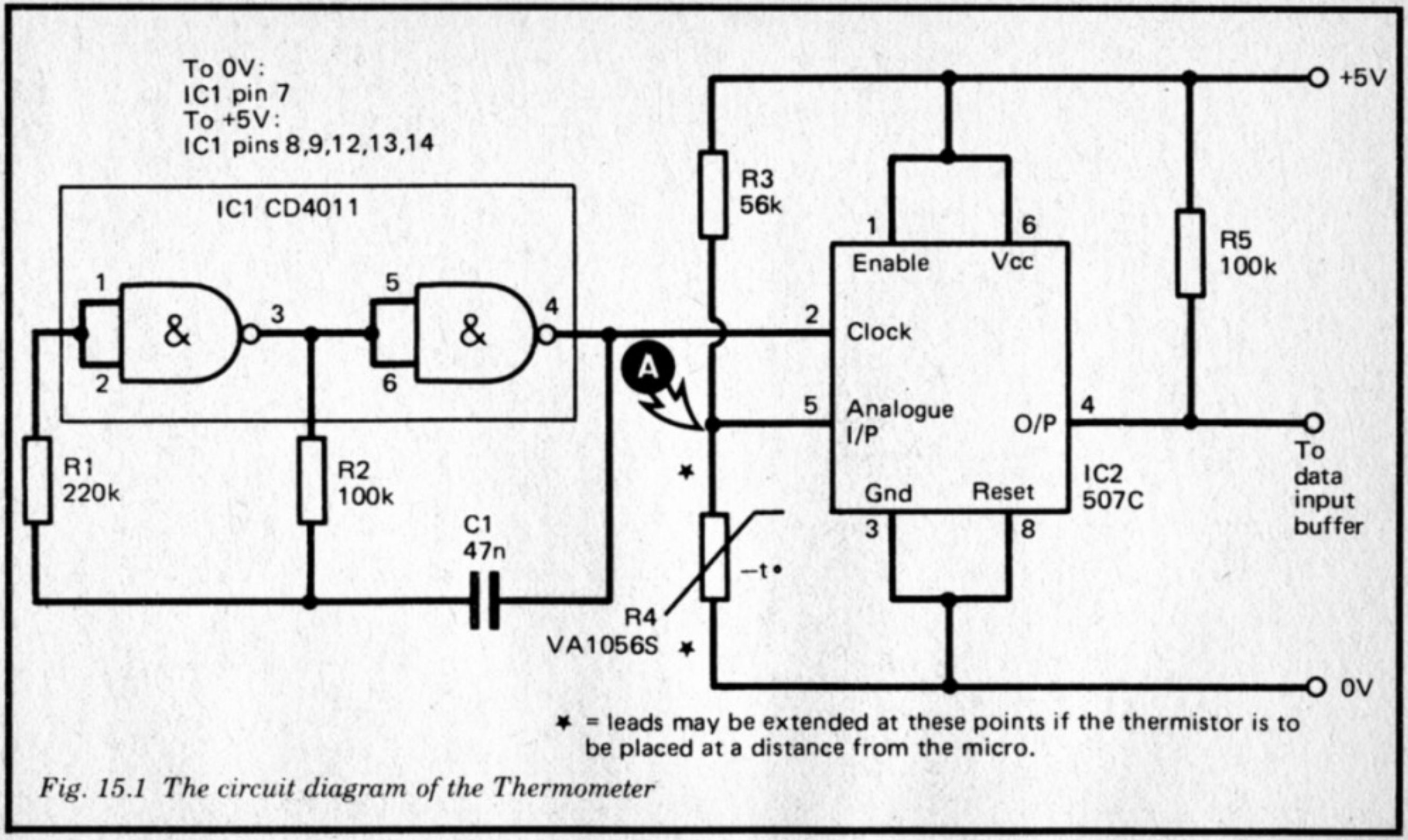


Fig. 15.1 The circuit diagram of the Thermometer

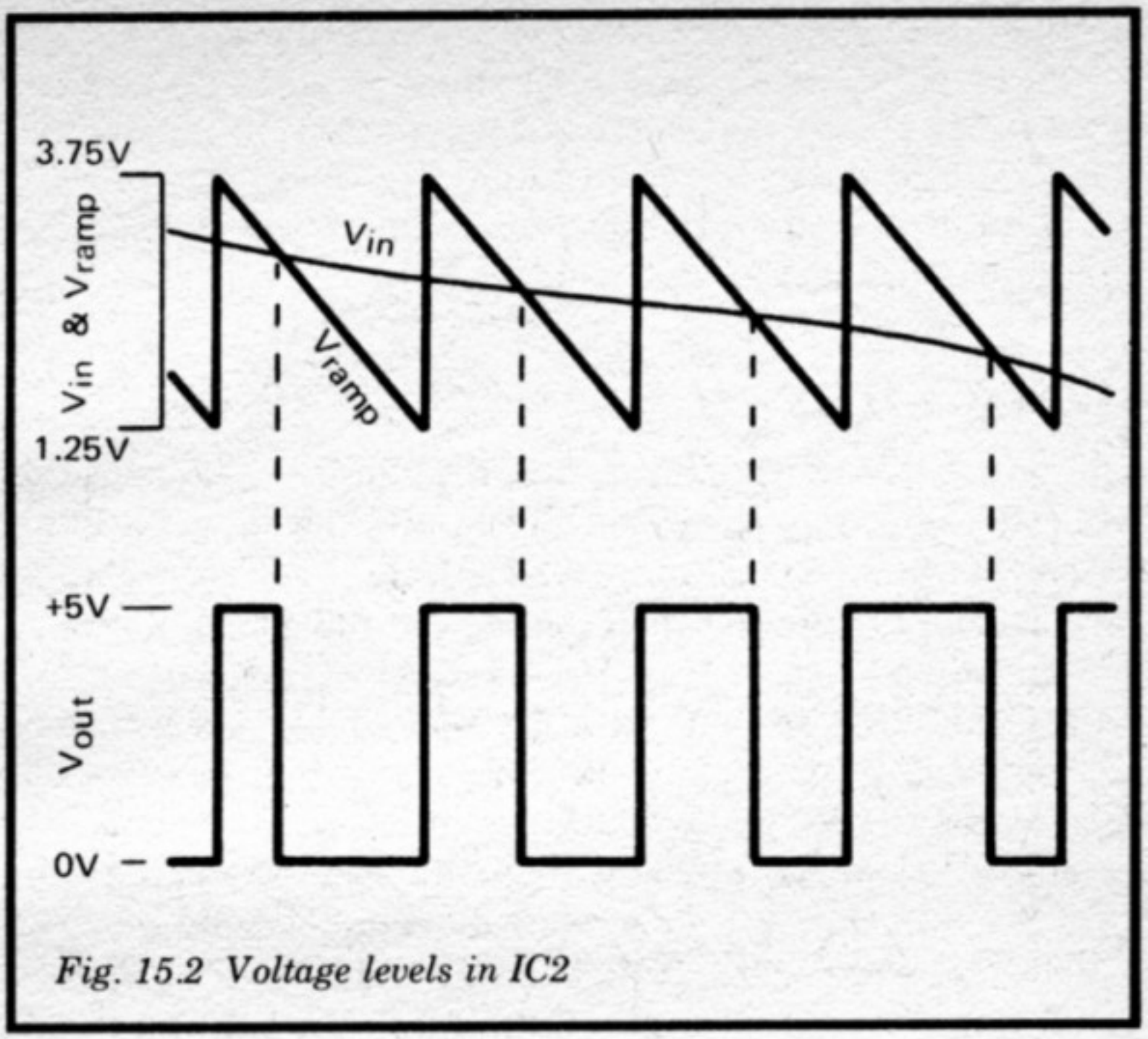


Fig. 15.2 Voltage levels in IC2

and, from this information it can work out the level of the input analogue voltage.

We call IC2 a *voltage-to-time converter*. It is not a true analogue-to-digital converter, for time, like voltage, is an analogue quantity. But time can be represented by "highs" and "lows" and in this form can be understood and measured by the micro.

IC2 produces digital signals which can be converted to analogue quantities by suitable software, as will be explained later. In effect, we have a *software* analogue-to-digital converter. Since the three computers in this book (except Model A of the BBC machine) all have analogue input terminals it might be thought that IC2 is unnecessary. With the VIC-20 and Commodore 64, the analogue inputs are suited only to receiving signals from games controls. But with the BBC Microcomputer Model B it is feasible to dispense with IC2 altogether. Point A of Fig. 15.1 can be con-

nected directly to one of the analogue inputs. Readings are taken using the ADVAL statement. The same remark applies to Projects 16 and 17. However, since the other weather projects need to be connected to the Decoder, it is generally more convenient to have these connected to the Decoder too, and to use IC2 as the analogue-to-digital converter.

Building It

Normally the thermistor is located outdoors. For use in weather recording, the best place is inside a meteorological screen. This shades it from direct sunlight yet allows air to circulate freely around it. If you do not have such a screen, mount it in a place where direct sunlight cannot reach it and where air circulation is good. It should be several centimetres away from any surface such as a wall or fence, otherwise it may take the temperature of the wall or fence, instead of that of the air. If it is in the open, protect it from rain, for this may cause a partial short circuit, and readings will be incorrect. One method of waterproofing is to melt some paraffin wax and to dip the thermometer in the wax to coat it. Another method is to mix a quantity of epoxy resin adhesive (e.g., Araldite) and coat it with this.

The rest of the circuit is built in a small case, located close to the micro. It needs three wires to connect it to the micro: the two power lines and a wire to a data input. The circuit does not require any address decoding except that provided by the Decoder.

If you have an oscilloscope, you can test the clock circuit and the output from IC2 after assembly. The clock runs at about 1 kHz. Its exact rate is not important. The output from IC2 has a frequency of about 8 Hz. As you alter the setting of RV1 the mark-space ratio varies from about 1:20 to about 20:1. If the output is continually low when RV1 is at one end of the track, the input voltage has fallen below 200 mV, which is the minimum required. This may happen if resistors are not quite of their specified values. To compensate, increase R4. If the output of IC2 stays "high" at the other end of the track, the analogue voltage is exceeding all values of

V_{ram} so increase the value of R3. Without an oscilloscope, the outputs can be detected by connecting a crystal earphone (e.g., from a tape recorder) to the output pins and the 0 V line, with a 100 nF (approx.) capacitor in series. You should hear a high-pitched note when it is connected to IC1 and a low buzzing sound when connected to IC2.

After completing construction, test for short circuits between the lines which are to be connected to the Decoder. Then plug the control into the Decoder. To test its output on the micro, you need a program which reads input repeatedly and displays it. You can try the effects of various temperatures by immersing the thermistor in warm water or in water containing ice cubes. In such extremes the output of IC2 should never be continuously "high" or "low". If this condition is found, replace R3 by a resistor of different value, or possibly wire an additional resistor between R4 (the thermistor) and the 0 V line. Exactly what you may need to do depends on the characteristics of the thermistor you are using and the range of temperature over which you require it to operate.

Calibration and Programming

The simplest method of programming is to read the output from the circuit many times and count how many "highs" are obtained. All that has to be done is to program a loop which repeats, say, 500 times, in which R is read repeatedly. Whenever R is *not* "0" a counter variable X is incremented (X=X+1). After the loop has been repeated 500 times, the value of R tells us how many times out of 500 the device had a high output. The ratio of X to 500−X is the proportion of time that the output spends in the high state. Measuring in this way takes a second or so to complete, but this is of no consequence in this application.

```
10 A%=&97:X%=&62:Y%=240:CALL&FFF4
20 REPEAT PROCtrigger(9)
30 TIME=0
40 IF TIME <10 THEN 40
```

```
5Ø PROCtrigger(1Ø)
6Ø X=Ø
7Ø FOR J=1 TO 5ØØ
8Ø PROCread(1,3)
9Ø IF R=1 THEN X=X+1
1ØØ NEXT
11Ø UNTIL X>42Ø
12Ø ... etc.
```

PROCtrigger and PROCread are defined on pp. 17–18.

Having obtained a value x which varies in proportion to temperature, the next step is to calibrate the Thermometer. Place the thermistor in a glass of water containing melting ice cubes. Stir occasionally, and allow at least 5 minutes for the thermistor to take up the temperature of the water. This is close enough to 0°C (32°F).

Run the program several times and find an average value for x. Results should be in the region of 20 to 50. If the count still seems to be decreasing, wait a while longer, for the thermistor has not cooled fully. If the ice cubes melt completely, replace them. Now repeat the procedure, but using a glass full of warm water at say 30°C (or say 90°F). Choose a temperature which is a little above the upper end of the range which you want to measure. You will need a thermometer to find out what the exact temperature is. Readings should now be at the upper end of the range, over 300 at least.

If the difference between the lowest and highest readings is too small you will not be able to obtain an accurate result. In this event you may need to replace R3, or wire in an additional resistor, as already mentioned.

Let us suppose that you obtained a reading of x_1 at 0°C and x_2 at T°C. The formula for converting the reading x at any temperature is:

$$t = (x - x_1) \times \frac{T}{x_2 - x_1}$$

For example, if the count is 40 at 0°C and 420 at 30°C, then a count of 150 is equivalent to a temperature of:

$$t = (150 - 40) \times \frac{30}{420 - 40}$$

$$= 90 \times 30/380 = 7.1°C$$

The same formula may be used on the Fahrenheit scale, except that having calculated t, you add 32° to it.

As well as reading temperature at a given instant, the micro can be programmed to take readings at regular intervals and record them in memory. They could be displayed daily or on demand and a table could be printed out. If the readings are taken frequently enough it is easy to program the micro to print out the maximum and minimum daily temperatures too.

Parts Required for the Thermometer

Resistors (carbon, 0.25 W, 5% tolerance)

R1	220 k
R2, R5	100 k (2 off)
R3	56 k
R4	VA1056S rod thermistor, or any negative temperature coefficient thermistor having resistance about 47 k at 25°C (disk and bead types equally suitable)

Capacitor

C1	47 n polyester

Integrated Circuits

IC1	CD4011 CMOS quadruple 2-input NAND gate
IC2	507C voltage-to-time converter

Miscellaneous

Suitable housing for thermistor
Case for circuit
Circuit board
1 mm terminal pins (5 off)
8-pin IC socket
14-pin IC socket
3-way socket to fit 3-way plug of Decoder board
Connecting wire

PROJECT 16

Barometer

A Barometer measures the pressure of the atmosphere. It is almost essential for forecasting the weather. The approach of depressions or anticyclones is announced by a fall or rise in pressure. This gives us advance warning of what kind of weather to expect. The rate of change of pressure, tells us how soon we may expect the weather to change. In addition, rapid changes of pressure mean strong winds, while slow changes or no changes at all are associated with calm weather.

Devices for measuring atmospheric pressure electronically are usually very expensive. This project is far cheaper and although it does not give an accurate reading, it is the *changes* which are important. It is much less important to know what the exact pressure is. So this device is a useful one for the forecaster.

How It Works

The Barometer is based on an extremely simple idea which is often featured in books of science projects for young people (Fig. 16.1). The inverted tube contains air. As atmospheric pressure increases, the air inside the tube becomes more compressed and the water rises up inside the tube. When pressure falls again, the air expands and the water level falls. Pressure is measured by measuring the level of the water. Unfortunately this apparatus has one big snag as a barometer. Air expands when heated. If the temperature increases, the water level falls, even though the atmospheric pressure may not have changed. It is possible to allow for the change in temperature by working out how much it would alter the volume of air. This is too much trouble to be bothered with for the simple Barometer of Fig. 16.1, but if we make a similar barometer and attach it to a micro, the micro

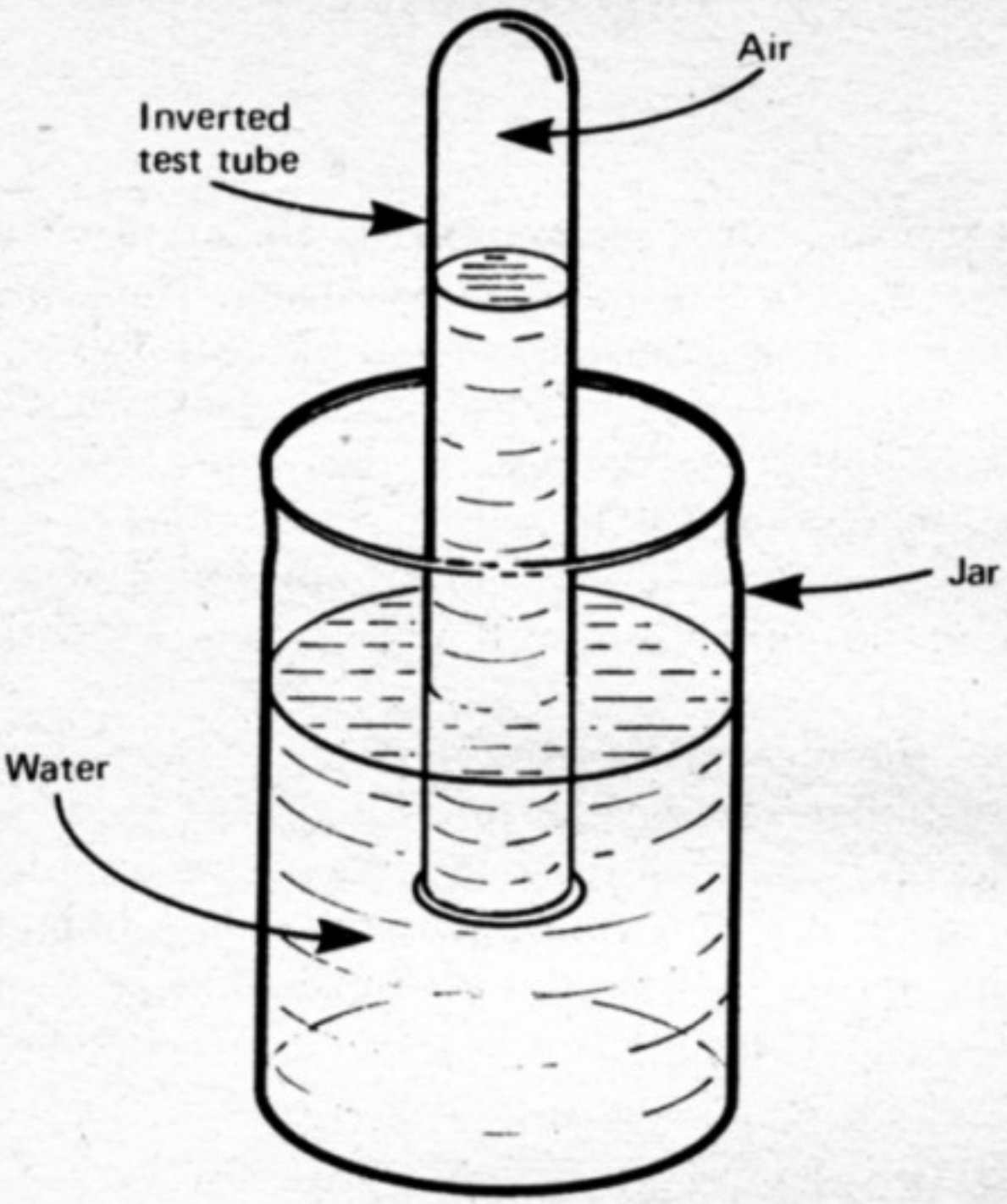


Fig. 16.1 A simple "water" Barometer

can do all the working out for us!

In order to compensate for changes in temperature, the micro must know what the temperature is in the region of the barometer. It can find out this if we have the thermometer (Project 15) operating at the same time. There is an equation which brings volume, temperature and pressure together:

$$\frac{\text{Pressure} \times \text{Volume}}{\text{Temperature}} = \text{constant}$$

Pressure and volume can be measured in any units we like (provided that we always keep to the same units), while temperature must be measured in kelvin. To work out any temperature in kelvin, just take the temperature on the Celsius scale (sometimes wrongly called the Centigrade scale) and add 273.

If we measure pressure, volume and temperature of the air inside the barometer on two separate occasions, and since the calculated result is a constant we can say:

$$\frac{\text{1st pressure} \times \text{1st volume}}{\text{1st temperature}} = \frac{\text{2nd pressure} \times \text{2nd volume}}{\text{2nd temperature}}$$

or, rearranging the above:

$$\text{2nd pressure} = \text{1st pressure} \times \frac{\text{1st vol.}}{\text{2nd vol.}} \times \frac{\text{2nd temp.}}{\text{1st temp.}}$$

We use an ordinary barometer to measure the 1st pressure when we take a reading with our home-made barometer. At the same time we measure 1st temperature and the 1st volume. Some time later we (or the micro) measure 2nd volume and 2nd temperature and can then calculate the 2nd pressure. Project 15 measures temperature, while this project measures volume. The micro then calculates the pressure.

In this project, the air is contained in a plastic or glass tube (Fig. 16.2). The top end of the tube has a tap to make it airtight. Its bottom end dips into a small jar containing water. The water has been made black by having had Indian ink added to it. The most extreme changes of pressure and temperature make the water level move over a range of about 2 cm.

In the region over which the water level will move, the tube is covered with black insulating tape, except for slits on opposite sides of the tube. Light from a small lamp passes through one slit, across the tube and out through the other slit. It is picked up by a light-dependent resistor (LDR). The resistance of this varies according to the brightness of the light. Since the water is black, it cuts off a varying amount of light, depending on its level. In this way, changes in water

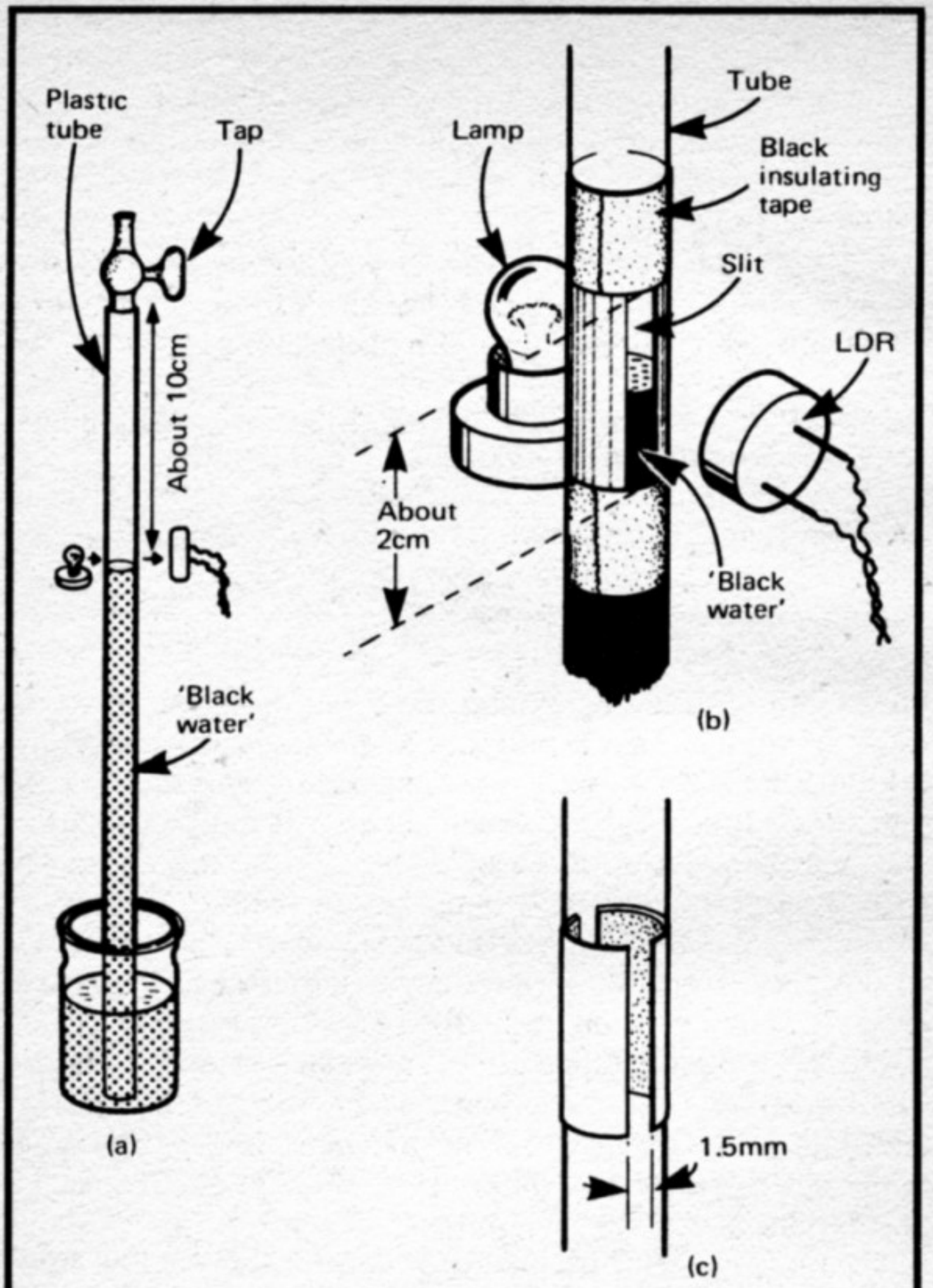


Fig. 16.2 The main features of the Barometer; (a) the whole barometer (masking on tube not shown); (b) how the tube is masked. There is an identical slit behind the tube to pass light coming from the lamp. Screening around the lamp and LDR not shown. (c) The first stage in masking

level are turned into changes in electrical resistance. In the prototype of this project the LDR resistance was 940 ohms when the water was at its lowest level but increased to 1700 ohms when the water had completely cut off the light. Changes in the resistance of the LDR (Fig. 16.3) affect the voltage at point A. This changing voltage is fed to IC2, which is the voltage-to-time converter. The way this works is explained on p. 137. The overall effect is that as pressure increases, the volume decreases, the voltage at A increases and the waveform from IC2 spends proportionally more time in the "high" state. If the micro is programmed to measure this time, the figure obtained is greater.

It is important to realise that the response of this device is far from linear. That is to say, equal increases in pressure do not bring about equal increases in the value found by the micro. However, the figure the micro obtains can be used to assess whether pressure has increased or decreased over the past hours or days, and whether it is changing slowly or quickly.

Building It

First of all build the barometer itself. There are several ways of doing this. The tube may be of glass, though clear plastic tubing such as is sold for aquarium aerating systems is very suitable. A tap to fit the tubing may also be obtained from a shop specialising in aquaria. The tube may be held straight by stapling it to a strip of wood. This is mounted vertically on a wooden base. The water container can be a small plastic or glass bottle such as is used to hold medicine tablets.

Prepare two strips of black insulating tape about 2 cm long and wide enough to leave two gaps about 1.5 mm wide when they are fixed on the tube (Fig. 16.2c). Wind two other strips around the tube about 2 cm apart, as in Fig. 16.2b. This leaves two slits about 2 cm long and 1.5 mm wide on opposite sides of the tube. Measure the length of the slit and the distance between the tap and the top of the slit to the nearest millimetre. Mount a small filament lamp to one side of the tube, about 1 cm from it. The lamp should be housed in

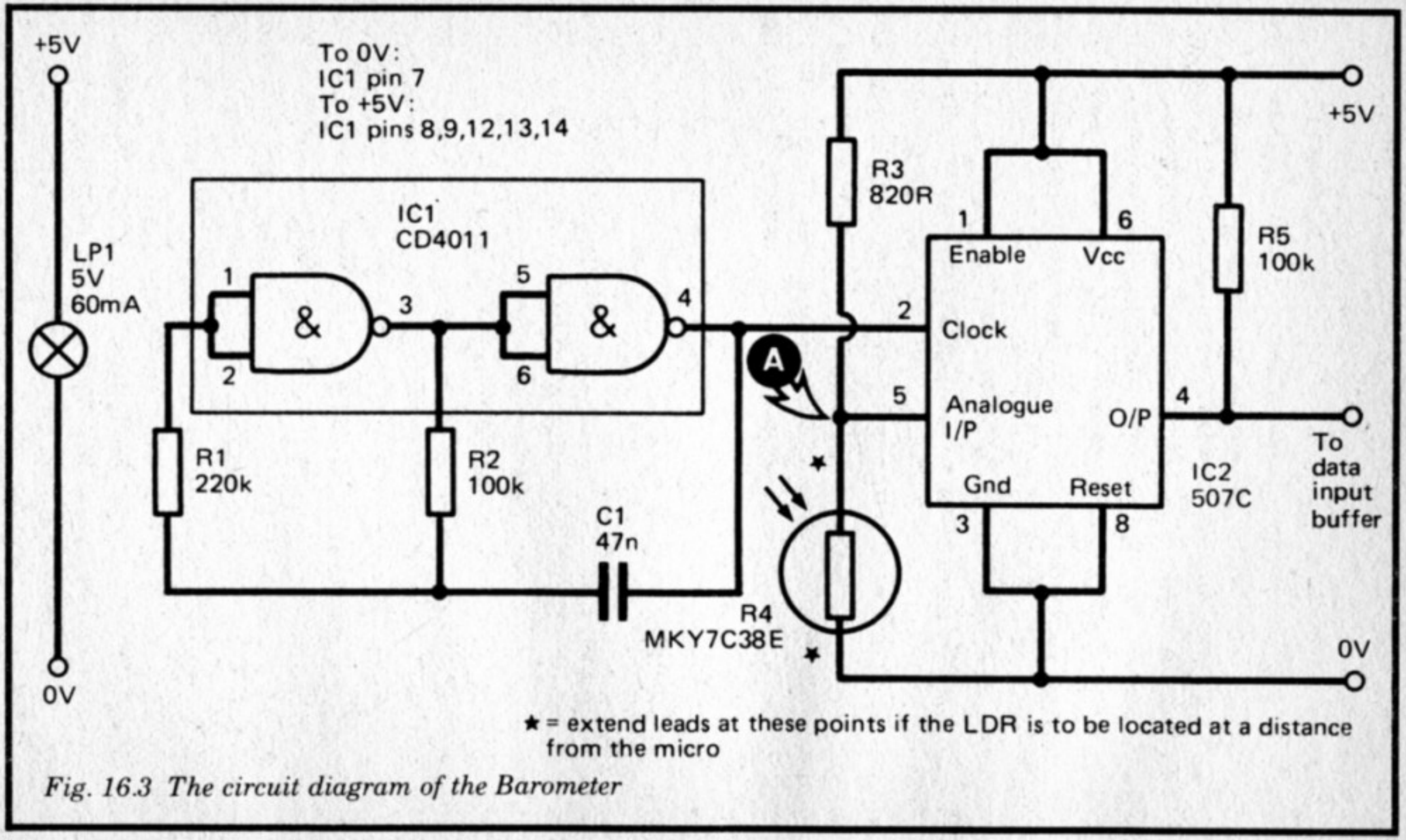


Fig. 16.3 The circuit diagram of the Barometer

a small box, made from thin white card, to keep light from reaching the LDR directly, and to reflect as much light as possible towards the slit. On the other side the LDR is enclosed in a similar box made of thin card. It is better if the whole assembly is housed in a lightproof case to prevent daylight from interfering with the readings.

The electronic side of this device requires three connections to the Decoder board: the two power lines and a wire to a data input. If the barometer is to be placed at some distance from the micro, the leads may be extended at the points marked * in Fig. 16.3. The circuit itself must be reasonably close to the micro. In practice there is much to be said for having the entire apparatus indoors, for atmospheric pressure is more or less the same indoors and out. If it is placed indoors, in a room which is kept at a reasonably steady temperature, there is no need to correct for changes of temperature, and programming is simplified.

When construction is complete, test the leads going to the Decoder to make sure that there are no short circuits between them. Then plug the barometer into the Decoder board and switch on the power. Since this device uses the same converter IC as Project 15, a similar program may be used in testing.

```
10 POKE 37138,240
20 POKE 37136,224
30 FOR J=1 TO 1000
40 R=PEEK(37136)AND2
50 IF R=2 THEN X=X+1
60 NEXT
70 PRINT X
80 X=0:GOTO30
```

Run the program with the tube empty. The result should be reasonably low, perhaps between 200 and 500, but the value obtained depends very much on the type of LDR used, and many other features of construction. However, it should not vary by more than about 2 or 3 each time the program repeats. Take an average of 20 such readings; these are the scale minimum readings.

Now make a mixture of 3 parts of water with 1 part of black ink (Indian ink preferred). Add one drop of washing-up detergent. Pour the "black water" into the jar. Open the tap and slowly suck the water up the tube. Take care, or you may find yourself sucking a mouthful of black water! Suck the water up to a level just above the taped region, so that the slit is completely blacked out. Now run the program again. This gives a series of readings higher than before. The values obtained should not vary by more than 2 or 3. They should be at least 50 or 60 greater than the values obtained with the empty tube. If the increase is smaller than this, try replacing R3 with a resistor of lower value. When you have settled on a satisfactory value, take the average of 10 such values; this is the scale maximum reading.

Now open the tap slightly and let the water run down until its level is about half-way along the slit. Allow the tube to stand for a minute to let water drain down from the walls of the tube above.

If the barometer is to be kept in a fairly constant temperature, it is now ready for programming. Then run the program and take an average of half-a-dozen results. This is the initial reading. At the same time, measure the temperature and pressure (use an ordinary aneroid or Fortin barometer). You now have several figures ready for the final programming (Fig. 16.4):

MAX	Maximum scale	values obtained from the test program
MIN	Minimum scale	
INIT	Initial reading	
P	1st pressure – in millibars, etc.	
T	1st temperature – in kelvin (Celsius + 273)	
LT	Length of tube above slit (in mm)	
LS	Length of slit (in mm)	

Programming

The first thing to work out is the value we are to use in the program to represent the initial position of the water level, as measured from the top of the tube. This is what we have

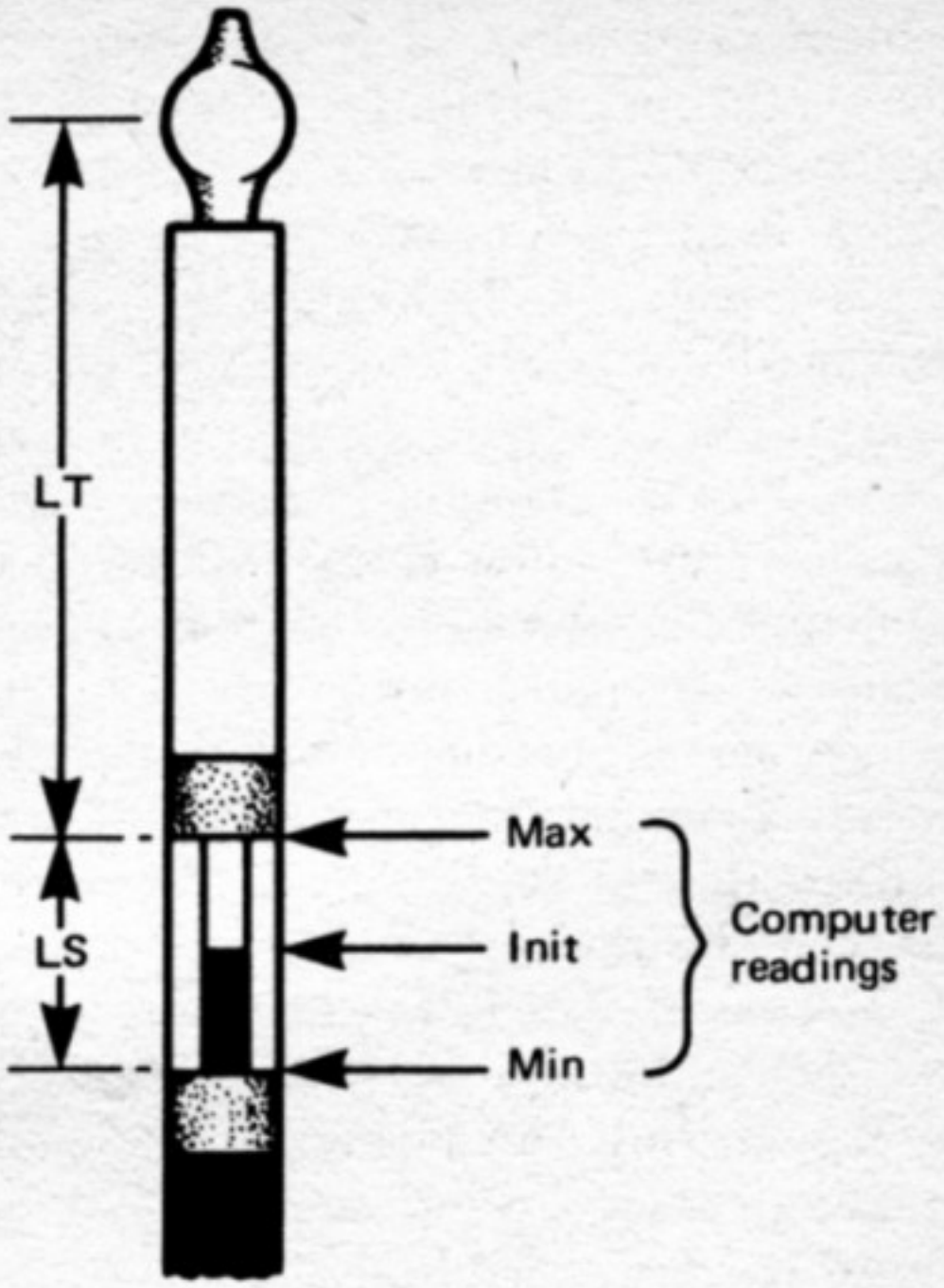


Fig. 16.4 The readings needed to calibrate the Barometer

called the "1st volume". It is the volume with which all later readings of volume are to be compared. Using the abbreviations listed at the end of the previous section, the formula for this is:

$$\text{1st volume} = \text{LT} + \frac{(\text{MAX} - \text{INIT}) \times \text{LS}}{\text{MAX} - \text{MIN}}$$

For example, if the tube is 100 mm long, the slit 20 mm long, MAX count is 270, MIN count is 200 and INIT count is 250, then we work out:

$$\text{1st volume} = 100 + \frac{(270 - 250) \times 20}{270 - 200} = 106 \text{ (rounded off)}$$

This value is used in all later calculations until, perhaps you find that the water has dried out and needs replacing, in which event you start again with new measurements and a new value for 1st volume.

The same formula applies for calculating the volume at other times, except that you use the actual count obtained in place of INIT. For example, if the reading (x) is 230, the computer calculates the new volume as 111, using the line:

```
6Ø new = 1ØØ + (27Ø-x)*2Ø/7Ø
```

We now have a new volume (the same thing as the 2nd volume referred to on p. 145), which is used to work out the new pressure. Actually, if you are not bothering to correct for temperature there is no need to go any further. You can use the new volume figures directly. If pressure is rising, these figures show a decrease, and conversely, if pressure is falling, the figures rise. You then discover, by experience, just how rapid the rise or fall has to be to predict a change in the weather.

If you want to convert the new volume into a pressure reading, the program line is:

```
7Ø pressure = 1Ø6*P/new
```

where P is the initial pressure. The figure 106 refers to the 1st volume as already calculated. Your 1st volume will probably require a different figure to be substituted here. The value of "pressure" can then be displayed, and its units are the same as those used when you read the aneroid or Fortin barometer to begin with.

If you want to take temperature into account, you first need to program the micro to read this, using Project 15. Let us assume that you have done this and the temperature in degrees Celsius is held in the variable "temp". The *un*converted pressure is the variable "pressure" calculated as already described. The line for conversion is:

```
8Ø pressure=pressure*T/(temp+273)
```

T is the initial temperature (p. 150).

If you want extra precision, you should allow for the fact

that the water vapour in the tube also exerts its own pressure, depending on temperature. An approximate correction for this, over the range that the barometer is likely to encounter can be added to the line above:

```
8Ø pressure=pressure*T/temp+273-(temp+1Ø)/27
```

This gives you a reading of pressure with as much exactness as the barometer is capable. You can use it for keeping records. Check it against a proper barometer, or against a recent weather map sometimes, as factors within the apparatus may change and you may need to alter some of the constants of the program. Certainly the water will need topping up occasionally, and eventually need renewing, when many of the constants will need recalculating. Thus it is better to regard this as an experimental project, suited for detecting short-term pressure changes and forecasting the weather accordingly.

Parts Required for the Barometer

Resistors (carbon, 0.25 W, 5% tolerance)

R1	220 k
R2	100 k
R3	820 R
R4	MKY7C38E light-dependent resistor (or similar type such as ORP12)
R5	100 k

Capacitor

C1	47 n polyester

Integrated Circuits

IC1	CD4011 CMOS quadruple 2-input NAND gates
IC2	507C voltage-to-time converter

Miscellaneous

LP1	5 V, 60 mA filament lamp, wire ended (socket required if wire-ended type not used)

Circuit board
8-pin IC socket
14-pin IC socket
3-way socket to fit 3-way plug of Decoder
1 mm terminal pins (5 off)
Plastic tubing about 20 cm long, about 5 mm external diameter (e.g., aquarium tubing, but must be transparent with glass-clear walls)
Tap to fit tubing (aquarium aerator line tap),or screw clip black PVC insulating tape
Small glass or plastic jar, bottle or medicine tube
Other materials for building the assembly

PROJECT 17

Sunshine Recorder

We finish this series of weather station projects and the book itself in an optimistic mood! This project measures the amount of light energy arriving from the Sun over a given period of time. It is an intriguing circuit, which means that instead of telling you how much energy is arriving at any given moment, it tells you how much has arrived since you last set it. Thus the micro has only to take a reading from time to time. The fact clouds may pass across the Sun or the shadow of a tree may pass across the garden while the micro is otherwise engaged, will be taken account of when the next reading is made.

How It Works

This is one of the more complicated projects, but certain features of it will be familiar to you if you have already built some of the other projects. Sunlight falls on a photovoltaic cell (B1, Fig. 17.1). This is a silicon cell of the type often referred to as a *solar cell*. This produces a voltage of approximately 0.45 V when fully illuminated, the voltage depending on the amount of sunlight reaching it. The voltage is fed to an operational amplifier (IC1) which is wired as an *adder*. Its function is to add the voltage from the photocell to the voltage from the potential divider RV1. The reason for this will be explained in a moment.

The output of IC1 goes to a transistor Q1. This is acting as a constant-current device which is charging capacitor C1. We rely on the fact that, for the given base current, the collector current is relatively unaffected by the PD between the collector and emitter. As C1 charges, the potential at point A rises. The result is that the PD between the collector and emitter of Q1 decreases. However, until this PD falls much lower than we ever allow it to fall, the reduction of PD

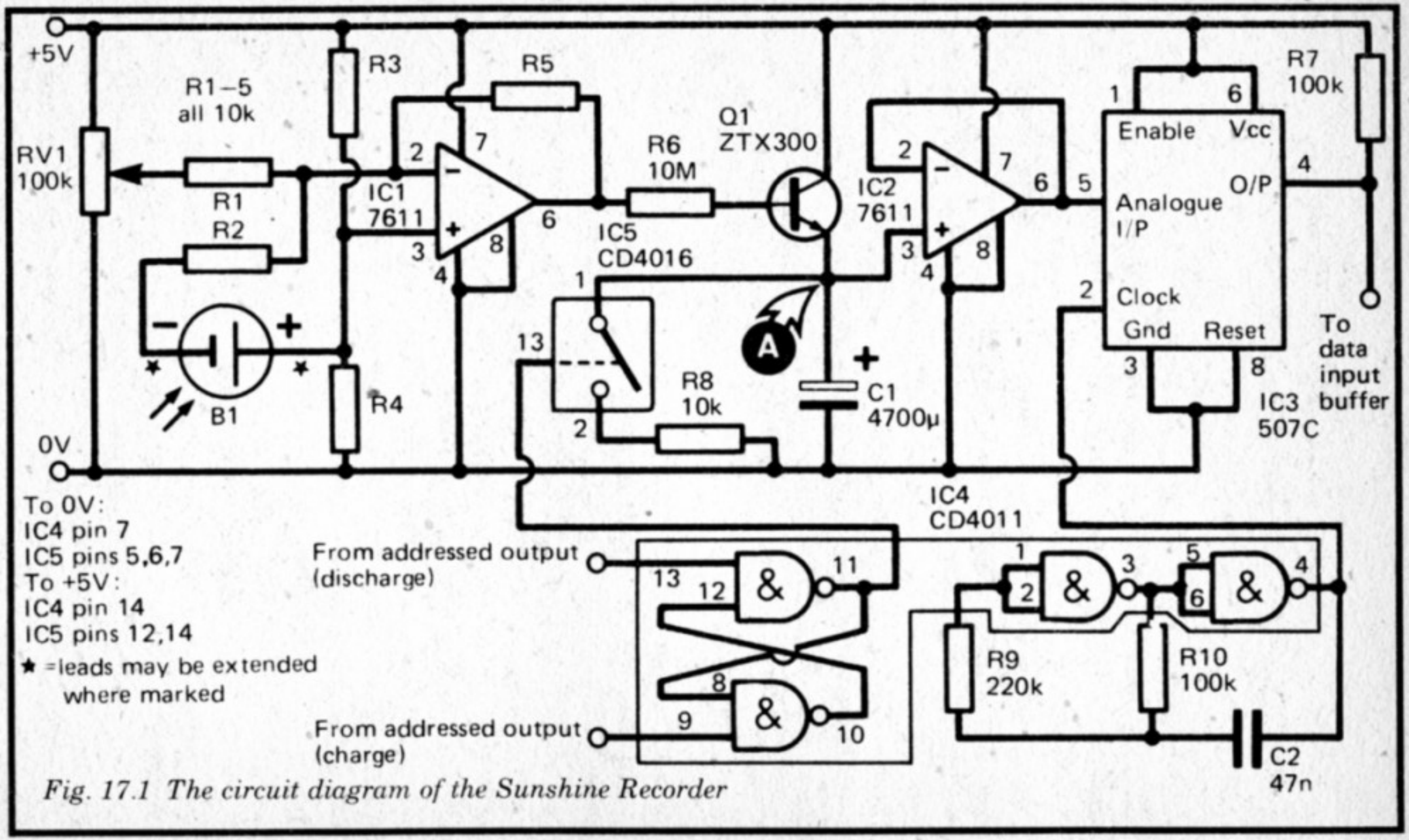


Fig. 17.1 The circuit diagram of the Sunshine Recorder

does not reduce the current flowing into C1.

The amount of current flowing depends on the base current flowing through to Q1 from IC1 and passing through R6. The value of R6 is high, to keep this current very small. Until the potential at the output of IC1 exceeds about 0.6 V there is no base current at all and Q1 is switched off, but, in dim light the PD across B1 may be *less* than 0.6 V. We have to ensure that as soon as *any PD* is developed across B1 a current begins to flow to Q1. This is the reason for the adder. The output of IC1 is the sum of the two inputs. RV1 is set to deliver an input of 0.6 V, so that any additional input due to B1 goes fully towards driving Q1.

Actually this is an inverting adder, so we apply 1.9 V from RV1 (this is 0.6 V less than the half-way voltage at pin 3 of IC1) and the cell is connected with its negative terminal to R2. This results in a positive output to Q1.

Over a period of several tens of minutes, or perhaps an hour, C1 charges at a rate depending on the amount of sunlight from moment to moment. The PD across C1 rises, faster when the Sun shines strongly, more slowly when a cloud covers it or at the end of the day, and not at all during the night. The PD is fed to another operational amplifier, IC2, which is wired as a *unity gain voltage follower*. This is used as a buffer between C1 and the voltage-to-time converter, IC3. The input of IC3 has an impedance of about 100 k, which would allow current to leak away from C1 almost as quickly as it was arriving from Q1. But the input impedance of IC2, which is a CMOS IC, is extremely high (about 1 Teraohm, or 1×10^{12} ohms!). Leakage to this input is virtually non-existent. The output of IC2 follows the PD which is across C1 exactly. As explained in Project 13 (p. 137), IC3 converts this voltage into a waveform. The shape of this depends on the level of the voltage.

For IC3 to operate properly the input voltage must be in the range 1.25 V to 3.75 V. During a period of measurements the PD should start at a little above 1.25 V and must not exceed 3.75 V before the reading is taken.

The micro is given the task of adjusting the PD across C1 to a little more than 1.72 V each time a measurement period

is to begin. It discharges C1 through resistor R8, which is switched into circuit by a CMOS switch (IC5). This is controlled by a flip-flop made from two gates of IC4. The other two gates of this IC make up the clock for IC3 (*see* p. 135). The flip-flop is controlled by two addressed outputs. Addressing one of these sets the flip-flop, which closes the CMOS switch and begins to discharge C1. The micro checks the PD by reading the data input. When the PD has reached a suitably low level, the micro addresses the other addressed output. This action resets the flip-flop, which opens the CMOS switch and C1 begins to charge again. It requires a "high" level on pin 13 of IC5 to close the switch.

The length of each measurement period is fixed so that, even with continuously brilliant sunshine, the PD across C1 never exceeds 3.75 V.

Building It

The project requires 5 lines to the Decoder: two power lines, a wire to a data input, and two lines from addressed outputs. If the solar cell (B1) is to be mounted away from the micro, the leads may be extended where marked * in Fig. 17.1. The solar cell need not be a large one, in fact a small one is better, for it delivers less current and so allows longer measurement periods. It is often possible to buy small fragments of solar cells very cheaply from hobby suppliers. One of these may well prove to be suitable. This is a reminder that solar cells are easily broken, unless you have one that is in a protective case. It must be mounted where it is protected from the weather and physical damage yet receives full sunshine for as much as possible of the day. The cell must be horizontal, facing upwards. Fig. 17.2 shows one way of protecting it. If, later, you find that the capacitor charges too rapidly, a layer or two of white paper may be fixed inside the cover to reduce the amount of light reaching the cell. Remember to clear away any debris which may fall on the cover, especially leaves in autumn.

Wire up IC1 and Q1 and their associated components first. Do not include the connection to IC5 at this stage. The

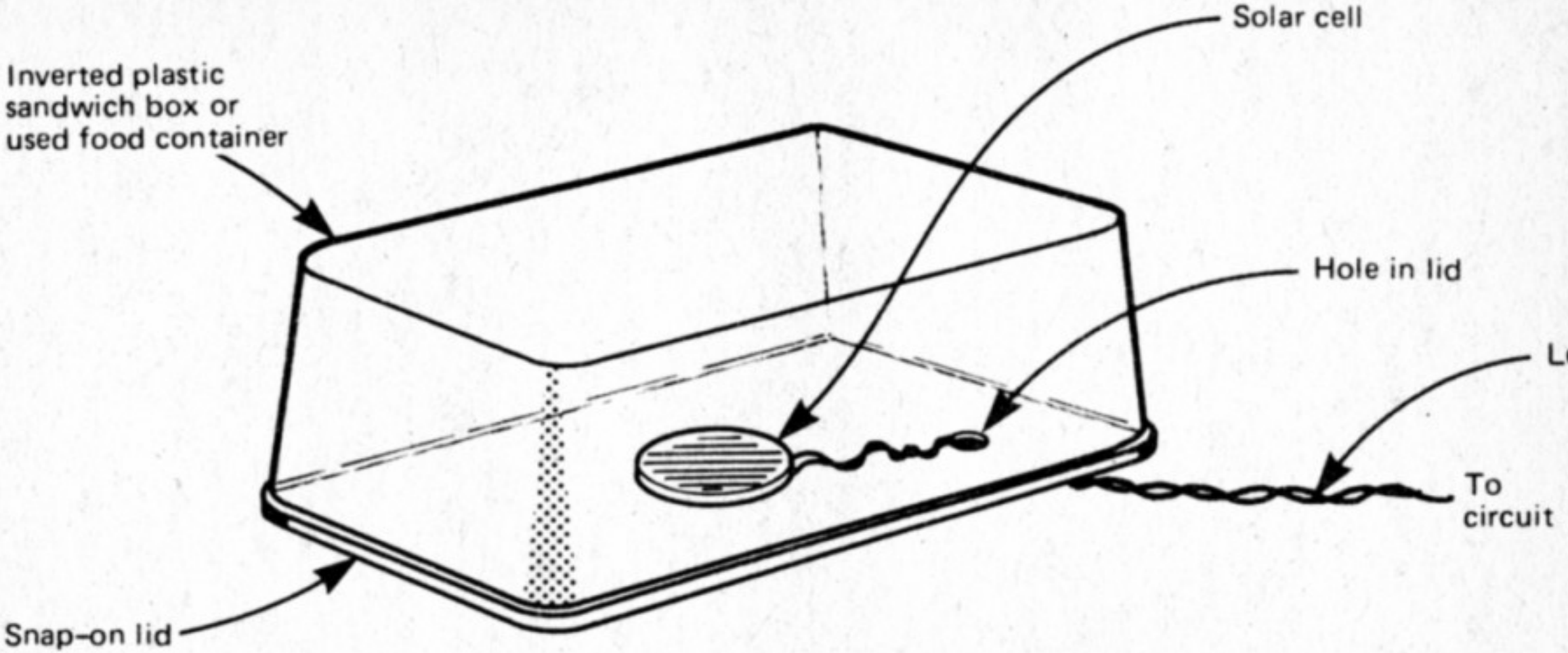


Fig. 17.2 Mounting and protecting the solar cell. The plastic should be clear, or white and translucent. The cell may be held in place by adhesive or by a double-sided "sticky fixer"

circuit may be tested by connecting it to a +5 V or +6 V supply. Connect a voltmeter across C1 (at point A). Shade B1 so that no light reaches it. Alter the setting of RV1 several times. In some positions you will see the PD across C1 slowly increasing. It should take several minutes to increase by as much as 0.1 V. Adjust RV1 so that the PD *just* does *not* rise. This is the point at which it is delivering approximately 1.9 V. You would need to wait for, say, 10 minutes to make certain that no rise in PD is occurring, but for the present be satisfied if it does not apparently change in one or two minutes. Now uncover B1. The needles should start to move, showing a rising PD. The rate of rise depends on the amount of light. Let bright sunshine fall on it, or hold a table lamp a few centimetres away from it and the rate of rise should increase perceptibly.

Next add IC2, IC3 and IC4 to the circuit. When tested as above, with a voltmeter connected to the output of IC2 (pin 6), the same result should be obtained. With an oscilloscope connected to the output of IC3, you see a waveform similar to that of Fig. 15.2. The proportion of time spent in the "high" state is gradually reduced as C1 charges.

Finally, wire up the flip-flop of IC4 and the single CMOS switch of IC5. Note that the control inputs of the other switches need to be connected to either 0 V or +5 V (*see* list of connections on Fig. 17.1). The flip-flop may be controlled by wiring its inputs to +5 V temporarily, then briefly connecting either one or the other to 0 V instead. As the flip-flop changes state the switch is turned off or on. When the switch is on, C1 discharges.

The capacitor discharges much more rapidly than it charges, but still takes several seconds. It must not discharge too readily or the micro will be unable to stop the discharge at the correct level.

Before connecting the completed device to the Decoder, test all the leads to ensure that there are no short circuits between them. Plug the device into the Decoder and switch on the power. Allow C1 to charge until the PD across it is between 1.25 V and 3.75 V, as measured by a voltmeter.

Use a program like that on p. 139 to read the input from

the voltage-to-time converter. Cover the photo-cell to exclude all light and set RV1 so that the reading obtained by the micro does not decrease over a long period, say 10 minutes. Of course, individual results may vary by 2 or 3 counts, but the average should remain unchanged. The setting should be such that any slight movement of the wiper of RV1 towards the 0 V end of its track, causes the capacitor to begin charging, and the figure displayed by the micro to fall. Now uncover the photo-cell. The reading should begin to fall immediately, and its rate of fall should vary with the amount of light reaching the photo-cell.

Discharge C1 until the PD across it is a little more than 1.25 V. You can do this by temporarily connecting a 10 k resistor across it. When the capacitor is discharged to the correct level, run the program again several times. The reading should be at the high end of the range. Select a value as the "discharge level". The program for discharging the capacitor requires two output addresses. Supposing that these are 9 for discharging the capacitor and 10 for charging (i.e., stopping discharge), and supposing that the value for the "discharge level" is 420, the following program will control the process:

```
10 POKE 37138,240
20 POKE 37136,144:POKE 37136,0
30 TI$="000000"
40 IF TI<6 THEN 40
50 POKE 37136,160:POKE 37136,0
60 X=0:POKE 37136,224
70 FOR J=1 TO500
80 R=PEEK(37136)AND8
90 IF R=1 THEN X=X+1
100 NEXT
110 IF X<420 THEN 20
120 ... etc.
```

The program assumes that the Recorder sends its output to line PB3. Lines 20 to 50 turn on the CMOS switch for a tenth of a second. Lines 60 to 110 read the voltage to find if it has fallen to the "discharge level". If it has not, the process is

repeated. When it has reached "zero level", the program drops through to line 120. Here, the micro goes into some other routines while the capacitor charges again at a rate depending on the amount of sunlight. These routines must include some kind of timing sub-routine so that, say, 30 minutes later, the micro returns to read the value shown by the sunshine recorder. This is proportional to the amount of sunlight which has reached the photo-cell during the measurement period. This value may be displayed, or stored in memory to be added to values obtained throughout the day.

The program for the Commodore 64 is the same as the above, except for the two addresses (*see* p.16). The program for the BBC Microcomputer is:

```
10 A%=&97:X%=&62:Y%=240:CALL&FFF4
20 REPEAT PROCtrigger(9)
30 TIME=0
40 IF TIME(10) THEN 40
50 PROCtrigger(10)
60 X=0
70 FOR J=1 TO500
80 PROCread(1,3)
90 IF R=1 THEN X=X+1
100 NEXT
110 UNTIL X>420
120 ... etc.
```

PROCtrigger and PROCread are defined as on pp. 17–18.

Parts Required for the Sunshine Recorder

Resistors (carbon, 0.25 W, 5% tolerance, unless otherwise specified)

R1–R5	10 k, 1% tolerance (5 off)
R6	10 M
R7, R10	100 k (2 off)
R8	10 k
R9	220 k
RV1	100 k, miniature preset resistor

Capacitors

C1	4700 μ electrolytic
C1	47 n polyester

Semiconductor

Q1	ZTX300 or similar npn transistor

Integrated Circuit

IC1, IC2	7611 CMOS operational amplifiers (2 off)
IC3	507C voltage-to-time converter
IC4	CD4011 CMOS quadruple 2-input NAND gate
IC5	CD4016 CMOS quadruple analogue switch

Miscellaneous

B1	Silicon photo-voltaic cell (solar cell), small

8-pin IC sockets (3 off)
14-pin IC sockets (2 off)
3-way sockets to fit 3-way plugs on Decoder board (3off)
Circuit board
1 mm terminal pins (7off)
Materials for making protective case and for mounting the cell

APPENDIX A

The Decoder

It is essential for you to make this, for you need it when you come to connect the projects to the micro. It does several things beside decode addresses. We will explain how it works in some detail, but if you would rather go ahead and build it, skip the description that follows.

Fig. D.0 is the circuit diagram. Lines coming from the left side of the diagram are the 8 lines of port B of the micro's user port, and the 0 V or ground line of the micro's own power supply. Appendix B describes a special version of the Decoder for the Electron which can be plugged directly on to the edge connector at the rear of the machine. Except when it is used with the BBC Microcomputer, the Decoder has its own power supply (Figs. D.2 and D.3) which also supplies power to add-ons connected to the Decoder. The only exception is when an add-on requires a voltage higher than +5 V (e.g., the Model Controller), in which case an independent power supply is required. One essential point is that there *must* be a connection between the 0 V line of the micro and the 0 V line of the Decoder, as shown in Fig. D.0. There is more about power supplies later.

How It Works

The Decoder provides three kinds of connection for add-ons:

(1) Addressed outputs
(2) Data inputs (one-bit)
(3) 4-bit data inputs

The addressed outputs come from IC4. This has 16 such outputs and the logic of this chip is such that at any one time all outputs are "high" except for one. This one "low" output is on the line which is currently being addressed. Addressing is done by putting "high" and "low" levels on the four "select"

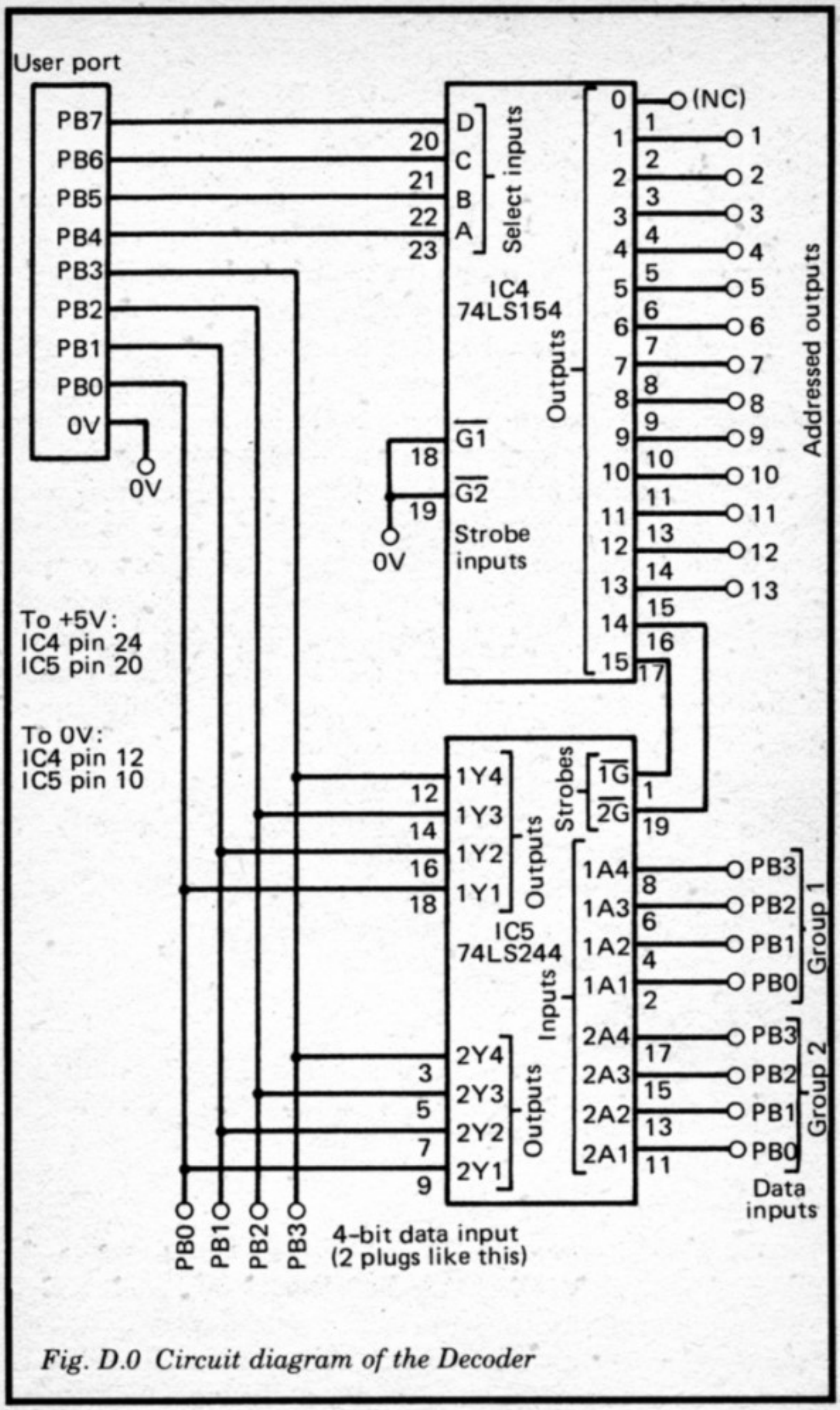


Fig. D.0 Circuit diagram of the Decoder

inputs of IC4. These four inputs are treated as if they hold a 4-bit binary number. When all inputs are "low", the equivalent number is "0000", or zero. In this condition, a "low" output appears at output "0" (pin 1) while all other outputs are "high". If we put a "high" level on inputs A and C, and a "low" level on inputs B and D, the equivalent binary number is "0101", which is the same a"5" in decimal. Immediately, output 5 (pin 6) goes "low", and all other outputs are "high".

The action of IC4 is controlled also by the two "strobe" inputs, G1 and G2. When either one of these is "high", all outputs of IC4 go "high". In this circuit we hold these inputs permanently low by connecting them to the 0 V line. This means that there is always one low output from the IC.

Outputs are used to trigger add-ons into action. The "select" inputs are controlled by lines PB4 to PB7 of port B. To make output 5 go "low", for example, we need to put "0101" on the top four lines of the port. This is equivalent to putting "0101 0000" on the port as a whole. The value of "0101 0000" is 80 in decimal. The rule is that to make a given addressed output N go low, we need to place 16 times N on the lines of port B (*see* Table 0.2).

Except when we are commanding Project 4 (Model Controller) which needs to use lines PB0 to PB3 as *outputs*, these lower four lines are defined as inputs. Any device connected to these lines *must* do so through a three-state buffer. Projects such as the Picture Digitiser and the Weathercock, which have 3 or 4 bits of data to send simultaneously, have the required buffers on their circuit board. Lines from the outputs of these are plugged directly into the data lines at the 4-bit data input plugs. These projects also need to be connected to an addressed output, which is used to enable the buffer by being made "low". Since only one output of IC4 can be "low" at any one instant, it ensures that only one set of buffers can be enabled at a time.

It is uneconomical for the other projects to each have their own buffer IC, since they only have one bit to send. IC5 provides eight buffers on the Decoder board. These may be used by any eight add-ons with one-bit data outputs. These buffers are in two groups of four, all buffers of a group being en-

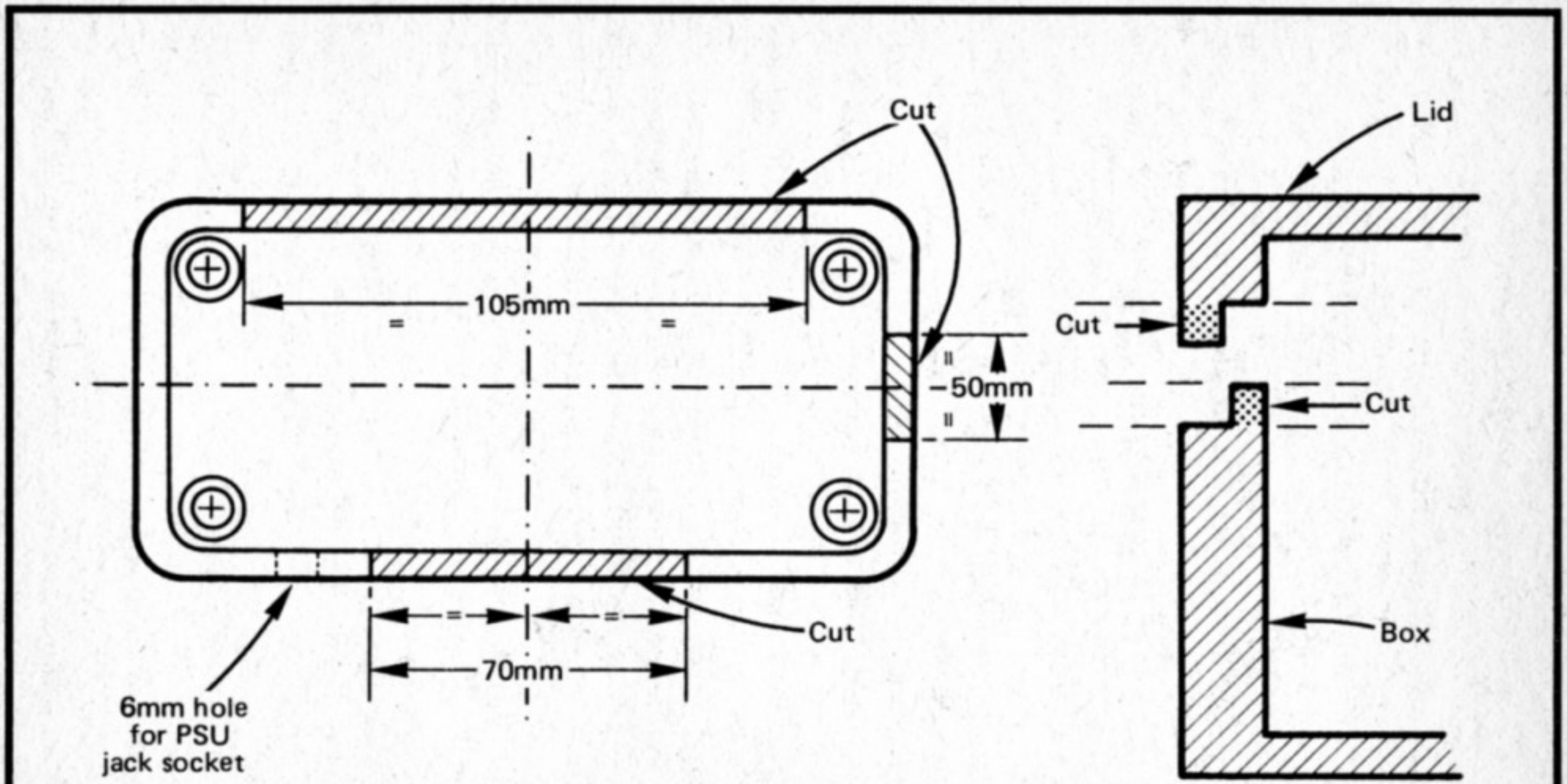


Fig. D.1 How to modify box and lid to take PCB

abled together. The "strobe" inputs 1G and 2G enable groups 1 and 2 respectively. These inputs are each controlled by an addressed output of IC4. This arrangement ensures that only one group of add-ons can send data to the micro at any one time. Note that it is possible to read the outputs from four add-ons simultaneously, if they are all plugged into the same group of one-bit data inputs. Normally we do not want to do this, so we read only the bit we are interested in by using the AND operator to mask out the other bits (p. 12).

Power Supplies

The VIC-20, Commodore 64 and Electron have limited capabilities for supplying power to external devices. Only about 100 mA is available and it is best to allocate this to powering devices such as the Light Pen and Games Controller used on the games port. An external power supply is required to power the Decoder and devices attached to it.

The BBC Microcomputer can supply 1.5 A at +5 V, which is enough for the Decoder and a reasonable range of attached devices. However, if you are using a disk device which takes its power from this source, it is preferable to power the Decoder and attached devices independently. A typical "complex" IC, such as IC4 and IC5 of the Decoder takes about 20–30 mA. Simpler ICs usually take a little less than 10 mA. If you propose to attach a large number of devices to the Decoder at the same time, it is safer to calculate their total power consumption, using the above figures as a guide. It is not likely that you will overload the supplies but, if in doubt, it is a simple matter to put a milliammeter in series with each device and measure how much current it takes.

The Decoder board (Figs. D.4 and D.5) provides for a +5 V regulator IC and its associated components (Fig. D.2). It requires a smoothed *direct current* (DC) supply of (preferably) at least 1 A and not more than 25 V. A lower supply voltage (say, 9 V), is recommended; if you use a higher voltage, be sure to put a heat sink on the regulator. It has been found that the 9 V power-supply units such as are supplied with the Sinclair Spectrum and ZX81, or the Jupiter Ace are

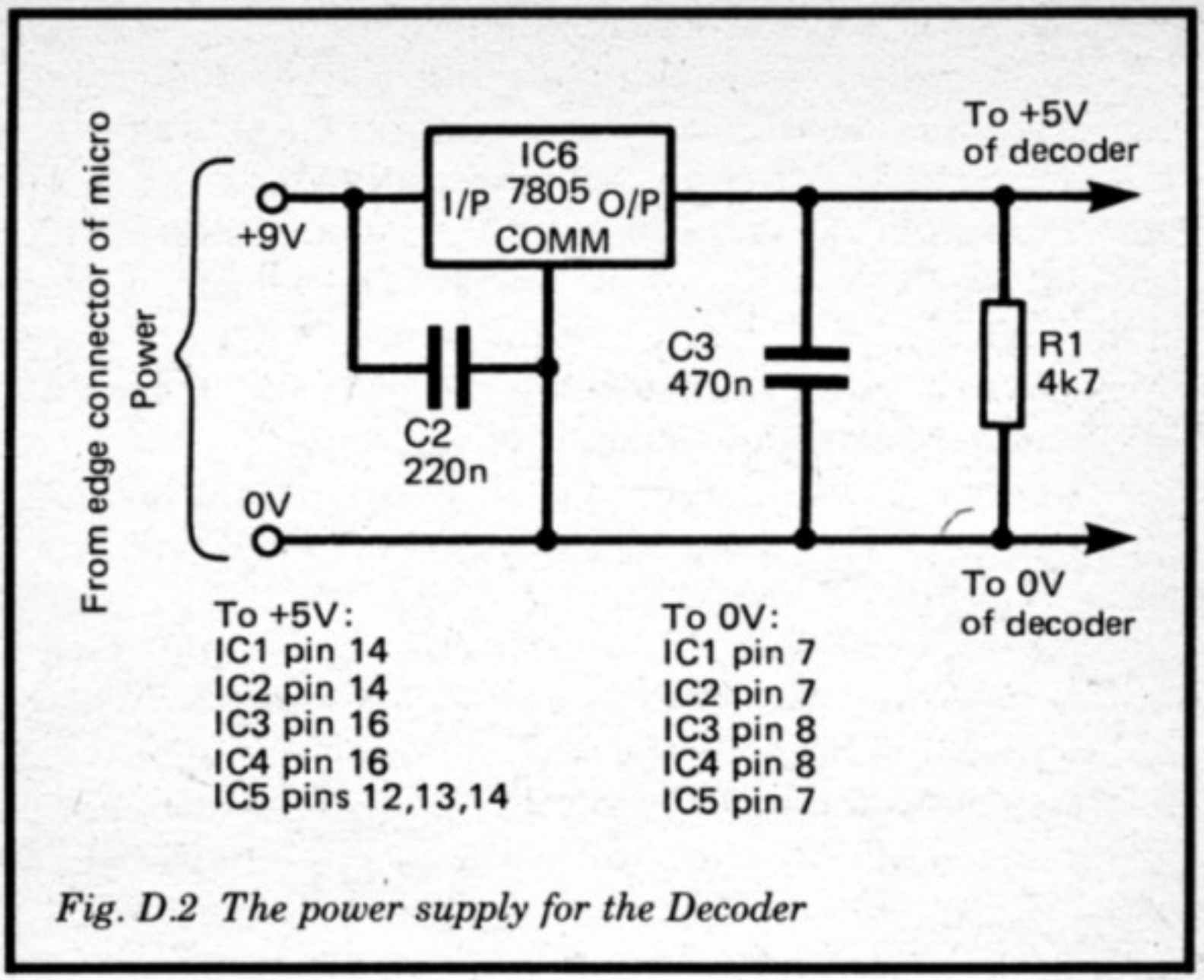


Fig. D.2 The power supply for the Decoder

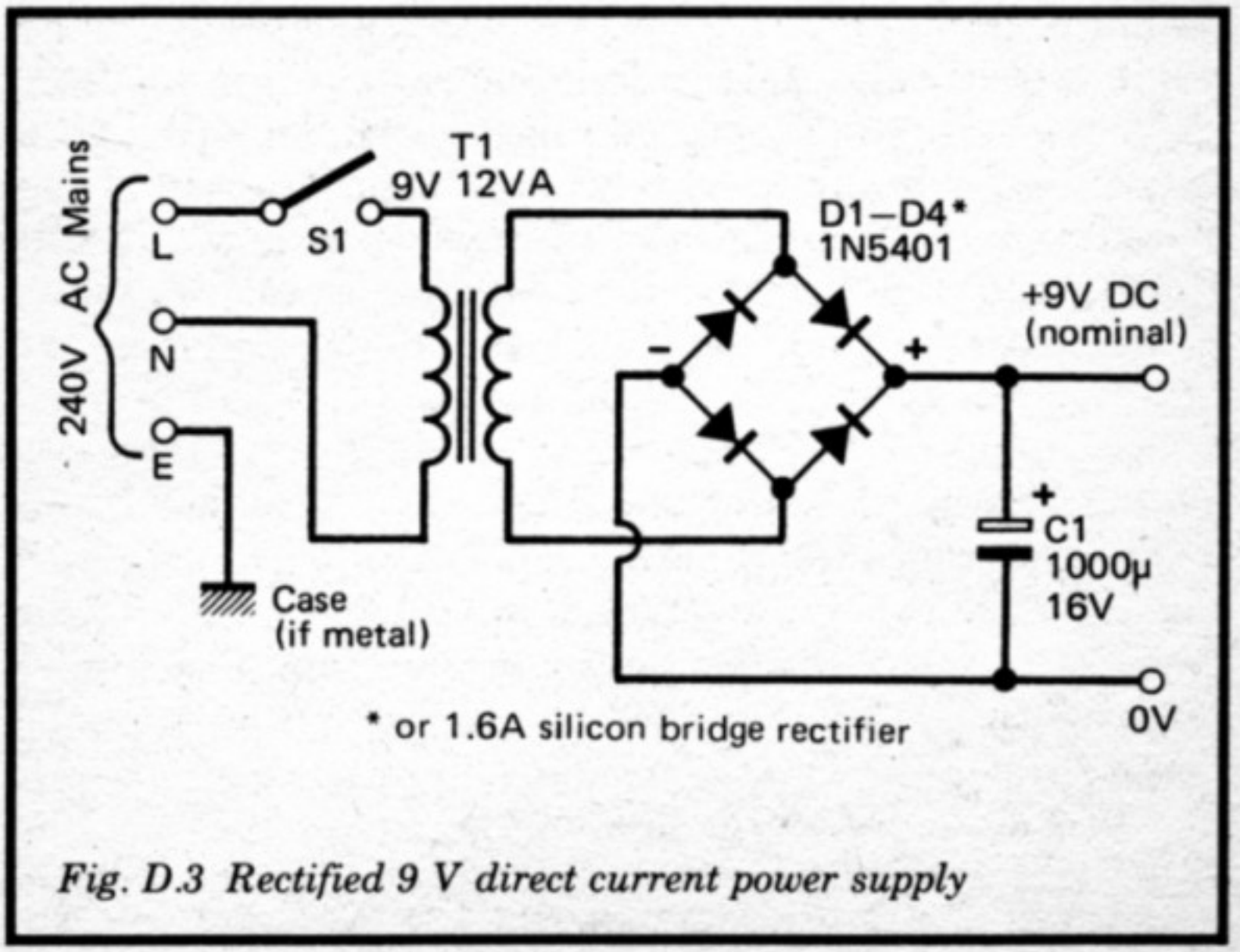


Fig. D.3 Rectified 9 V direct current power supply

suitable for connection to the regulator circuit of the Decoder. Failing this, or something similar, it is not difficult to construct your own supply unit. A suitable circuit is given in Fig. D.3.

Building the Decoder

The circuit can be assembled on ordinary stripboard, but there are so many connections to make that a p.c.b. is much to be preferred. The p.c.b. design (Figs. D.4 and D.5) shown is a universal layout which will suit all four computers. It can be fully assembled and used on all four, or assembled just for one particular model. Its complete circuit is shown in Fig. D.6. Description of the action of ICs 1, 2 and 3 is given in Appendix B.

For all models except the Electron, only ICs 4, 5 and 6 are required; so you may prefer to design your board without the Electron edge connector and without ICs 1, 2 and 3.

The universal p.c.b. is available, together with all associated components, from the supplier listed in Appendix C and, if this is used, the Decoder can be updated at any later stage to suit any of the four machines. The p.c.b. offered is double-sided, and the kits contain all the necessary components.

It is not recommended that beginners or those inexperienced at making their own p.c.b.s should attempt to construct their own double-sided board for a design as complex as this.

Assembling the P.C.B.

1. Side B of the p.c.b. has pads for (a) IDC connector, (b) add-on plugs, (c) through-linking pins (largest pads, 78 off *total*). Fit all through-linking pins and solder both sides. Do *not* fit IDC or plugs at this stage.

2. Turn the p.c.b. over and fit all IC sockets, capacitors and resistors, and carefully solder using a fine iron.

3. Fit IC6 and PSU terminals. Solder carefully. (This step is not required on BBC models.)

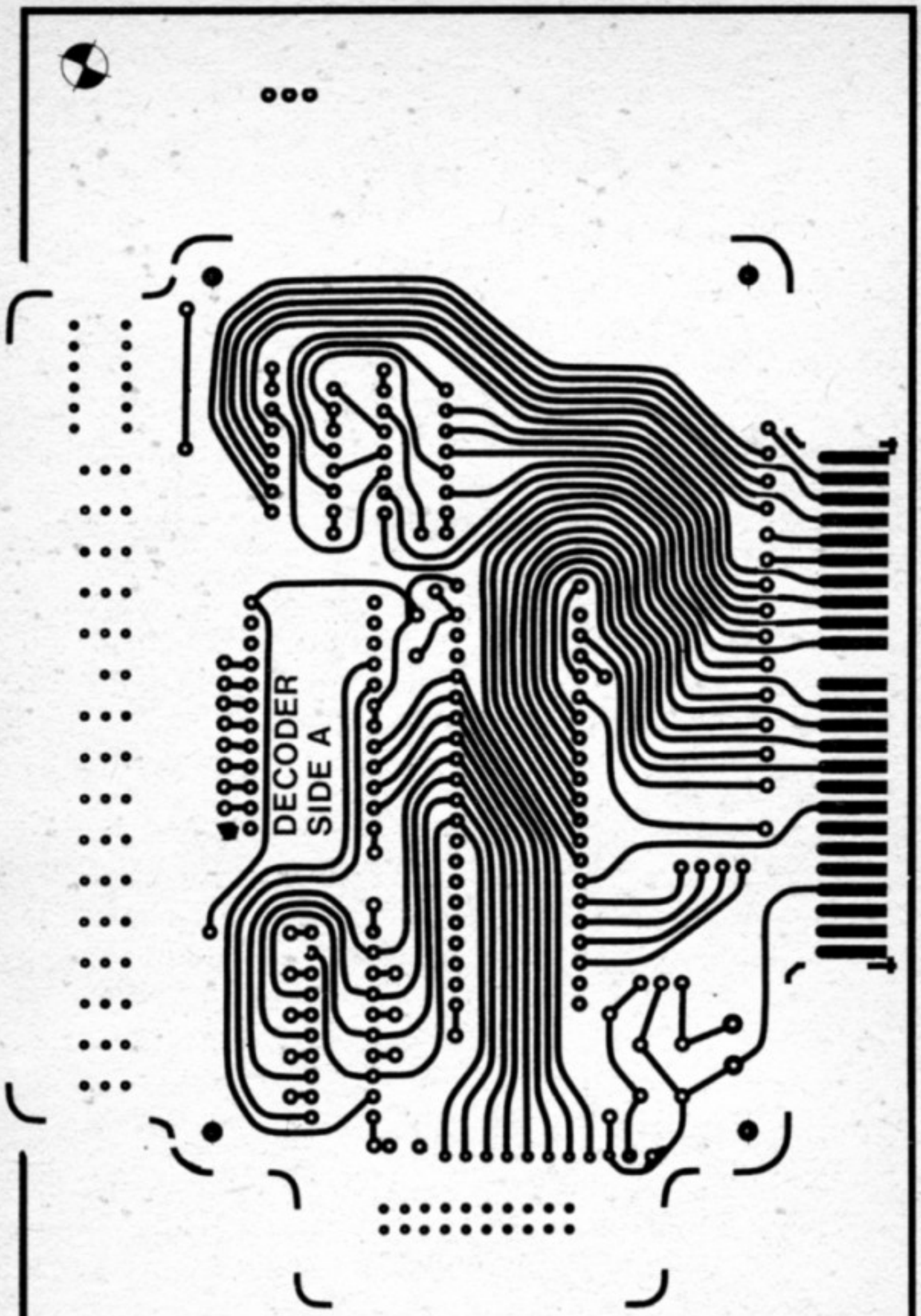


Fig. D.4 Universal Decoder p.c.b. (Side A)

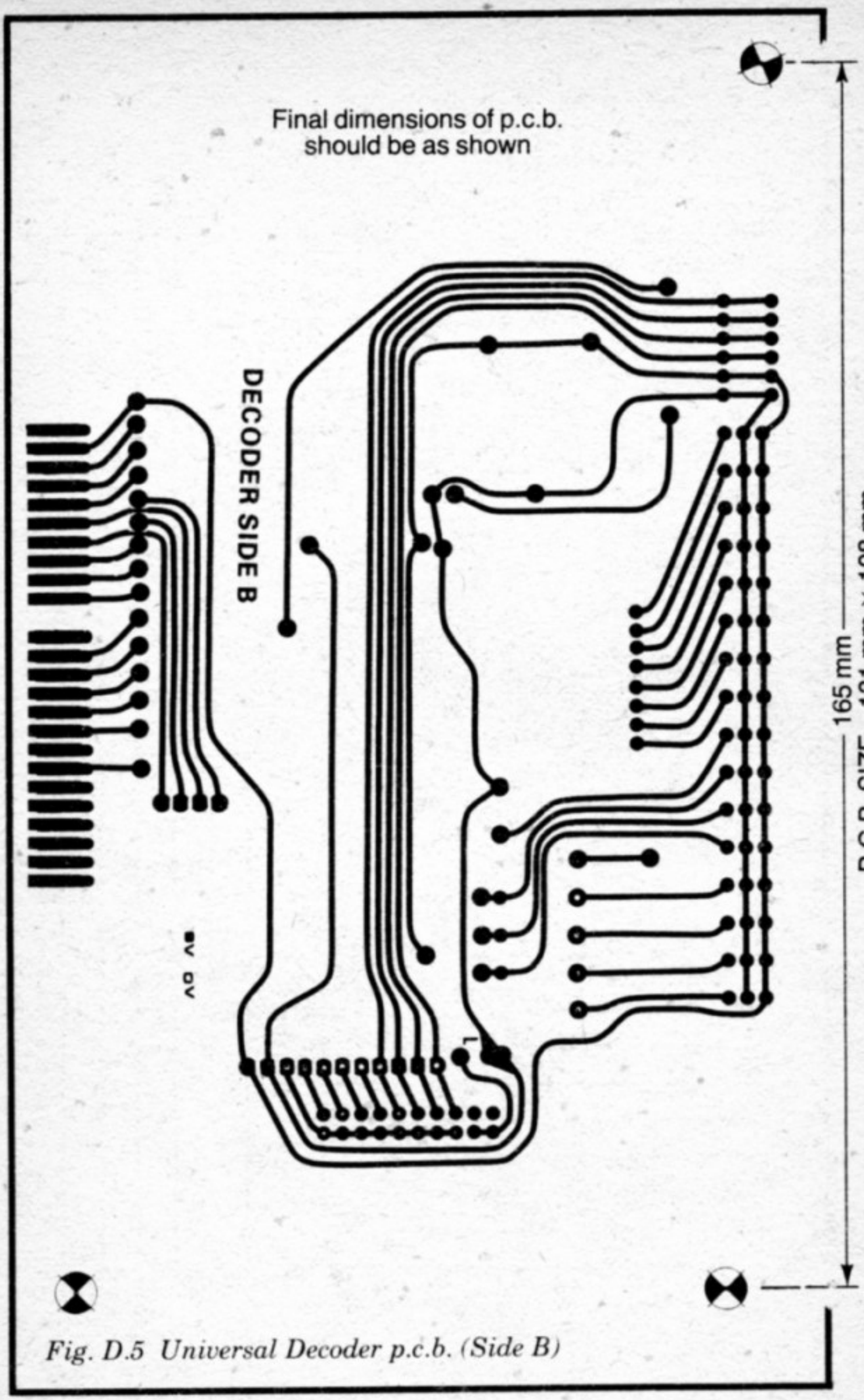


Fig. D.5 Universal Decoder p.c.b. (Side B)

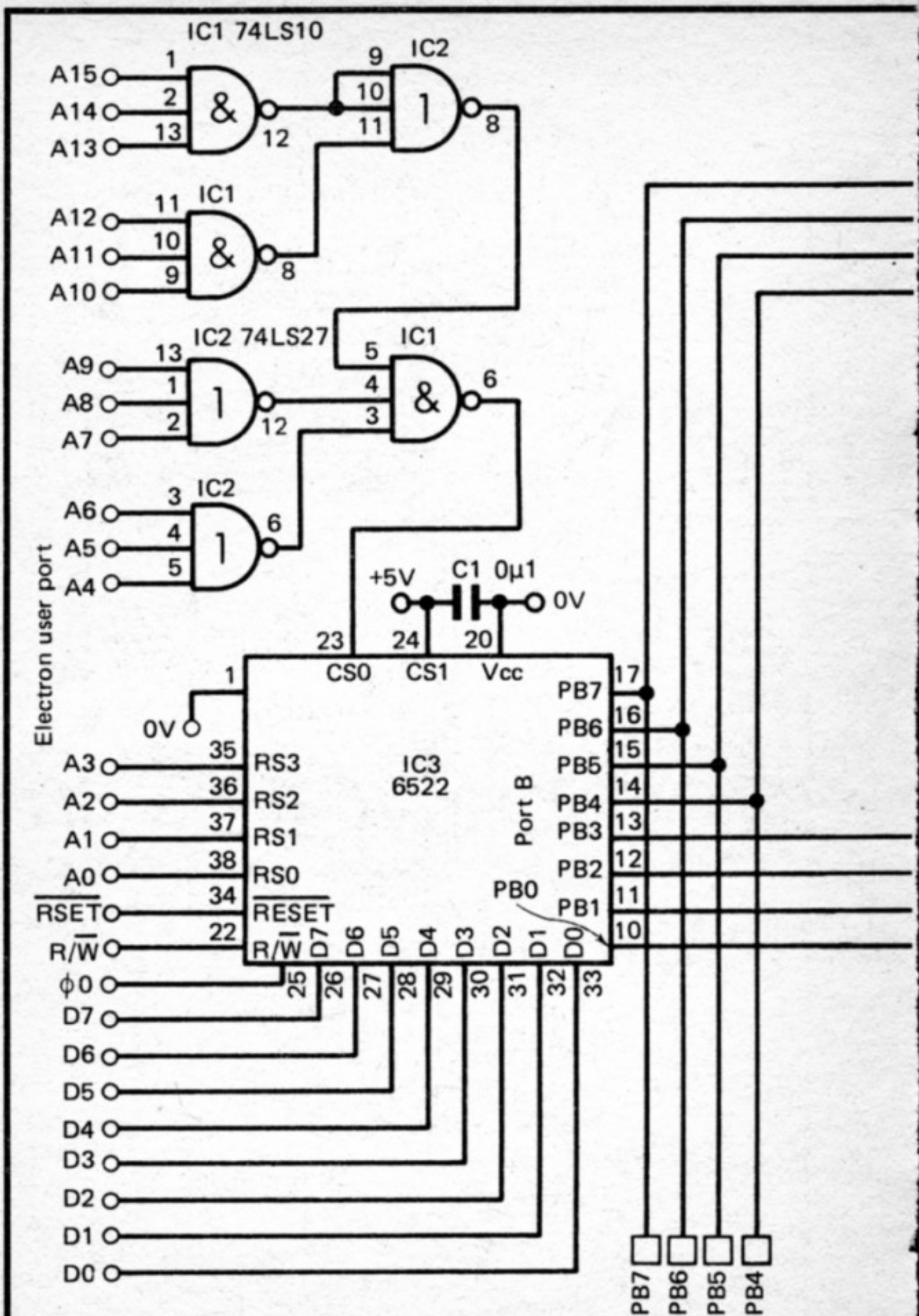
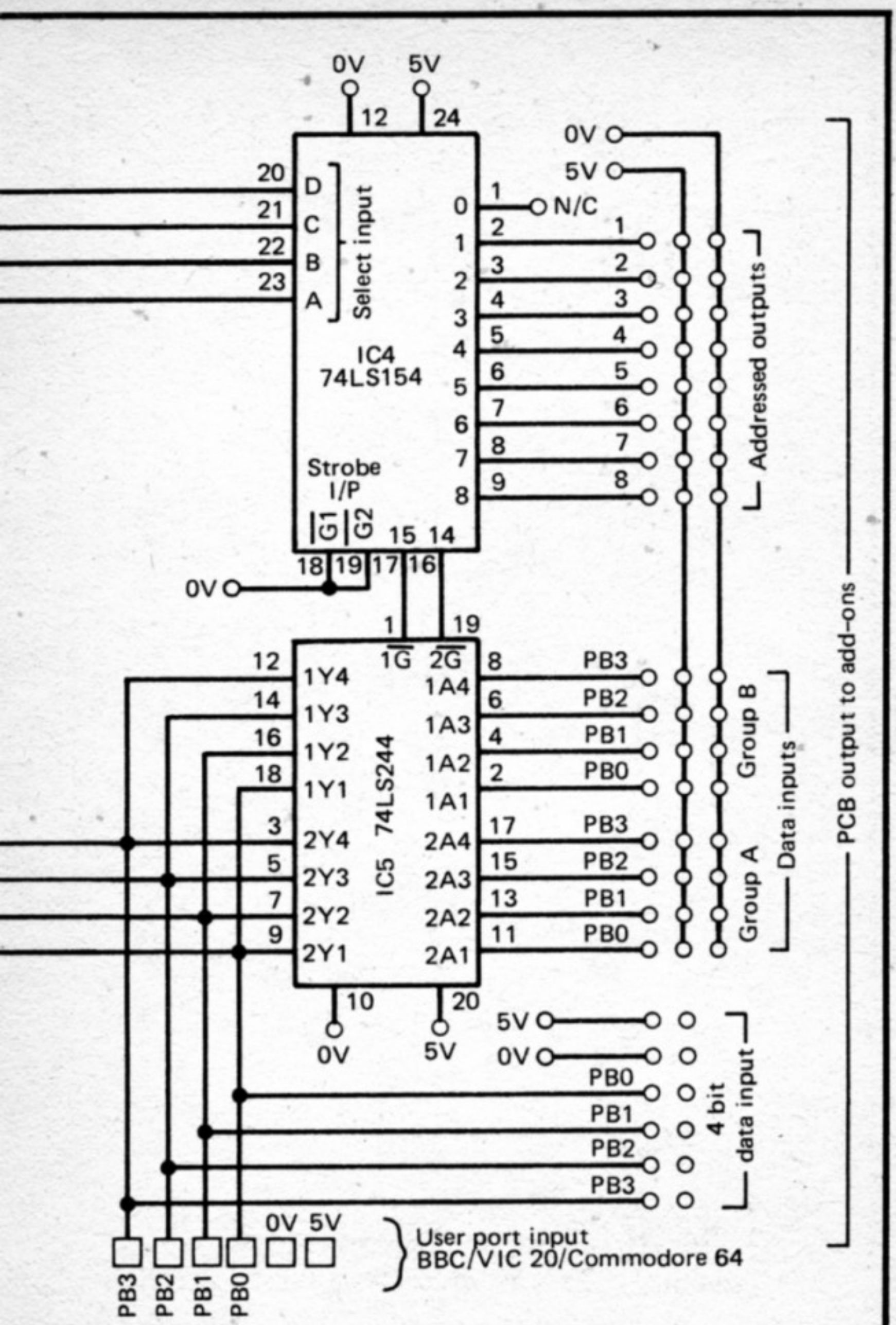


Fig. D.6 Circuit diagram of the Universal Decoder board


0V
5V
12
24
20
21
22
23
D
C
B
A
Select input
IC4
74LS154
Strobe
I/P
G1
G2
15
14
18
19
17
16
0V
0
1
2
3
4
5
6
7
8
9
N/C
Addressed outputs
1
2
3
4
5
6
7
8
1
19
1G
2G
IC5 74LS244
1Y4
1Y3
1Y2
1Y1
2Y4
2Y3
2Y2
2Y1
1A4
1A3
1A2
1A1
2A4
2A3
2A2
2A1
12
14
16
18
3
5
7
9
8
6
4
2
17
15
13
11
PB3
PB2
PB1
PB0
Group B
Group A
Data inputs
PCB output to add-ons
10
20
0V
5V
5V
0V
PB0
PB1
PB2
PB3
4 bit
data input
0V 5V
PB3
PB2
PB1
PB0
User port input
BBC/VIC 20/Commodore 64

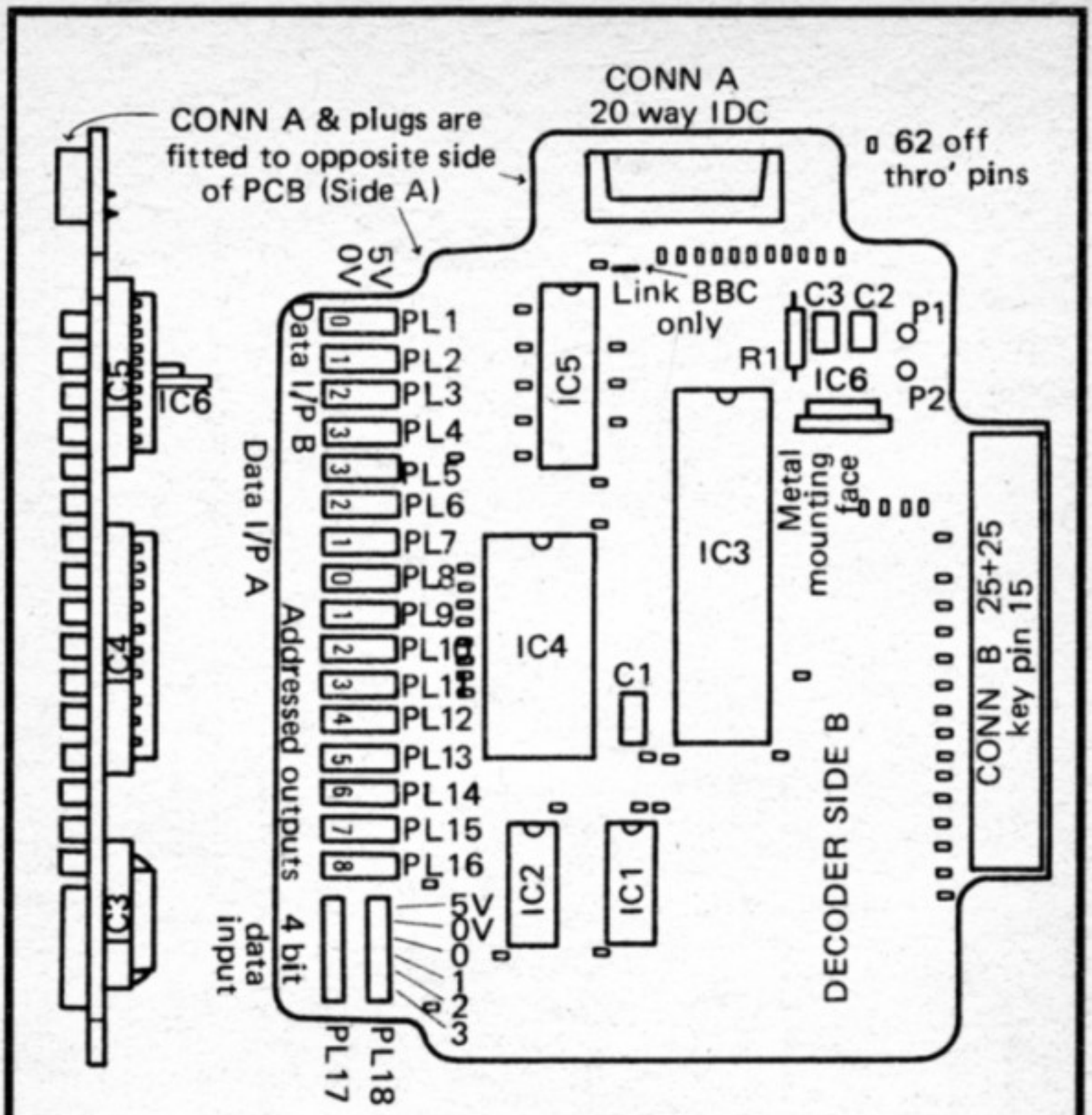


Fig. D.7 Component layout of the Universal Decoder board

4. Turn p.c.b. back over and fit IDC; solder. Then fit all add-on plugs and solder.

5. Fit ICs into sockets, taking care not to bend any legs.

Notes on specific machines

BBC: Fit IC4, IC5 and 5 V link only. Fit IDC connector.

VIC-20 and Commodore 64: Fit IC4 and IC5. Fit IC6 with associated components (C2, C3, R1, P1, P2). Fit IDC connector.

Electron: Fit all components except IDC connector. (This can be fitted if you wish to have a BBC-type user port as an additional bonus, though some additional wiring to the port will be required.)

IC Sockets

It is possible that the ICs could work loose from the sockets, as they will normally be upside down when in use. If it is likely that the box will be moved or knocked about a great deal, it may be preferable to solder the ICs directly into the p.c.b. This should be done with great care, bearing in mind that it is difficult to desolder to replace damaged or faulty chips.

Fitting the Case

The p.c.b. layout shown is designed to fit into a VERO GP case No. 202-21390, which is supplied in all the kits (*see* Appendix C). Very little work is needed to provide a very robust interface to professional standards.

Remove the lid from the box and cut away the flanges, as shown in Fig. D.1, from both box and lid. This is easily done with a sharp scalpel or Stanley knife.

The universal p.c.b. will now clip onto the box lid using the four fixing posts to locate it firmly. Ensure that when fitted into the lid the ICs are facing away from the lid (i.e., into the box).

Fit the PSU jack socket, and solder two wires from the jack to the 0 V and 9 V terminal pins.

Test the p.c.b. prior to fitting (*see* p.183).

Fit the lid to the box; fit the screws after checking correct alignment.

A ZX PSU will provide adequate power for all add-on projects, but Fig. D.3 shows a separate circuit for those who may wish to build their own. (The BBC does not require this, as sufficient power is available from the machine.)

Decoder/Computer Interconnectors

The supplier listed in Appendix C can supply all interconnectors (Figs. D.9 and D.10), ready made-up to suit all machines:

BBC: 2 × 20-way IDC connectors on ½ metre ribbon cable.

Vic-20 and Commodore 64: 2 × 20-way IDC, as BBC but with 20-way adaptor to fit into user port socket. (IDC header/p.c.b./12-way edge connector.)

Electron: 2 × 25-way 0.1 pitch edge connectors keyed in position 15 and fitted with ½ metre ribbon cable.

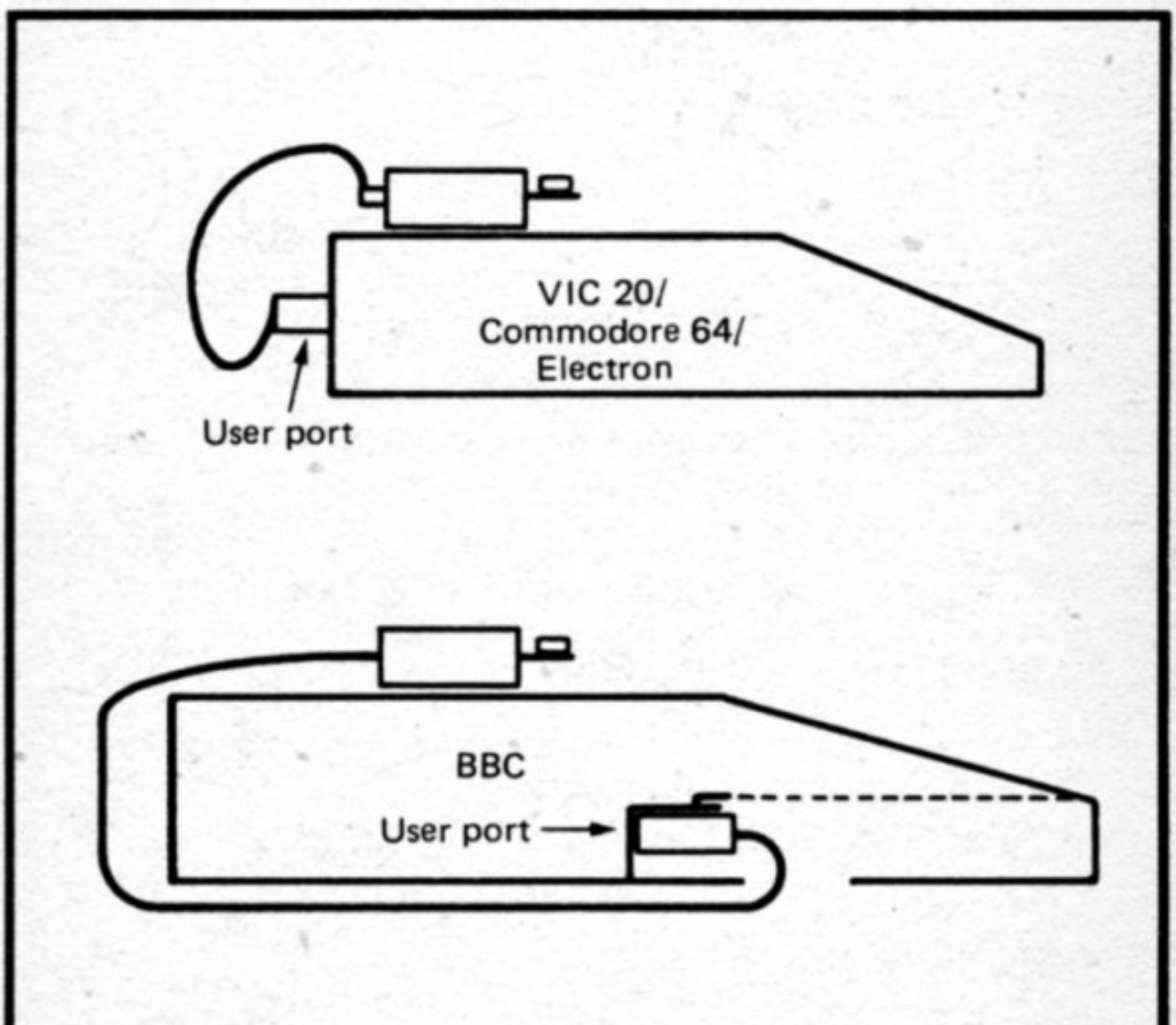
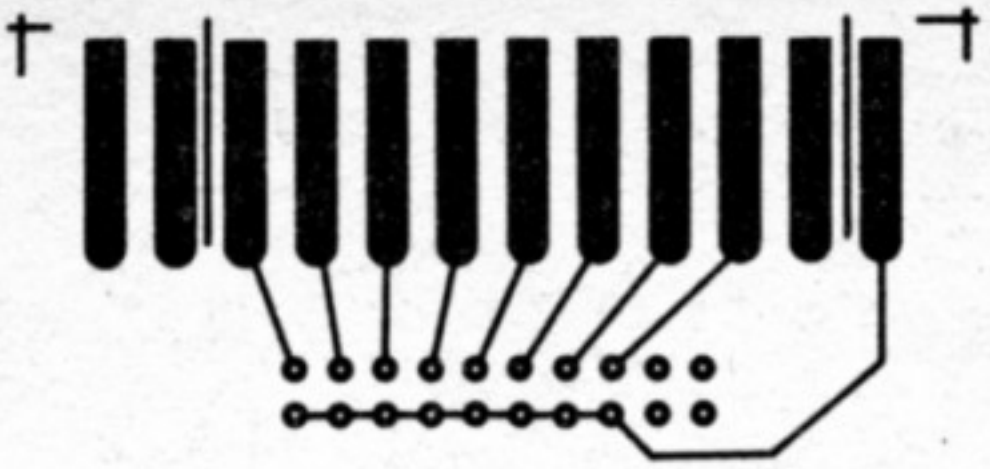


Fig. D.8 Connectors



COMMODORE USER EXTENDER

NB: Distance between centre to centre of each finger should be 0.156".

Fig. D.9a Converter board for use with VIC-20 and Commodore 64

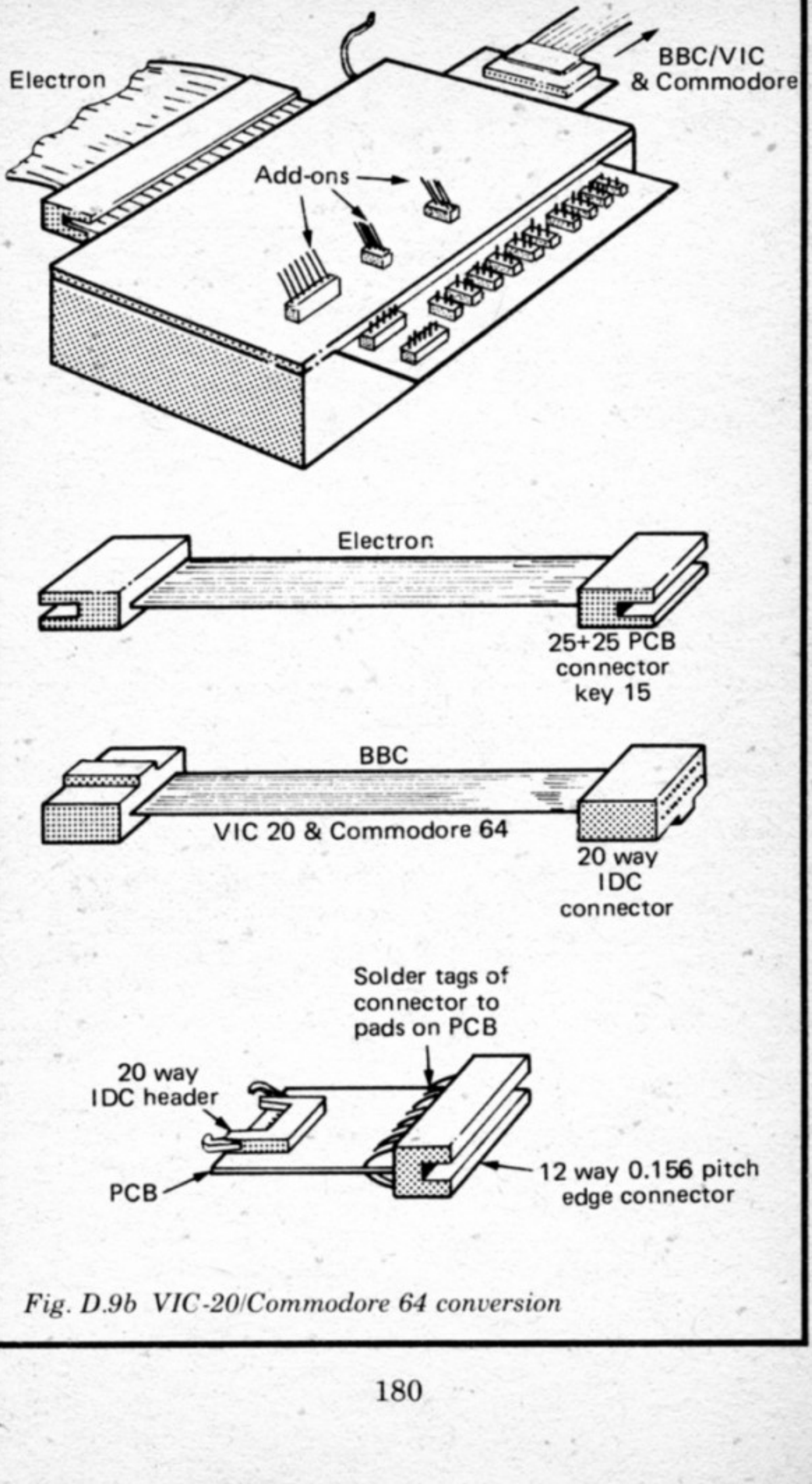


Fig. D.9b VIC-20/Commodore 64 conversion

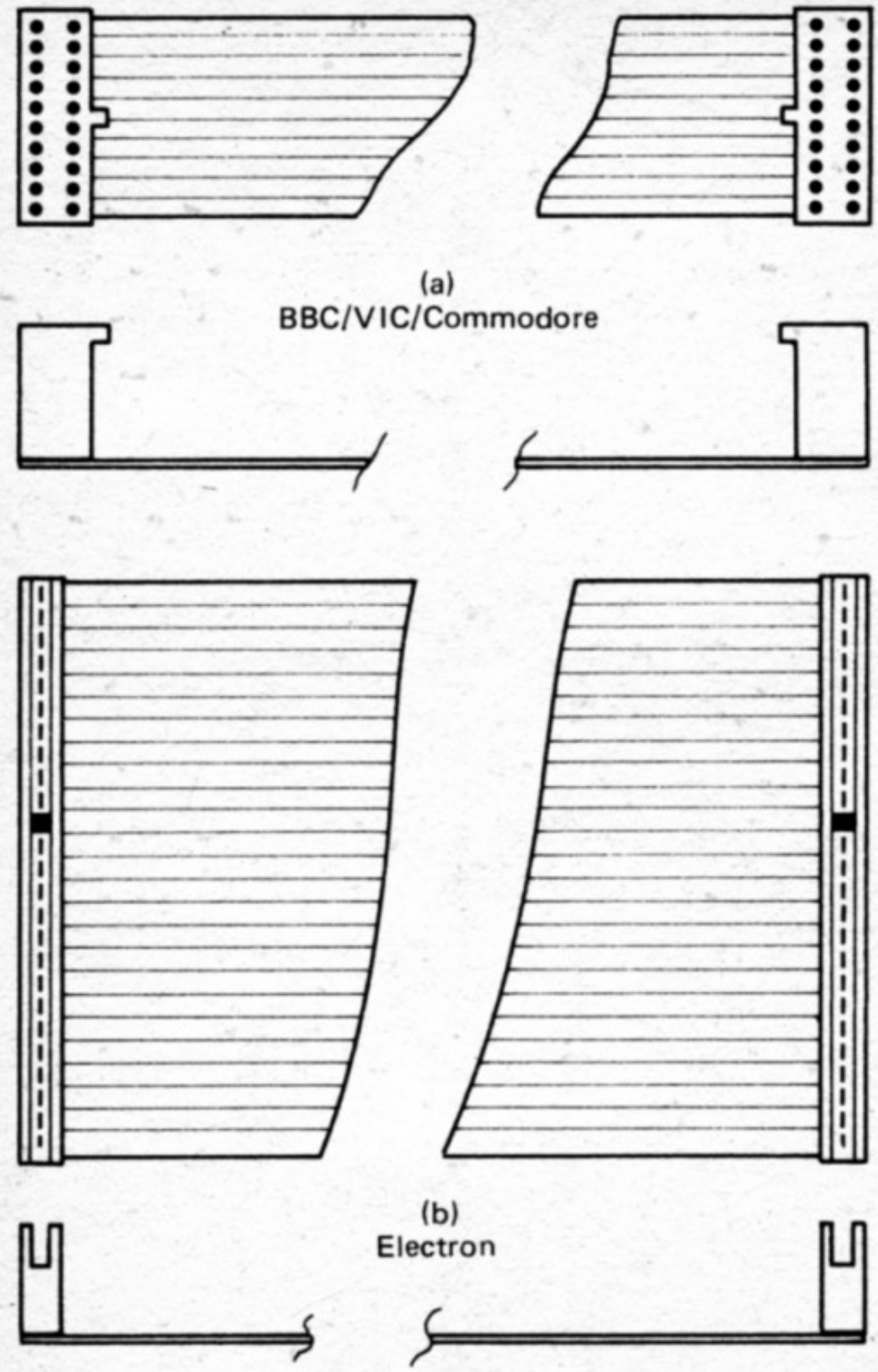


Fig. D.10 IDC ribbon assembly

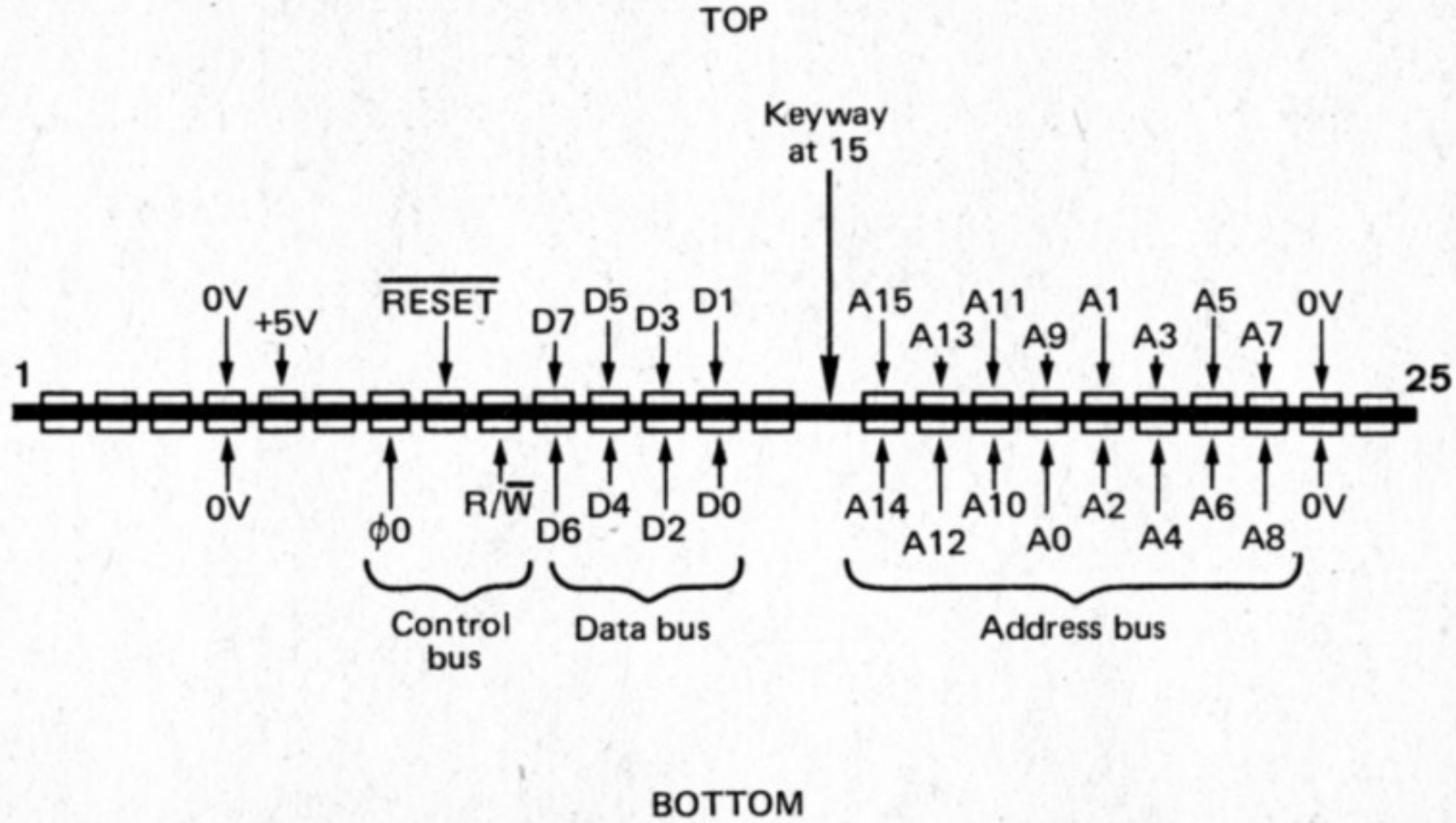


Fig. D.11 The edge connector of the Acorn Electron, as seen from the rear of the machine. Only those connections used for the port B interface are marked

Testing

Before plugging the power supply into the Decoder and before plugging the Decoder into the micro, use a test meter switched to its resistance range to check that there are no short circuits between adjacent cable lines and between these lines and the 0 V and +5 V lines. Skip the next 3 paragraphs if you are using the Decoder with the BBC Microcomputer.

If you are testing the Electron version, remove IC3 from its socket. Next plug in the power supply and switch it on. Use a voltmeter to test the outputs of IC4. With all 4 select inputs unconnected (and therefore acting as if they were receiving a "high" input), only output 15 should be low. Try connecting one or more of the "select" inputs to the 0 V line, and check that the correct output goes low. For example, if you are ground PB5, the equivalent binary number is "1101", or "13" in decimal. Only output 13 should be "low". If all outputs stay "high", check that the strobe inputs are properly grounded. Note that when we refer to "high" outputs this does not necessarily mean +5 V. A low-power TTL output is considered to be "high" if it is greater than 2.4 V.

With all "select" inputs unconnected, making output 15 go "low", group 2 of the one-bit data inputs is enabled. Since nothing is connected to the inputs of the buffers, their outputs (lines PB0 to PB3) should all be "high". If you connect any one or more buffer inputs to the 0 V line, the corresponding output should go "low".

If, when testing, you find that a given line fails to respond correctly, either remaining "low" or "high" or at some indeterminate voltage (0.8 V to 2.4 V), examine the circuit carefully. Look for "dry joints" in which the solder has not flowed smoothly on to the metal pads of the board. Look also for "bridges" of solder between adjacent tracks. A hair-thin solder bridge, too fine to be seen easily with the unaided eye can nevertheless cause an effective short circuit. Use a powerful hand lens to aid your inspection.

When the Decoder has been checked thoroughly, switch off all power supplies, including that of the micro. Replace

IC3 if you are using the Electron version. Plug the board into the user port of the micro. Switch on the power of the micro. Then switch on the power supply to the board. (With the BBC Microcomputer it is already switched on.) If at any stage the micro fails to produce its customary display on the screen, switch off immediately and re-examine the Decoder (or any device attached to it) for faults.

You can test the Decoder's response to the micro as follows (testing the Electron version is described in Appendix B). Check the addressed outputs as follows, using the program lines given on pp. 15–18. First define port B with PB0 to PB3 as inputs and PB4 to PB7 as outputs. Then trigger an output address. Then test the corresponding terminal of IC4, using a voltmeter, to see if it has gone low. Failure to respond may be due to short circuits between data lines, faulty contacts in the plug at the user port, or incorrect cable connections.

Next use the commands for reading one-bit data inputs. Inputs which are unconnected read as "high". Their readings will be "1", "2", "4" or "8", depending on the line concerned. If you read all 4 lines, you will get "15". When you temporarily connect one or more buffer inputs to 0 V, you should find that the corresponding line reads as "0".

The Decoder is now ready to accept the add-ons.

APPENDIX B

Decoder with an Interface for the Acorn Electron

This circuit provides a port B input/output for the Acorn Electron computer so that the Decoder may be used with this machine. The interface is on the same board as the Decoder, allowing all add-ons to be used with the Electron.

How It Works

The circuit is based on the 6522 VIA, which is the I/O device used in the VIC-20 and the BBC Microcomputer. In order to operate it, there must be an address decoding circuit. The circuit shown in Fig. D.6 (IC1 and IC2), places the interface at address &FC00, or 64512 in decimal. This is the address of port B itself. The address of Data Direction Register B (DDRB) is &FC02, or 64514. Apart from this difference, programming devices attached to the interface is exactly the same as programming devices attached to the BBC Microcomputer. Follow the instructions as given in the summary (pp. 16–18) or under the various projects, but use the addresses given above.

The output of the address-decoding circuit (IC1, pin 6) goes low whenever the address lines carry an address in the range &FC00 to &FC0F. This output goes to the "CHIP SELECT 0" input of IC3, and enables the chip for action. The lower 4 lines of the address bus go directly to the "REGISTER SELECT" inputs of the VIA to decide which of its various registers (port B, DDRB, etc.) is being addressed. At the same time, there may be data on the data bus to be sent to the VIA.

This goes to one of its registers, and may be sent on to the add-on through the port B terminals. If data is being sent to the VIA, the READ/WRITE (R/W) line carries a low level. Alternatively, the micro may be reading data from registers

in the VIA, in which event the R/W line carries a high level.

The RESET input of the VIA is connected to the RESET line of the micro. Whenever the system is reset (for example, when you switch on, or when you press the "BREAK" key) the registers in the VIA are all set to zero.

The VIA needs an input from the system clock (ø) so that its operation may be synchronised with that of the Electron's microprocessor.

Testing the Electron Version of the Decoder

First test the circuit as described in Appendix A. Then connect the device to the Electron, switch on and check the action of IC1, 2 and 3 as described below. Testing is done by typing statements at the keyboard and then pressing RETURN. This statement configures all lines of port B as outputs:

```
DDRB = 64514 : ?DDRB = 255
```

Now send various values to port B, using a statement such as:

```
PORTB = 64512 : ?PORT B = 1
```

The example makes line PB0 go high (almost +5 V), and all other lines are low (0 V). Try several other values ranging between 0 and 255, and check that the corresponding output pins of IC3 go high.

If you do not get the expected result, switch off and remove the decoder from the Electron. Check for crossed lines and "dry" joints. If there was no response of any kind from the VIA, check the R/W and clock lines. It is also possible that the address-decoding circuits have crossed lines too. After checking lines, reconnect the board to the Electron and use Project 1 to test for a low pulse from pin 8 of IC2 whenever either of the statements above is used.

Now convert port B to inputs:

```
DDRB 64514 : ?DDRB = Ø
```

Read the port:

```
PORTB = 64512: X = ?PORTB :PRINT X
```

With nothing connected to the port, all lines act as high inputs, so the result is 255. Temporarily connect one or more of the port B lines (PB0 to PB7) to the 0 V rail. The value obtained should alter correspondingly. For example, if PB0 is connected to 0 V, the value is 254. If PB7 is connected to 0 V, the value is 127.

Programming

In general, programming is exactly as described for the BBC Microcomputer, except that the addresses of DDRB and port B are as given above.

There are two other differences:

Light Pen: The Electron has no light pen input, so the method of using this is as described for BBC Microcomputer Model A.

Games Control: There is no analogue input on the Electron. Build the controller circuit as described for the BBC machine (Fig. 11.1c), but feed its input to a 507C, i.e., as in Fig. 15.1. The result is read as described for Project 15.

APPENDIX C

Suppliers of P.C.B.s and Kits

All parts are available in full kits (except for PSU) from the address below. Please write for details specifying which computer your kit is required for. (PCB only also available.)

Kelan Engineering Ltd
Circuit Products and Components Division
Farnell Technology Park
Boroughbridge
N. Yorks.
YO5 9UY

PLEASE NOTE

The Publishers of this book are in no way responsible for the manufacture or supply of the aforementioned kits or PCBs and all enquiries must be sent directly to Kelan Engineering Ltd.

Parts Required for the Decoder

	Component Type/Description	*BBC*	*Elec-tron*	*VIC-20*	*Com 64*
IC1	74LS10	—	1	—	—
IC2	74LS27	—	1	—	—
IC3	6522(VIA)	—	1	—	—
IC4	74LS154	1	1	1	1
IC5	74LS244	1	1	1	1
IC6	7805 regulator	—	1	1	1
C1	0μ1 capacitor – ceramic	—	1	—	—
C2	220 n capacitor	—	1	1	1
C3	470 n capacitor	—	1	1	1
R1	4k7, ¼ W, 5% resistor	—	1	1	1
CONN A	20-way IDC (2 off) conn. with 9 m ribbon cable	1	—	1	1
CONN B	25 + 25-way (2 off) edge conn. with 1 m ribbon cable	—	1	—	—
PL1–16	3-way p.c.b. plugs	16	16	16	16
PL17–18	6-way p.c.b. plugs	2	2	2	2
SK1–16	3-way sockets	16	16	16	16
SK17–18	6-way sockets	2	2	2	2
SK19	PSU jack (to suit PSU)	—	1	1	1
P1, P2	P.c.b. pins/stakes	—	1	1	1
Case	G.P. plastic (120 mm × 80 mm)	1	1	1	1
Extender	(for VIC-20 & Com. 64) 20-way IDC conn. & p.c.b. & 12-way 0.156 pitch edge conn.	—	—	1	1
SK20	IC socket – 14-pin	—	2	—	—
SK21	IC socket – 40-pin	—	1	—	—
SK22	IC socket – 20-pin	1	1	1	1
SK23	IC socket – 24-pin	1	1	1	1
PSU1	External power supply, 9 V, 1 amp (mW)	—	1	1	1
Pins	Track pins	62	62	62	62

APPENDIX D

Pin Lead-out Details

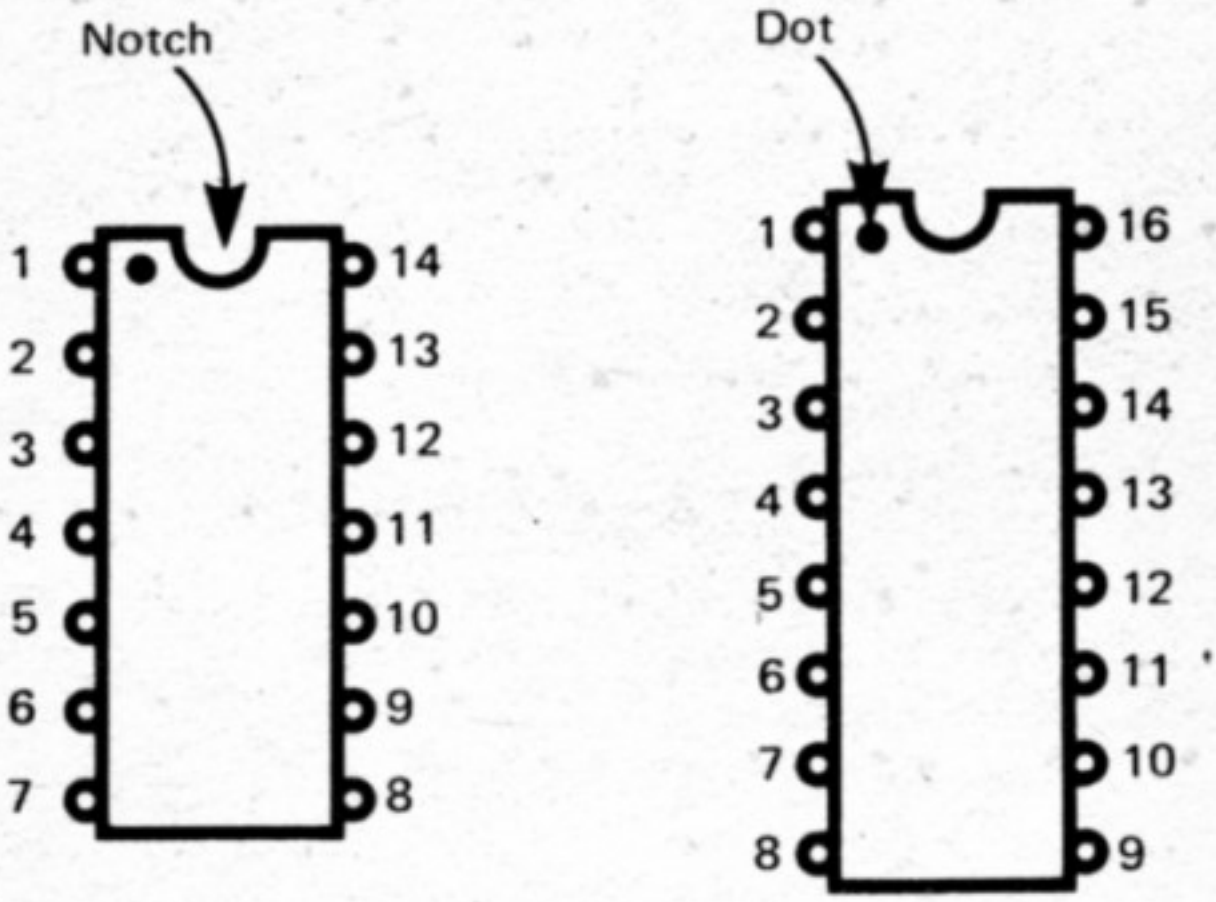


Fig. A.1 Pin numbering of integrated circuits, as seen from above

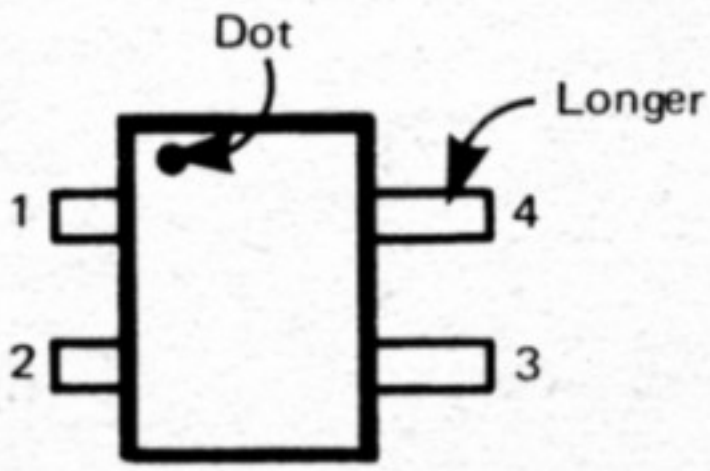


Fig. A.2 Pin numbering of the 634SS2 Hall Effect IC, as seen from above

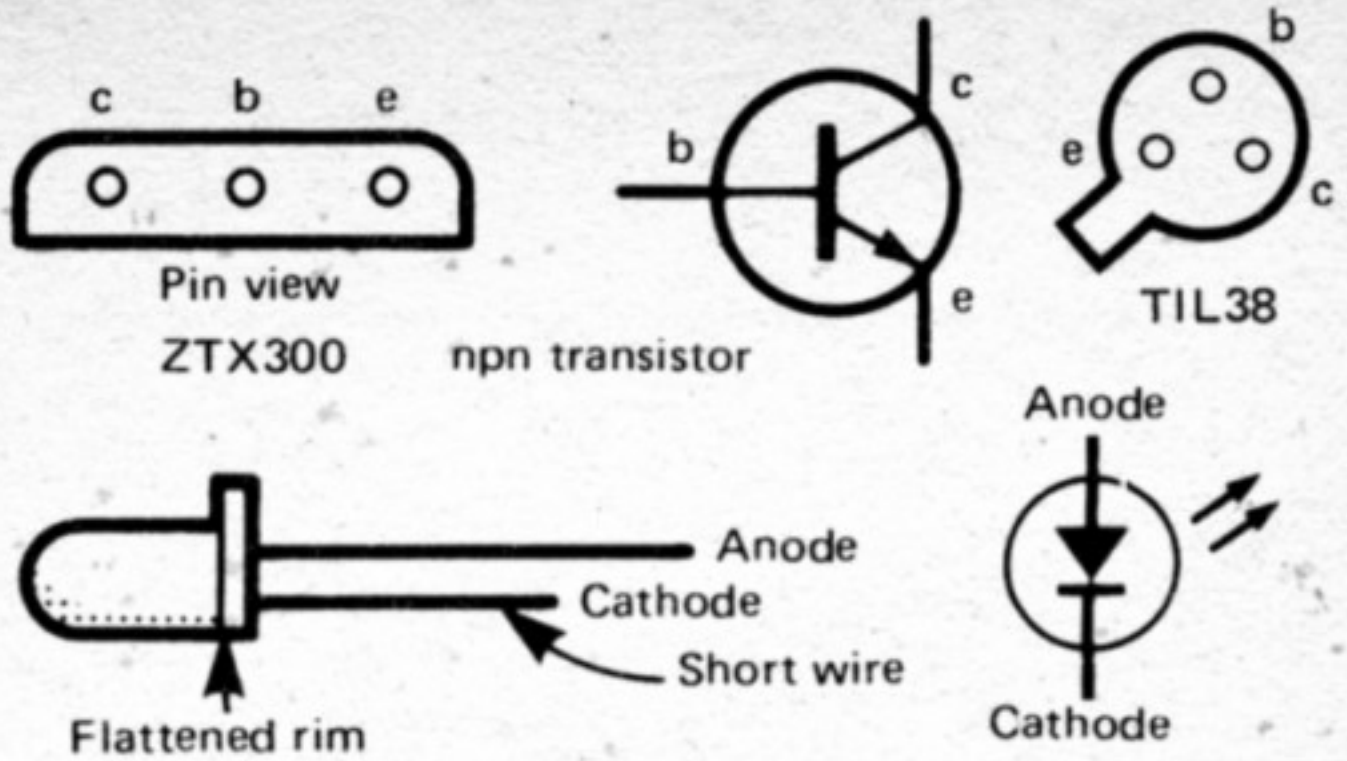


Fig. A.3 Pin connections and symbols of semiconductor devices used in this book

Notes

Notes

Notes

BERNARD BABANI BP134

Easy Add-on Projects for Commodore 64, VIC-20, BBC Micro & Acorn Electron

■ This book describes how to build a number of electronic projects which you can use with your Commodore 64, VIC-20, BBC Micro or Acorn Electron.

■ The projects include a Pulse Detector, Picture Digitiser, Five-key Pad, Model Controller, Bleeper, Lamp Flasher, Light Pen, Magnetic Catch, Lap Sensor, Photo-flash, Games Control and six more projects that make up a Weather Station.

■ All the projects are fairly simple and inexpensive to construct. The most complicated part, the Address Decoder, is constructed as a separate item that can then be used with any of the projects.

■ Once built, the projects are easy to operate and a simple program or two is included to get you started. Of course, those readers who are more experienced at programming can have a lot of fun in writing elaborate programs for these projects, but the beginner can start with a short program and perhaps add extra features later.

GB £ NET +002.95

ISBN 0-85934-109-7

00295

£2.95

9 780859 341097

Easy Add-on Projects for Commodore 64, VIC-20, BBC Micro & Acorn Electron

BP134 OWEN BISHOP