

# SPECTRUM



Melbourne  
House

# SPECTRUM *Hardware* MANUAL



NEW & REVISED  
FOR  
SPECTRUM  
E3

© 1983 Adrian Dickens

All rights reserved. This book is copyright and no part may be copied or stored by electromagnetic, electronic, photographic, mechanical or any other means whatsoever except as provided by national law. All enquiries should be addressed to the publishers:

IN THE UNITED KINGDOM —

Melbourne House (Publishers) Ltd  
Castle Yard House  
Castle Yard  
Richmond, TW10 6TF

IN AUSTRALIA —

Melbourne House (Australia) Pty Ltd  
1st Floor, 70 Park Street  
South Melbourne, Victoria 3205

ISBN 0 86161 115 2

dition 7 6 5 4 3 2  
Printing F E D C B A 9 8 7 6 5 4 3 2 1  
Year 90 89 88 87 86 85 84



## ACKNOWLEDGEMENTS

I would like to acknowledge the contributions of the many people who have provided valuable assistance and suggestions for improvements to the book. Alfred Milgrom of Melbourne House was very helpful from the start. My many friends at Cambridge, especially Mark Plumbley and Julian Bane were always prepared to discuss technical problems and provide suggestions for improvements. My brother Nigel and John Kimmitt at Churchill College proof read the book. The continuous support of both my parents was invaluable. Finally, I will be grateful to receive any further suggestions from Spectrum users for improvements to the book.

# CONTENTS

Chapter 1	Introduction .....	7
Chapter 2	General overview of the complete Spectrum .....	9
Chapter 3	The power supply — how the Spectrum supplies work — current limits — adding your own. ....	15
Chapter 4	The Z80A central processing unit — what it does — pin descriptions .....	23
Chapter 5	Video and program memory .....	27
Chapter 6	The BASIC read only memory .....	31
Chapter 7	The Keyboard .....	33
Chapter 8	The Uncommitted logic array — what it does — pin descriptions — avoiding video memory conflicts — interrupts — clocks — keyboard and cassette input — cassette, buzzer and border colour outputs — amplifying the buzzer .....	37
Chapter 9	The memory expansion sockets — upgrading to 48K .....	46
Chapter 10	The video circuit .....	55
Chapter 11	Tuning the Issue 2 circuit for better quality displays — changing the colour or grey scale .....	58
Chapter 12	The Edge connector — contact by contact description .....	62
Chapter 13	Fault diagnosis — no display? No colour? — random graphics? .....	65



Chapter 14	Experimenting with the edge connector — construction hints — using the accessible signals — RESET, INT, IORQGE, NMI, HALT 28 I/O ports .....	67
Chapter 15	Adding a Z80A PIO chip — a practical example ...	74
Chapter 16	Adding your own keyboard — a construction project .....	80
Chapter 17	Adding Sinclair compatible joysticks construction project — APOLLO game program .....	84
Chapter 18	Constructing an 8 channel analogue to digital converter — use as a joystick input drawing a program .....	90
Chapter 19	Adding a Centronics Parallel printer interface the hardware and software .....	96
Chapter 20	Connecting a Video Monitor to your spectrum .....	106

## APPENDICES

A	Glossary .....	109
B	References .....	111
C	Parts list cross reference .....	112
D	Component layout diagram .....	114
E	Complete circuit diagram .....	116
F	INDEX .....	121

## 1. INTRODUCTION

The ZX Spectrum as supplied includes an introductory booklet and a Basic programming manual. These books together contain virtually no detailed information on the hardware aspects of the computer. This book fills in that gap by providing an in depth look into how the Spectrum works. For the absolute beginner who knows nothing about computer hardware, this book provides a good introduction to what makes a computer tick. For the advanced electronics wizard who is desperate to use his Spectrum to perform all sorts of weird and wonderful tasks, this book also holds the key, by providing invaluable circuit diagrams and operational descriptions.

Starting off by explaining the fundamental principles behind the Spectrum's operation, the book progresses with full descriptions of how each part works and how all of the parts interrelate. After a complete explanation of all the edge connector signals, simple illustrative experiments making use of these signals are given. The remainder of the book consists of practical circuits which you can build. Constructional hints, drawings and photographs are included to help beginners with no previous experience.

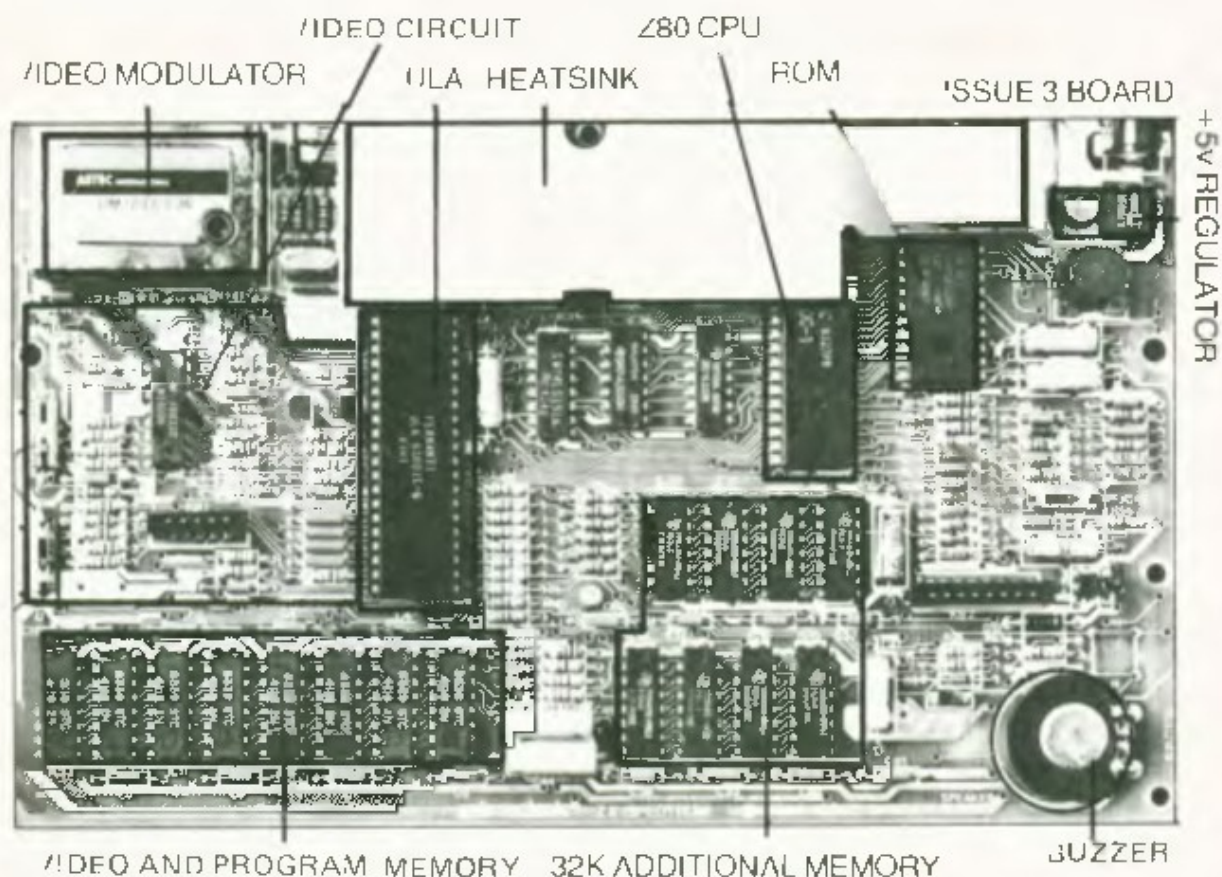
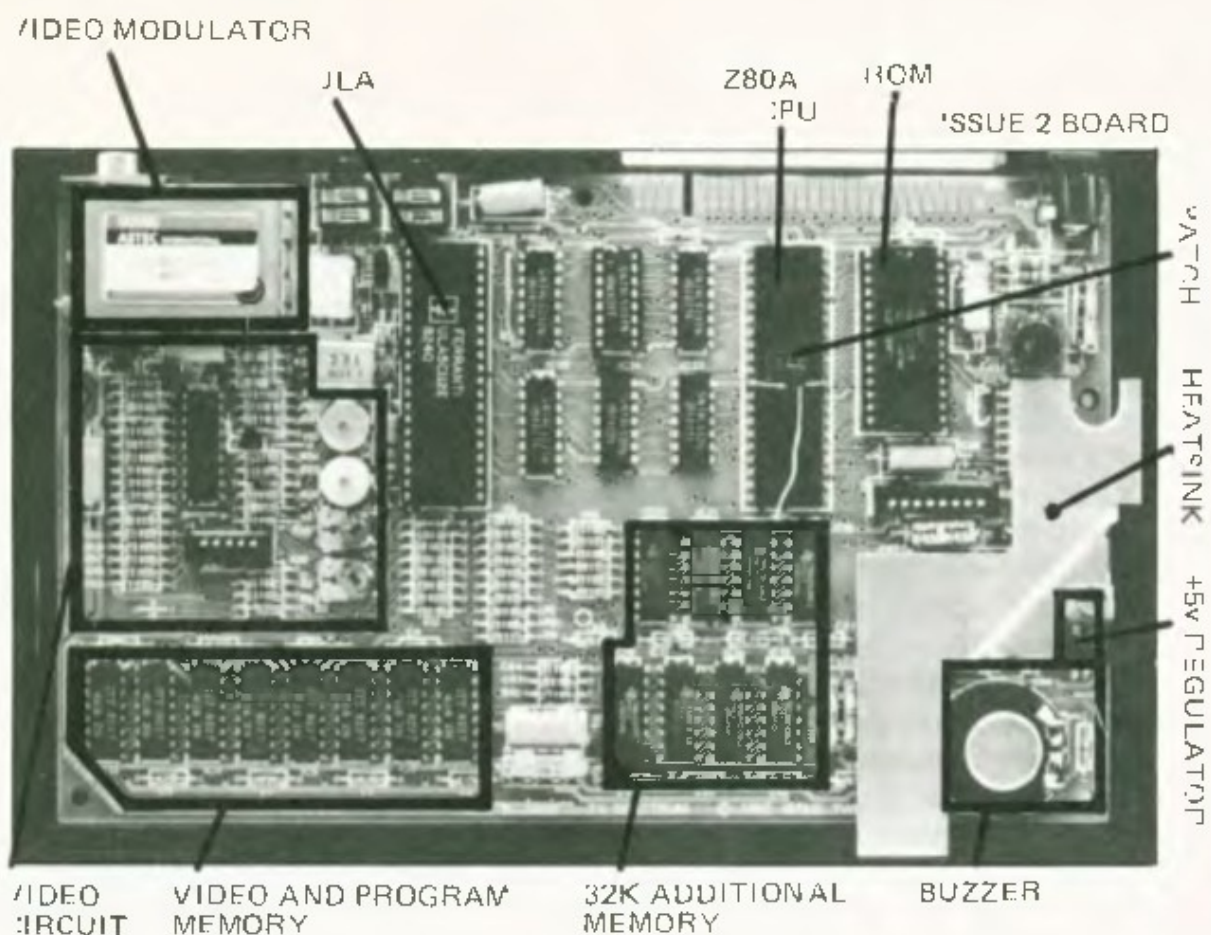
All of the Spectrum circuit diagrams contain component numbers such as TR4. These are generally the codes printed by the components on the Spectrum circuit board. There is a full layout diagram of the Spectrum board together with marked positions for each component in Appendix D. The complete circuit diagram can be found in Appendix E. You should refer to these appendices if you wish to find the position of any components. The component codes given for the additional hardware projects refer to the parts list for the project, not to the main Spectrum parts list.

There are currently three different versions of the Spectrum in circulation. The first 60,000 machines were 'Issue 1', the next half-a-million were 'Issue 2', and all new machines are 'Issue 3'. Earlier editions of the Hardware Manual dealt with the Issue 1 Spectrums. This edition only deals with Issue 2 and Issue 3 Spectrums. Whilst the differences between these two types of Spectrums are in most areas minor, certain features on the Issue 2 are no longer present on Issue 3's. Major differences are described in the relevant chapters as they occur. Some minor differences, such as a new capacitor or resistor are not included on all of the circuits throughout the book. However, the full circuit diagrams in Appendix E show all of these differences.

Before we start delving deeper into the intricacies of hardware, a few words of warning will be given. The Spectrum contains a lot of sensitive and expensive chips. These will not be damaged provided that you take adequate precautions. If you are adding any kind of interface to the Spectrum, always switch off all power supplies before making the connection to the Spectrum. Always switch on the Spectrum before or simultaneously with any external power supplies. Finally, if you are testing various voltages inside the Spectrum or on its edge connector, take care not to short any pins together by mistake. Check and double check everything you do!

**PLEASE NOTE** — whilst every effort has been made to ensure that all information given in this book is correct, no responsibility can be accepted for any errors which may have occurred. All Spectrum circuits are the copyright of Sinclair Research Limited. Other designs and programs are the copyright of the author.





'LATE 1 — PHOTO OF MAIN SPECTRUM BOARD & ANNOTATIONS



## 1. GENERAL OVERVIEW OF THE SPECTRUM

This chapter is essentially in two parts. The first part aims to introduce the basic concept of binary numbers. The second part explains each of the main sections of the Spectrum in a general way.

### SECTION A

Computers are essentially two state devices. They rely upon logic to operate. This logic can be in one of two states only. For convenience, these two states are usually represented by a 0 and a 1. At its simplest level, the computer manipulates 1's and 0's to produce the answers. Consider for example the simple operation of addition. This could be represented by a black box with two inputs, A, B and two outputs C, D. A and B could be added together to produce their sum represented by C and D. The addition would be carried out by several simple transistors inside the black box. The addition function would be defined in binary as follows:

A	+	B	=	D (sum)	& C(carry)
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

Note that  $1 + 1$  cannot equal 2 because in the binary number system, only 0 and 1 can be used. The carry bit here is similar to that in the decimal number system. If we added  $8 + 9$ , this would give 7 carry 1 in the decimal number system. The difference is that a carry in binary represents 2, whereas it represents 10 in decimal.

### SECTION B

In the following descriptions of the various sections within the Spectrum, you will find it helpful to refer to the overall block diagram in fig 1.

The central processing unit (CPU) is, as its name implies, central to the operation of the Spectrum. It is connected to other parts of the computer by data, control and address buses (more about these later). The CPU in the Spectrum is a Z80A and lives inside the large chip IC2 at the centre of your Spectrum board.

This processor is an 8 bit device which means that there are 8 separate connections in its data bus. The CPU can send information to other devices in the Spectrum along this data bus and the other devices can send data back to the CPU via the same bus. Because there are eight connections, each one of which can be either a logic 0 or a logic 1, any number between 0 (all 0's) and 255 (all 1's) can be sent via the data bus ( $255 = 2^8 - 1$ ).

You may wonder how the CPU can understand very large decimal numbers or words typed at the keyboard when it is using BASIC. After all, if you type HELLO at the Spectrum keyboard it might be difficult to see how this can be represented by a number between 0 and 255. In fact, the answer is not very difficult. The CPU only deals with a small part of the operation at a time. To understand HELLO, the CPU first deals with the H (seen by the CPU as 72 decimal), then it deals with E



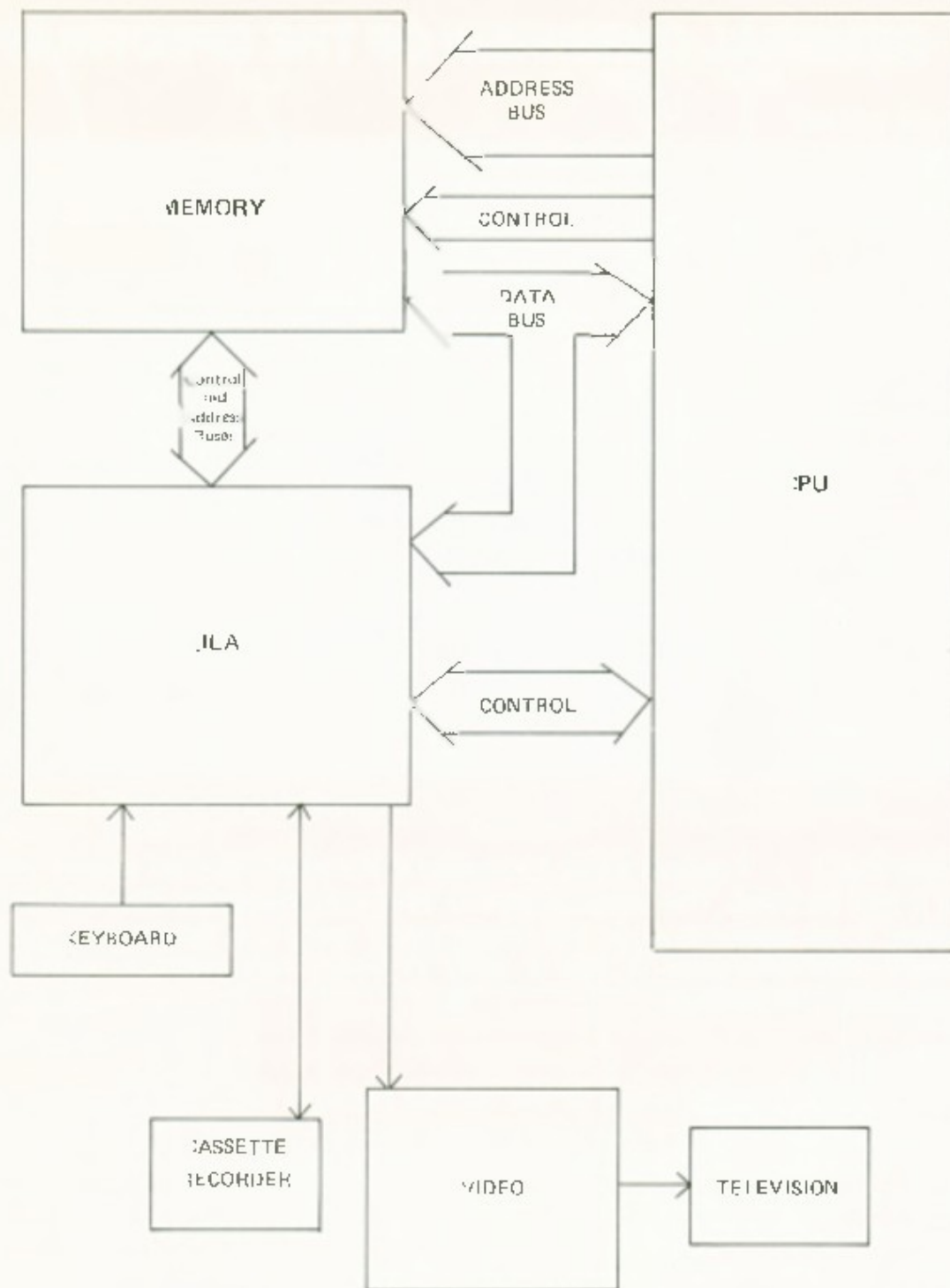


FIG 1 — SPECTRUM BLOCK DIAGRAM

seen as 69 decimal) and so on (see Appendix A of your Spectrum BASIC manual or a complete list of characters and their decimal equivalents). Large decimal numbers are dealt with in a similar way. Each decimal number is stored in the Spectrum in 5 bytes of memory. Chapter 24 of the BASIC manual explains how these bytes are used.

Before the CPU can actually start doing anything, it must be instructed what to do. The instructions for running BASIC are held in memory. The BASIC operating system program contains all of the information required by the CPU to understand BASIC. This operating program is written in machine code and starts to run when you switch the Spectrum on. The actual program is stored in the read only memory (ROM) chip IC5. Read only memory cannot be modified by the CPU and the program remains fixed in the memory chip even when the power is switched off.

The BASIC programs which you enter into the Spectrum go into random access memory (RAM). Unlike the ROM, this type of memory can be changed by the CPU. When the power is switched off, RAM forgets everything that was stored inside it. You therefore have to save your programs on a cassette before switching off. If you do not, the program will be lost forever.

Having found what to do from the program in ROM, the CPU must get inputs from the keyboard or cassette and send outputs to the video display or cassette.

The uncommitted logic array (ULA) helps the CPU to interface with the outside world. The ULA gets information directly from the keyboard and cassette inputs.

This information is then sent to the CPU. When the CPU wants to record a program on cassette or buzz the buzzer, it tells the ULA to do it. Output to the television is rather more complex. The ULA copies the screen output from the video memory to the video output circuit 50 times every second. This creates the illusion of a continuous display. All that the CPU has to do when it wants to output to the TV display is to put the video information into the video memory. The ULA then does the rest.

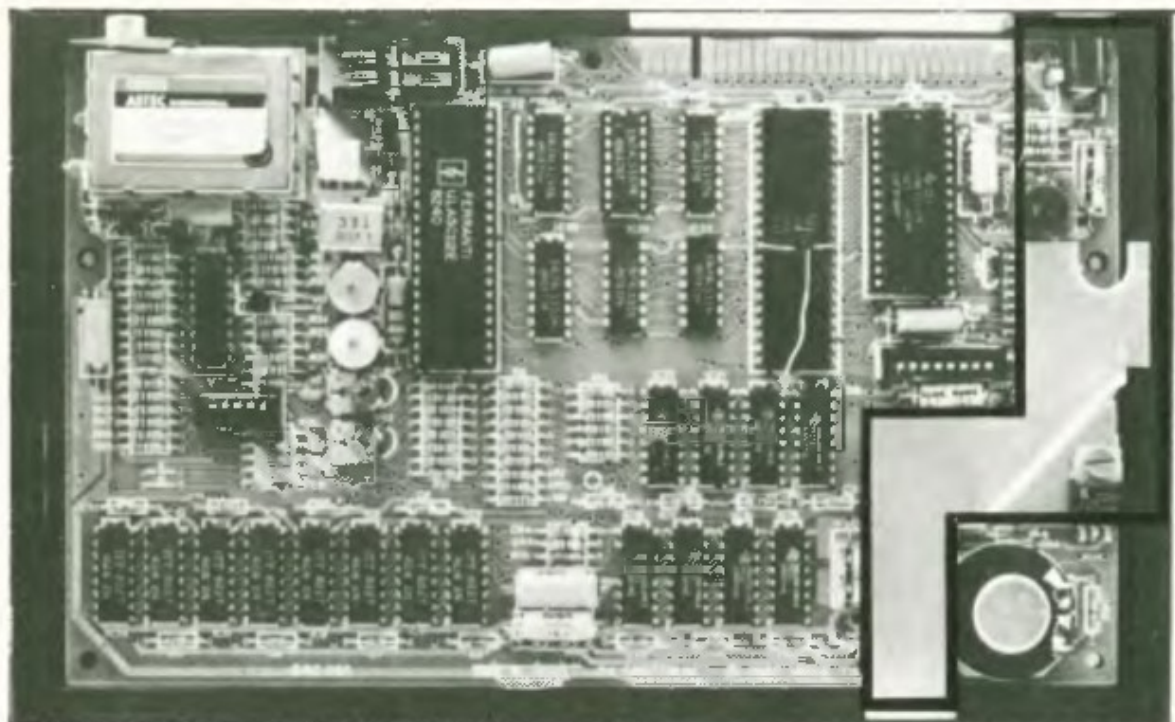
So far, transfer of information between the CPU, memory, ULA, keyboard etc. has been taken for granted. How is it actually done? All data is transferred via the data bus. The type of transfer being carried out is defined by various control signals on the control bus. For example, the CPU sends out a read signal if it wishes to read some data from the ULA or memory. This tells the ULA or memory to send some data to the CPU. If the CPU wishes to output something to be stored in memory, it sends out a write signal. This tells the memory that it must store the data from the data bus. So that the memory knows where it should store the data, the CPU also supplies an address on the 16 bit address bus. The address bus therefore allows the CPU to send or read data from up to  $2^{16} = 65536$  different memory locations.

The buses can only deal with the two logic states 0 or 1 on each of their lines. In practice, these logic states are represented by voltages. By convention (so that most modern computer integrated circuits are compatible with one another), logical 0 is represented by a voltage between 0 volts and 0.8 volts. Logical 1 is represented by a voltage between +2 volts and +5 volts (the maximum supply voltage for logic chips). If the voltage is between 0.8 volts and 2 volts, the signal is in the process of changing from 0—1 or 1—0. All logical data transfers occur within these voltage limits. The chips are designed so that they are not reading data at times when it may be changing.



You should now have a basic understanding of the blocks which make up the Spectrum. The following chapters explain each of these blocks in much greater detail. Each block is considered in isolation, but you should always try to remember how it is connected into the rest of the system. The full circuit diagram of the logic systems which make up the Spectrum can be found in Appendix E. You can refer to this diagram if you have any difficulty in seeing how the individual circuits interconnect.

SSUE 2 BOARD



SSUE 3 BOARD

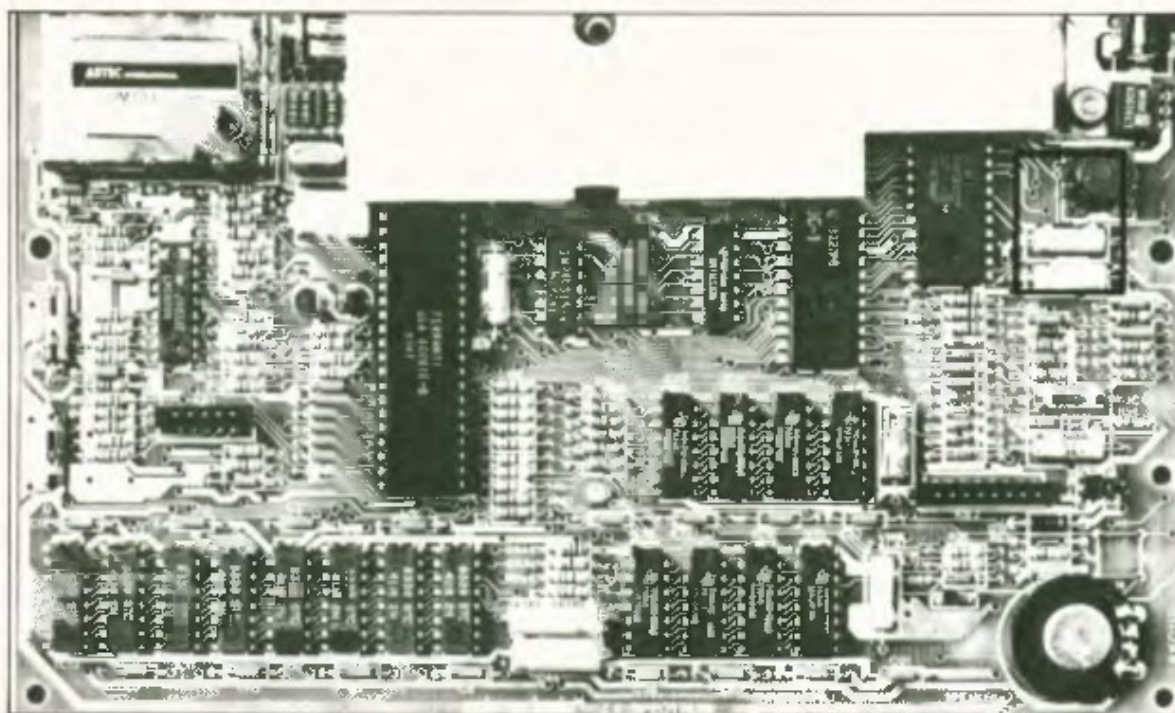


PLATE 2 — PHOTO OF MAIN SPECTRUM BOARD  
WITH POWER SUPPLY OUTLINED



## 3. THE POWER SUPPLY

The power supply of any computer is probably the piece of circuitry most often overlooked by the user. The computer is connected to the mains supply and it comes to life. It is just taken for granted that the correct currents and voltages will be generated. If you are going to add on your own hardware however, a complete understanding of the power supplies is necessary. This chapter therefore takes a close look at that forgotten piece of circuitry and its limitations. The chapter explains how to make the most of your Spectrum power supply. It also provides circuits for additional power supplies for use with your external circuits.

In the Spectrum, all of the power comes in at +9 volts at a current of up to 1.2 amps. This is supplied from a ZX Mains Power Supply. Unfortunately, none of the chips in the Spectrum use a 9 volt supply. Most of the logic, including the CPU operates from a +5 volt supply. The video circuit requires +12 volts supply as well. The video memory chips are most awkward of all, requiring +12v, +5v and -5v all at once!

The problem isn't simply one of producing a supply that is roughly constant at roughly the right voltage for most of the time. The +5 volt supply must be within 1% of +5 volts and the +12v and -5v supplies within 10% of their nominal value all over the circuit all of the time. Even a microsecond's drop in voltage could spell disaster. How are these constant voltages produced?

### THE +5 VOLT SUPPLY

This is the major supply in the Spectrum. In the 48K version it is really stretched to its limit supplying a full 1 Amp of current. Looking at the regulator, you will see that it is bolted onto a large piece of aluminum. The little +5 volt regulator integrated circuit with only three connections to the outside world, contains complex regulation circuitry.

Referring to fig. 2a, the 7805 regulator accepts +9 volts at its IN pin. The internal regulation circuitry then reduces this to +5 volts at the OUT pin. The fact that the input is +9 volts is irrelevant (except for the amount of dissipated power). It could equally well have been anything from +7 volts to +25 volts, the output would still remain at a steady +5 volts. You might wonder where the lost 4 volts has gone to. It is dissipated as heat by the aluminum heatsink. When the regulator is supplying 1 amp, 4 watts have to be dissipated by the heatsink (about one quarter of the heat from a small soldering iron!). That's why the Spectrum soon gets quite hot after it is switched on.

### THE +12 VOLTS SUPPLY

The 5 volt supply was relatively easy to produce. 4 volts were simply dropped by the regulator. Producing the +12 volts with only +9 volts available is rather more complex. Referring to fig 2b, TR5, TR4 and their associated components produce the 12 volt supply. TR5 forms a current feedback for the oscillator formed by C43, R61, L1 and TR4 (the main power drive transistor). Operation of the circuit relies upon the induced reverse voltage across L1 which occurs on every cycle of oscillation. This reverse voltage pushes the collector of TR4 up above 9 volts to a maximum of about +13 volts. At this level, D15 conducts to charge up the +12 volt supply capacitor C44. C44 then discharges to provide a constant



12 volts to the memories and video circuit when D15 is not conducting. If the 12 volt supply starts to fall too low, this causes TR5 to conduct more. The oscillation frequency of the oscillator then increases, boosting the voltage back up to its original level.

### THE -5 VOLT POWER SUPPLY

Again refer to fig 2b. The -5 volt part of the circuit consists of C46, D11, R55, D12, D16, R54 and C47. This circuit operates on the "charge pump principle". Remember that the collector of TR4 in the 12 volt circuit is oscillating rapidly between about +13 volts and 0 volts. When rising up to 13 volts, C46 is charging via D11 up to a maximum of about 12 volts (0.7 volts are dropped across all silicon diodes during conduction). When the collector of TR4 drops to 0 volts, this pushes the negative plate of C46 down to -12 volts. C47 then charges via D12 and R55 from C46. The voltage across C47 is held constant at -5 volts by the zener diode D17. C46 is then charged up again on the next oscillator cycle and so on.

### CURRENT LIMITS

Currents which can be drawn from the Spectrum's own internal power supply for use by external circuits are fairly limited. For the 16K Spectrum, about 300 mA extra can be taken from the +5 volt supply. The absolute maximum limits are rather difficult to define. Generally, the more you take from the power supplies, the worse the regulation becomes. Bad regulation can cause the computer to crash more frequently, but shouldn't cause any damage provided that you stay within the limits recommended above. For 48K Spectrum owners, it is advisable to add an extra +5 volt supply for all external circuits. The current limits on the other supplies are similar to those for the 16K version.

### THE -12 VOLT SUPPLY

You may now be wondering what has happened to the -12 volt supply. It is shown on the edge connector in the Sinclair BASIC manual. However, it isn't shown on the main circuit diagram. In fact, it doesn't exist. The '-12 volt' connection was wrongly labelled. It should really be 'unregulated -12 volt', since it is connected to the collector of TR4. Luckily, it is very easy to use this oscillating -12 volt supply to produce a -12 volt supply, with the addition of a few common components.

Referring to fig 2c, you will almost instantly recognise a circuit very similar to that for the -5 volt supply in the Spectrum. It works on exactly the same charge pump principle. No zener diode has been incorporated in this circuit, because the 100 uF capacitor charges up to 12 volts anyway and is therefore at the correct voltage. Regulation of this type of circuit is never very good under load. The voltage will rise to about -10.5 volts if you take 20mA from the circuit, so this is the recommended maximum current. If you decide to build the circuit then make sure that the two capacitors are rated at 16 volts or above. Apart from this, none of the component values are critical.

### EXTRA +5 VOLT SUPPLIES

For all but very low current circuits on the 48K Spectrum (such as the simple resistor and switches type of experiments in this book), you will have to add an external power supply. The same will apply to the 16K version if you want to draw more than 300 mA from the +5 volt supply. In either case, you have two options:



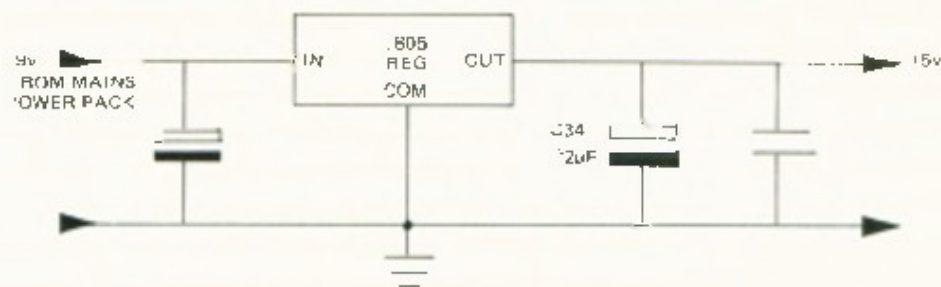


FIG 2a - +5volt POWER SUPPLY CIRCUIT DIAGRAM

10P  
V  
650/1  
(27X) 213  
BC

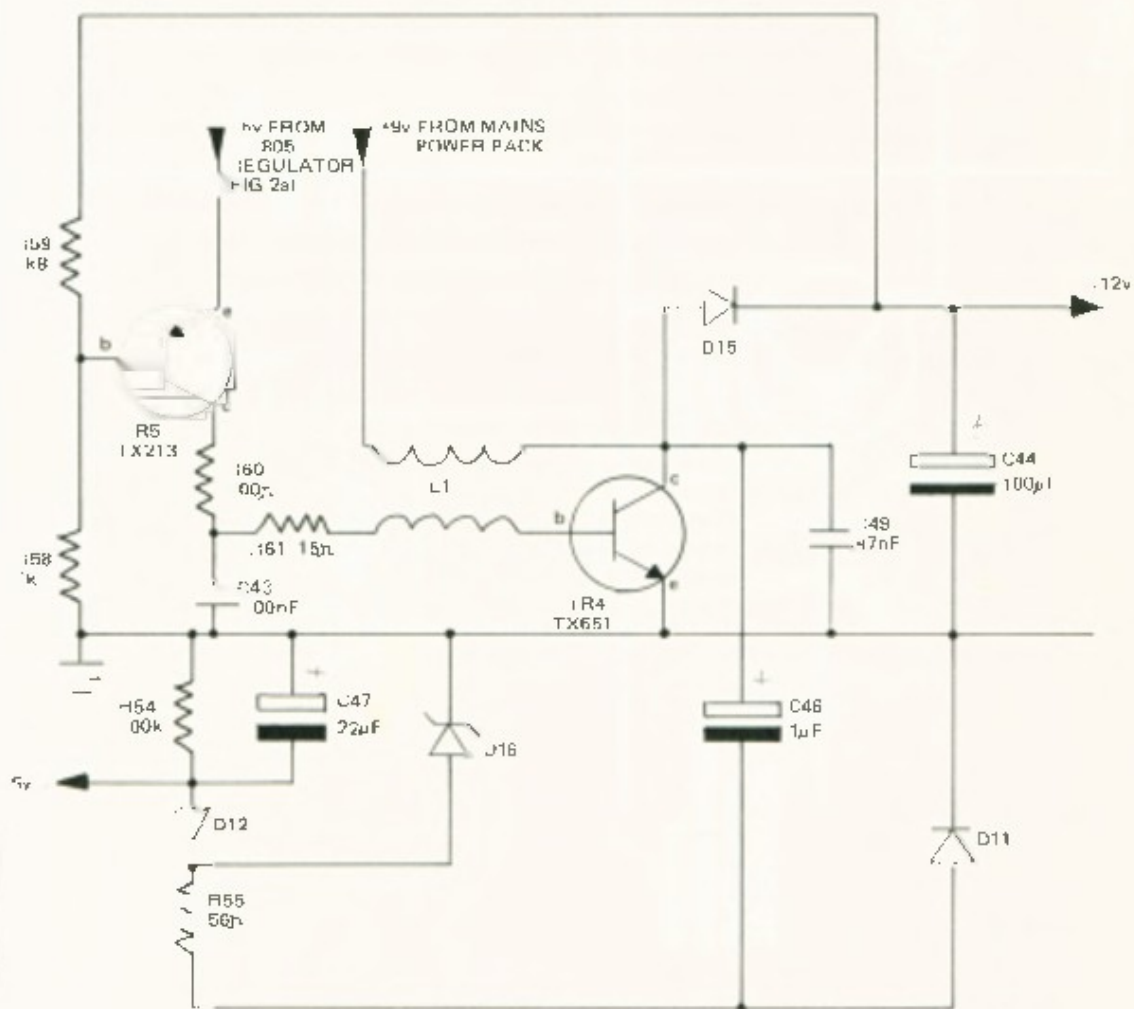


FIG 2b - +12volt AND -5volt POWER SUPPLY CIRCUIT DIAGRAM

10  
416  
-5V  
Pin 1

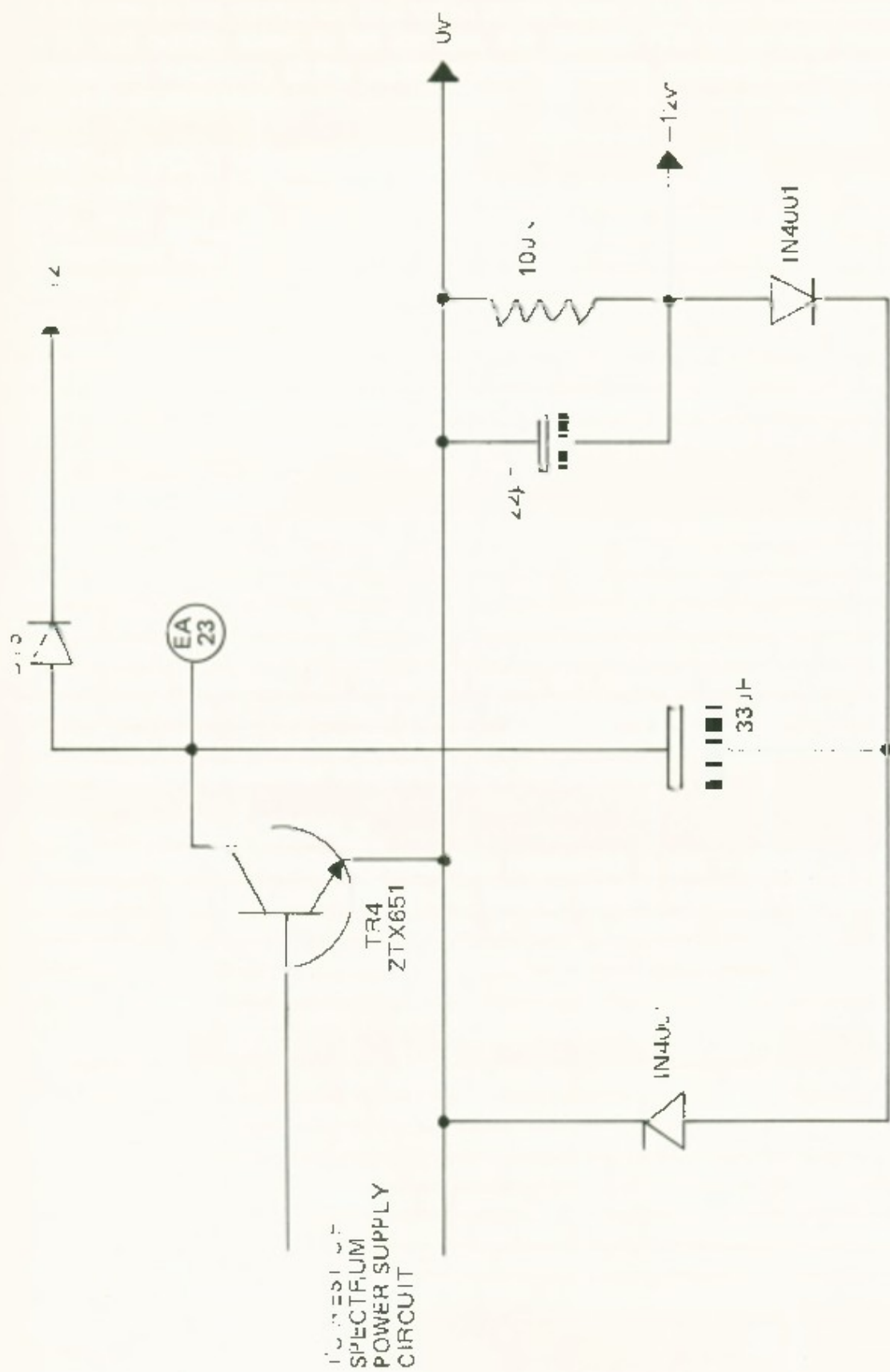


FIG 2c — ADDITIONAL CIRCUIT FOR -12V SUPPLY



### Using the ZX Power supply

There will be some extra current capacity available from your +9 volt Spectrum supply, especially if the +12 volt supply is not fully loaded. The author found it quite acceptable to obtain up to 500 mA more from the ZX supply. The circuit for doing this is shown in fig 2e. Only that part of the circuit to the right of the figure is required. This circuit is basically identical to the one used inside your Spectrum. Connect the 7805 regulator as shown in fig 2d, its input pin being connected to +9 volts. This pin diagram shows a view looking down onto the black plastic part of the regulator, which has the device number stamped on it.

Do not forget to bolt a heatsink onto the regulator. Small vaned heatsinks of a suitable type are available from most good electronic stores. A cheap alternative is to bolt a piece of aluminum onto the regulator. A piece similar in size to that in the Spectrum will do nicely. C1\* in fig 2e should be included to help regulation. 120  $\mu$ F at 16 volts will do. Another 22  $\mu$ F capacitor should also be included across the output. It is advisable to scatter some 0.1  $\mu$ F decoupling capacitors around as well. About one for every two chips should be sufficient. The +5 volts provided by this circuit should not be connected to your main Spectrum +5 volt supply rail. Only the 0 volts connection should be made to all chips.

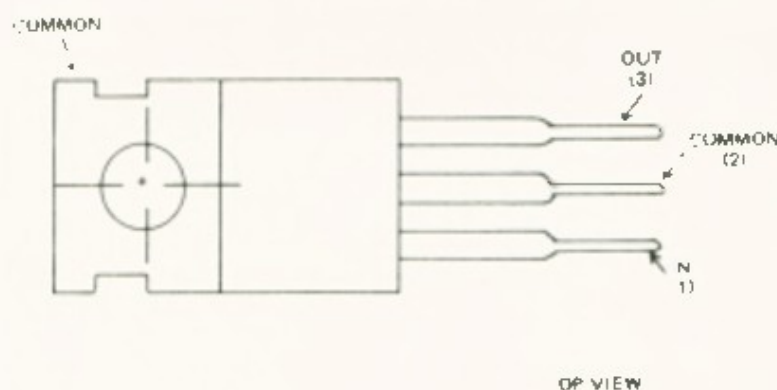


FIG 2d - 7805C 1 AMP/5 VOLT REGULATOR PIN CONNECTIONS

## 7. Using an extra mains supply unit

The full circuit is shown in fig 2e and will provide up to 1 Amp at +5 volts. Instead of connecting pin 1 of the regulator to the Spectrum +9v supply, as in the previous example, it should be connected to the new +9 volt supply. This extra +9 volt supply consists of a 240v to 9 volt at 1 amp mains transformer, a 1 amp bridge rectifier (BR1) and a large 1000uF +25 volt smoothing capacitor C1. You must ensure that the mains transformer is safely wired up. Build it into an earthed metal case. The 7805 can then be bolted to the case, which will act as the heatsink. Accessible mains connections must be avoided to eliminate the possibility of an electric shock. Apart from the differences mentioned above, the circuit is identical to that described in section 1.

You should ensure that the mains supply is connected to your extra power supply at the same time as the Spectrum. If power is not applied simultaneously, some chips could be damaged, because they don't like to have voltages applied to any of their pins without power being applied to them.

## DECOUPLING CAPACITORS

The power supplies are distributed all over the main Spectrum board. Therefore, if a chip on the far side of the board from the power supply suddenly requires extra current, a localised drop in voltage could occur. If the voltage drops too low (even for a microsecond), it could be enough to destroy essential data and programs stored in memory. Supply 'decoupling capacitors' are therefore placed at strategic positions around the board. They are able to smooth out the supply by providing large currents for short periods of time. The voltage is therefore held approximately constant. C1—C8 around the memory chips are there for this reason, as are the other capacitors around the board connected across the supply rails. You should put some around any external circuits which you build as well. One 0.1uF capacitor for every two chips should be enough.



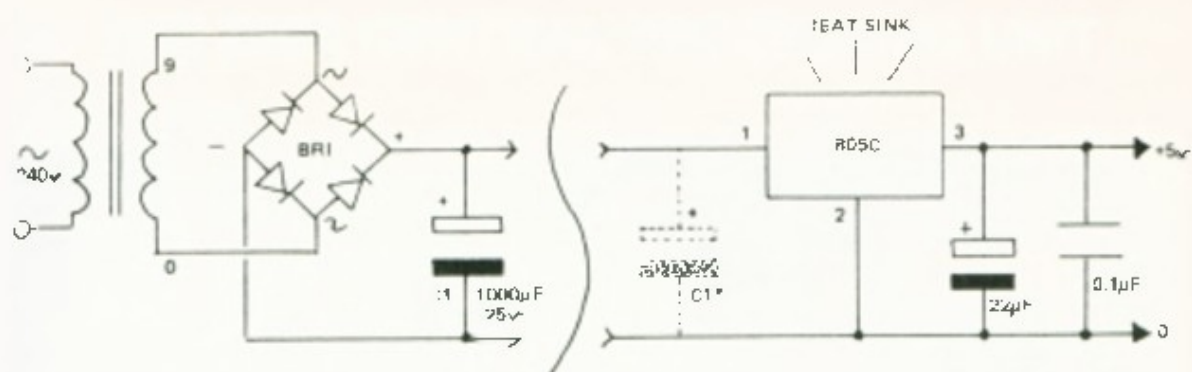
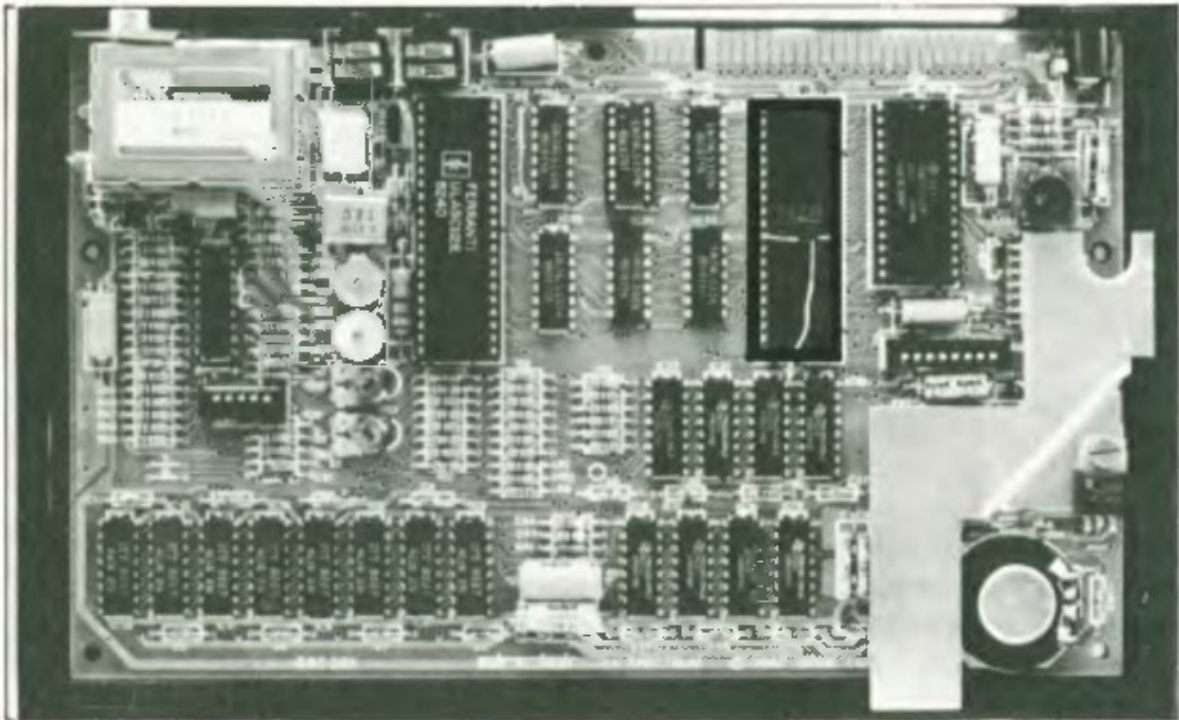
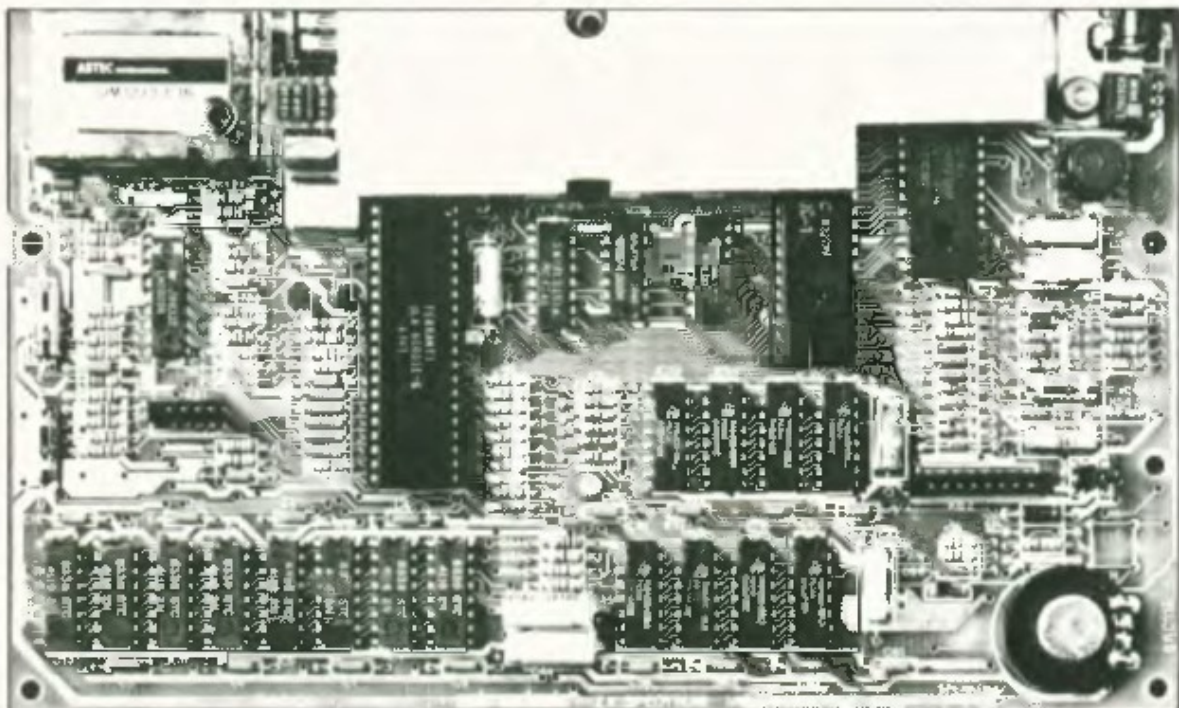


FIG 2e — ADDITIONAL +5 VOLT POWER SUPPLY

SSUE 2 BOARD



SSUE 3 BOARD



'LATE 3— PHOTO OF MAIN SPECTRUM BOARD  
WITH Z80A CHIP OUTLINED



## 1. THE Z80A CENTRAL PROCESSING UNIT

This is the microprocessor chip itself. Housed in its 40 pin plastic package, this chip is the computing centre of your Spectrum. It is the big chip labelled IC2 in appendix D. The CPU is able to perform several basic functions. These are:

- Read data from memory
- Write data to memory
- Read data from an input/output (I/O) device
- Write data to an I/O device
- Perform arithmetic and logical operations on data

What the Z80A does at any particular time is determined by the instructions which it obtains from memory. There are 158 different types of instructions which the Z80A can understand. Programming the Z80A in machine code is a complete subject in itself. You should look at a specialised book on programming if you wish to know more.

You may have seen other computers using the Z80 chip, and wondered what the difference between it and the Z80A is. Both are identical in their operation.

The only difference is that the Z80A can operate at speeds up to 4MHz whereas the Z80 can only operate at speeds up to 2.5MHz. More recently, a new chip called the Z80B has been introduced. This is still the same as the Z80 chip, but can operate at speeds up to 6MHz.

As was mentioned earlier, the Z80A is an 8 bit microprocessor. It transfers data 8 bits at a time. The first microprocessor chips were 4 bit devices, but now some microprocessors are 16 bit or even 32 bit devices. They tend to be rather faster than the 8 bit ones because they can transfer more information in the same time. However, they are much more expensive, so the 8 bit micros are more popular in home computers.

The Z80A clock signal runs at 3.5 MHz (except when accessing video/program memory — see chapter 8) in the Spectrum. All CPU operations are referenced relative to this clock signal, so the 0—1 and 1—0 transitions must be very fast to ensure that all operations start each cycle at exactly the same point. This is why

TR3 and its associated resistors and diode are required. The 3.5 MHz clock signal from the ULA does not have a sufficiently fast logic 0 to logic 1 transition. Since all internal CPU circuits may start at slightly different voltages in the +0.8 volt to +2 volt range, if the time for the clock signal to go from +0.8 volts to +2 volts is significant (in nanoseconds!), different parts of the internal circuitry may start at different times. If there is a difference in start times, the Z80A may malfunction. TR3 ensures that the transition at the CPU clock pin is fast enough.

### 80A CPU PIN DESCRIPTIONS\*

A0 — A15 The address bus — tristate outputs (ie 0 or 1 or floating. When loading, another device can provide the address bus instead of the CPU). This bus provides  $2^{16} = 65536$  different addresses for memory data exchanges or 256 input/output device addresses (the lower eight address lines are used during I/O). The CPU can also refresh memory (see refresh pin) by providing

Note that signals with a bar over them such as  $\overline{RD}$  are 'active low'. This means that a read is occurring when  $\overline{RD} = 0$ . Normally, 1 = true and 0 = false but all Z80A control signals with a bar over them are inverted so that 0 = true and 1 = false.



valid refresh addresses on the lower 7 bits of the address bus.

**D0 — D7** The data bus — these eight tristate bidirectional lines are used for data input and output transfers with the CPU. Transfers take place between memory or peripheral devices and the Z80A.

**M1** Machine cycle one — Output active low. The signal means that the CPU is currently getting the Op-code for the next instruction to be executed from memory. **M1** also occurs with **IORQ** to indicate an interrupt acknowledge cycle.

**MREQ** Memory request — tristate output, active low. This signal indicates to the memory that the address bus now contains a valid address for a read or write operation. This signal is required to distinguish between a memory or input/output operation. For example, **MREQ**, **RD** and **ROMCS** must all be active before a read from the BASIC ROM can take place (see fig 5).

**IORQ** Input/output request — tristate output, active low. Indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. This signal is required to distinguish between an input/output or memory operation. Note that **IORQ** and **MREQ** will never be active at the same time.

**RD** Read — tristate output active low. Indicates that the CPU wishes to read from memory or an I/O device. The addressed device should use this signal to put the relevant data onto the CPU data bus.

**WR** Write — tristate output, active low. Indicates that the CPU data bus holds data to be stored at the addressed location, or output to the selected I/O port. Memory should use this signal to store the data from the data bus.

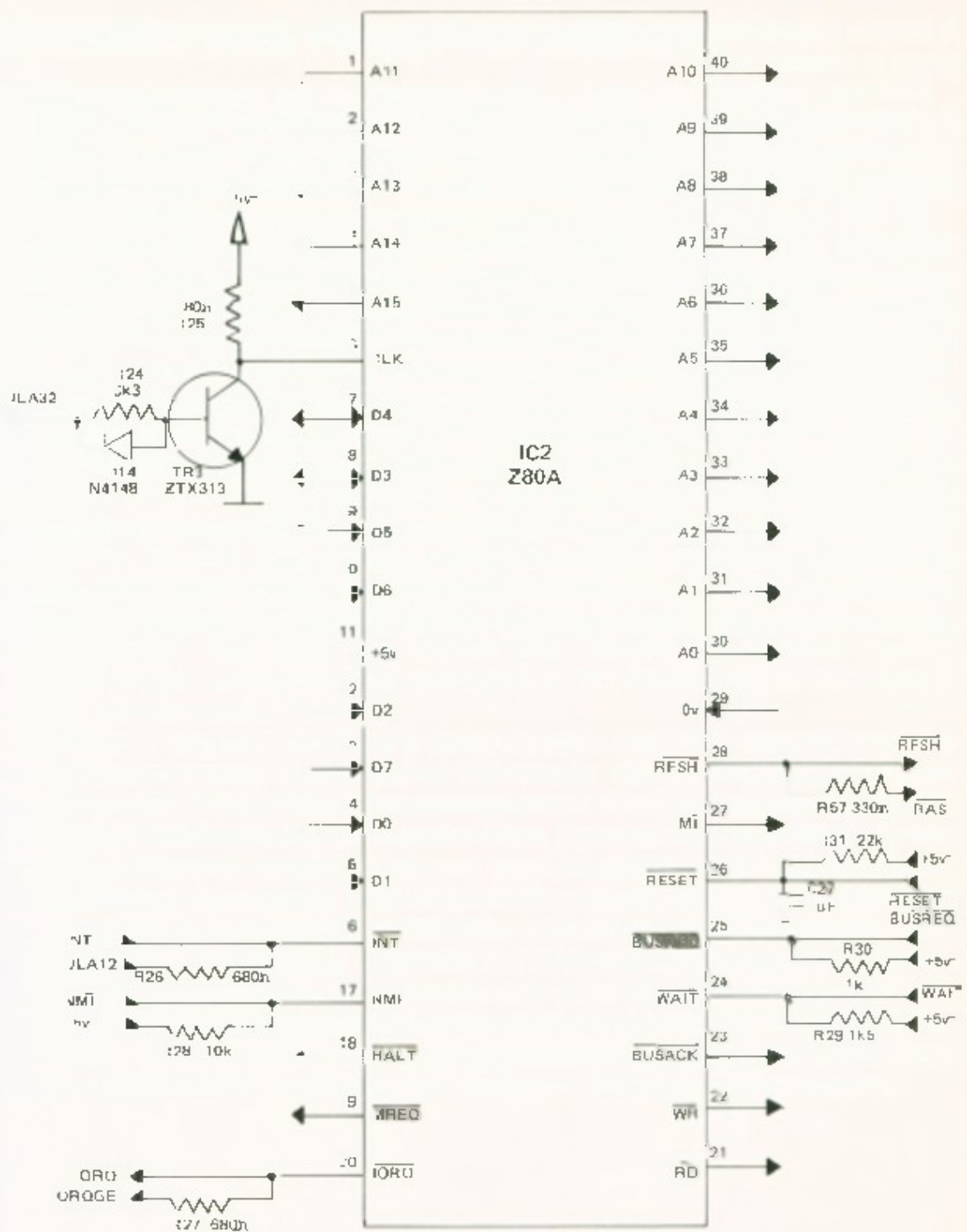
**FSH** Refresh — output active low. Indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories. This is required by dynamic memory chips so that they do not forget. If they are not fully refreshed at least once every two milliseconds, there is a danger that memory will be lost. When this signal is present, the upper 8 address bus lines contain the CPU register contents. This leads to a fault in the Spectrum hardware. By setting the register to any value between 64 and 127, interesting contention for the video RAM occurs. A15 and A14 from the CPU are selecting the 16K of RAM which the ULA normally has priority over. In this case, although the relevant address occurs with **MREQ** active, no **RD** or **WR** signal occurs because it is a refresh. This combination of signals seems to confuse the ULA so that it doesn't stop the CPU clock properly. A BASIC program which will illustrate this problem is included in chapter 8 on the ULA.

**HALT** Output active low. Indicates that the CPU has executed a software HALT instruction. It awaits an interrupt from another device before it will continue operation. There is a simple experiment illustrating the operation of a HALT in chapter 14.

**WAIT** Input active low. Used by slow memory or I/O devices to tell the CPU that they are not yet ready for a data transfer. The CPU sits back doing nothing until the slow device indicates that it is now ready.

**INT** Interrupt request — input, active low. This signal can be generated by external devices to make the CPU run a special machine code program somewhere in memory. If the internal software controlled interrupt enable flip-flop is enabled, the CPU will accept and acknowledge the interrupt. A flip-flop is equivalent to one bit of memory. The CPU uses this bit of its internal memory to remember whether or not it should accept interrupts from other devices. For more details on the use of interrupts, you should refer to a specialised book on the Z80.





IG 3 - Z80A CENTRAL PROCESSING CHIP

A simple experiment showing you how you can make use of the interrupt line is given in chapter 14.

**NMI** Non-maskable interrupt — input operated on the negative going edge of the interrupting signal. The NMI is always accepted by the CPU at the end of its current instruction. It forces the CPU to run the program in memory starting at address 102 decimal (66 Hex). See chapter 14 for a simple example of its use.

**RESET** Input active low. This signal forces the CPU to the reset state. In the Spectrum, when power is applied, C27 holds the reset pin low until it has charged up via R31. This allows the rest of the computer to reach an operational state before the CPU starts to run the program in memory from address zero. Details of adding a push button to reset the Spectrum are given in chapter 14.

**BUSREQ** Bus request — input, active low. Used by an external device to request the CPU address bus, data bus and tristate output control signals for its own use. The CPU will hand over control of all its buses as soon as the current machine cycle is terminated. To indicate to the requesting device that control can now be taken, the CPU sets its BUSACK output low.

**BUSACK** Bus acknowledge — output, active low. Indicates to the requesting device that it can now take full control of all the CPU buses.

More information about some of these Z80 signals and how they can be used is included in other chapters. In particular you are referred to chapters 8, 12 and 14.



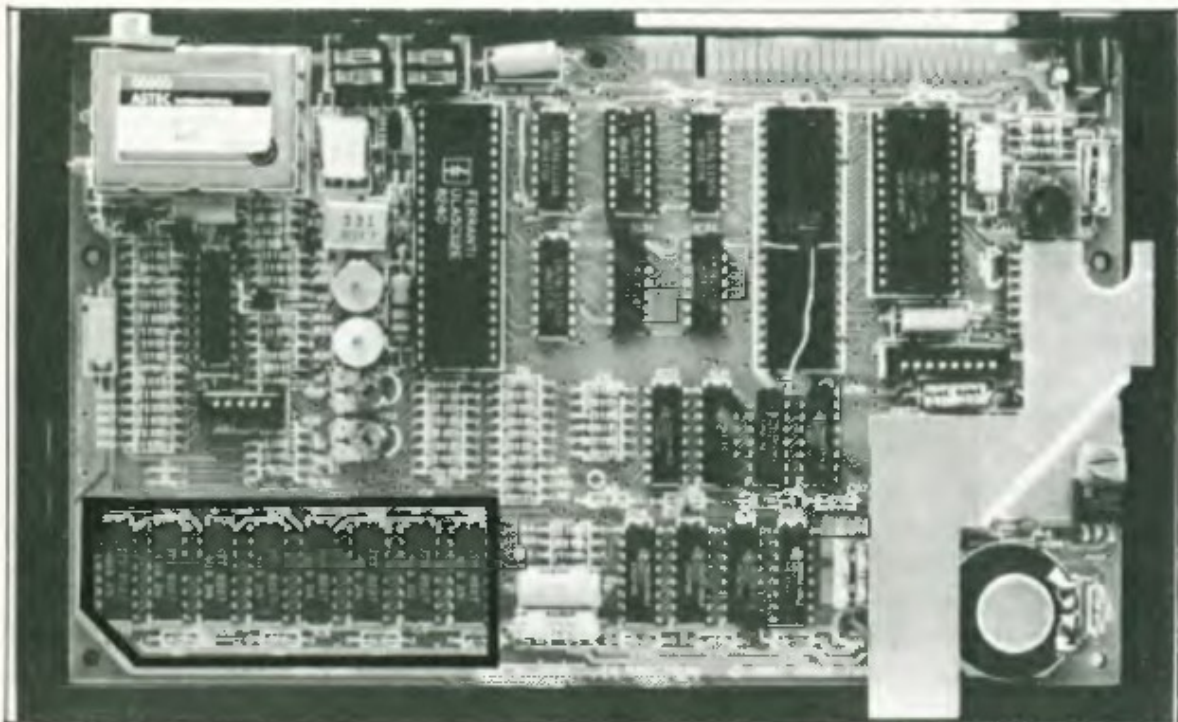
## 5. VIDEO AND PROGRAM MEMORY

This 16K of RAM is supplied on both the 16K and 48K versions of the Spectrum. It contains all of the data for generating the display on the television screen, the various variables required by BASIC, workspace, user defined graphics and your BASIC programs.

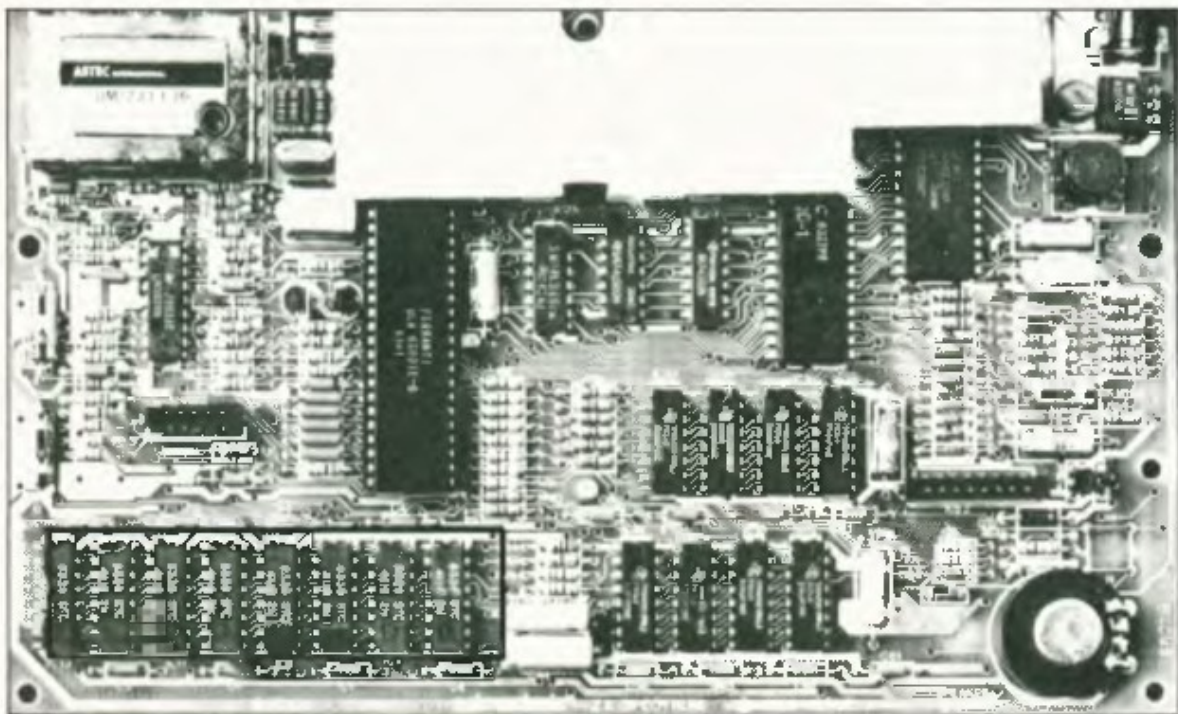
Now refer to fig 4. Each of the memory chips IC6 — IC13 can store  $2^{14} = 16K$  (K just means 1024) bits of information. Eight of them are required to make up the 8 bit data bus. So that the memories can fit into smaller chips with fewer connections, the address lines are multiplexed. This means that first of all A0 — A6 are presented to the memory chip, followed by A7 — A14. These two sets of 7 bits are latched within the memory chip. The memory chip is then able to select the correct location. IC3 and IC4 do this multiplexing, the selected address lines diverted to the memory chip inputs being determined by the state of the select pin 1 on IC3 and IC4. DRAM A0 — DRAM A6 on the ULA can override the outputs from IC3 and IC4 because of the 330 ohm resistors in series with the multiplexer outputs. This override capability enables the ULA to get data for output to the video circuit whenever it is required. You may now be wondering what would happen if the CPU and ULA both wanted to access this memory simultaneously. Obviously they cannot both access two different memory locations at the same time. The resolution of this conflict will be described in chapter 8 on the ULA.

The type of memory used here is called dynamic random access memory. The internal memory array is arranged as 128 rows by 128 columns of cells. The row address strobe (RAS) and column address strobe (CAS) signals are used to latch the relevant addresses into the memory chips. Data can then be written to or read from the addressed location. Each row must be accessed at least once every 64ms otherwise the memory may forget what is stored inside it. In the Spectrum this refreshing is not a problem whilst video output is occurring, because the video memory has to be regularly accessed to produce a continuous video display. During the video field sync, when the memory is not accessed for about 5ms, the normal CPU refresh takes over. Memory which can be read from and written to, but which does not require this continual refreshing procedure is called Static RAM. In both types of RAM, the data is lost when the power is disconnected.

SSUE 2 BOARD

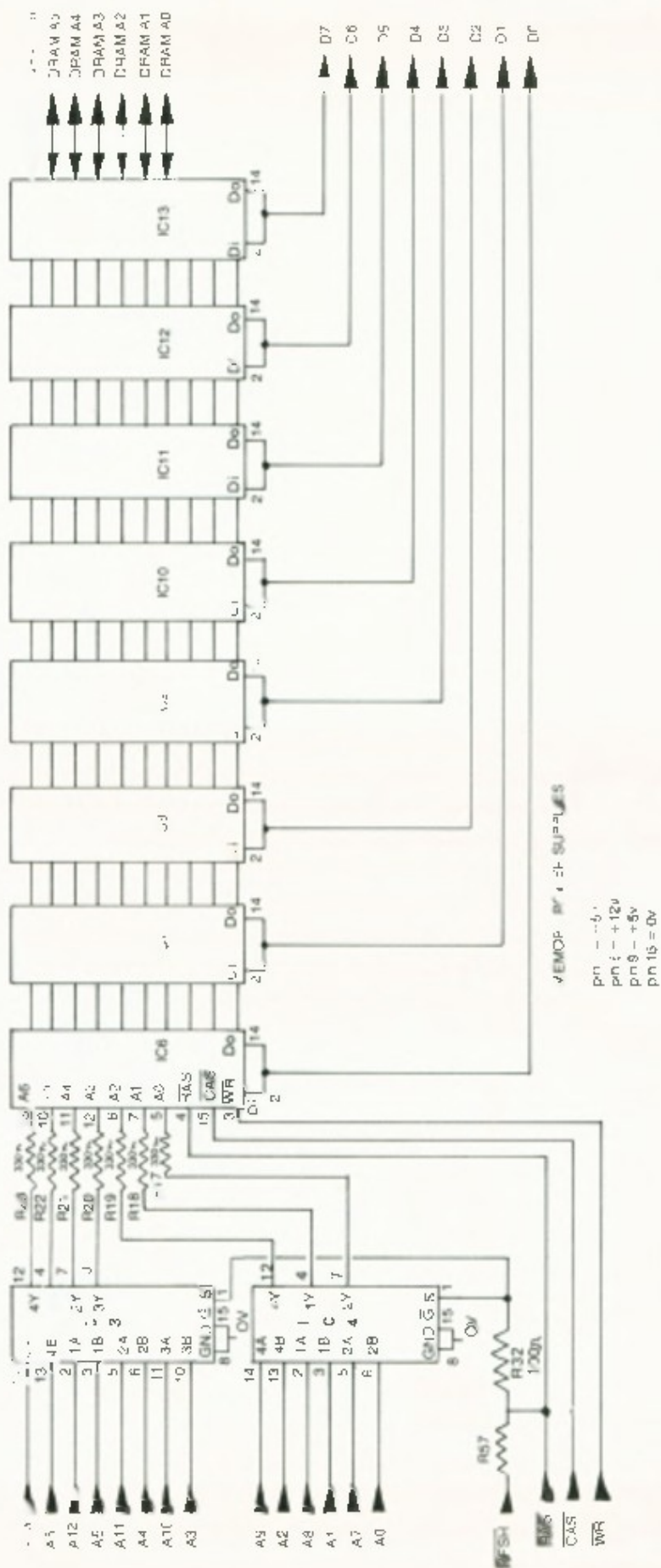


SSUE 3 BOARD

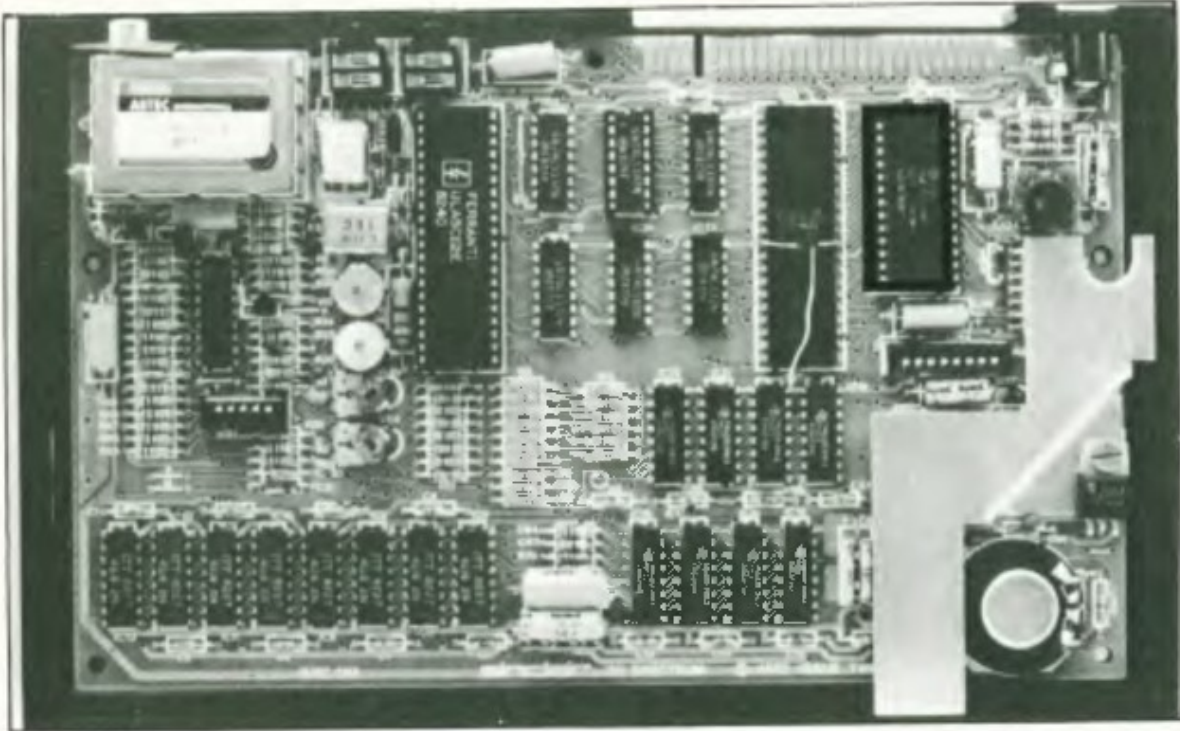


'LATE 4 – PHOTO OF MAIN SPECTRUM BOARD  
WITH RAM OUTLINED

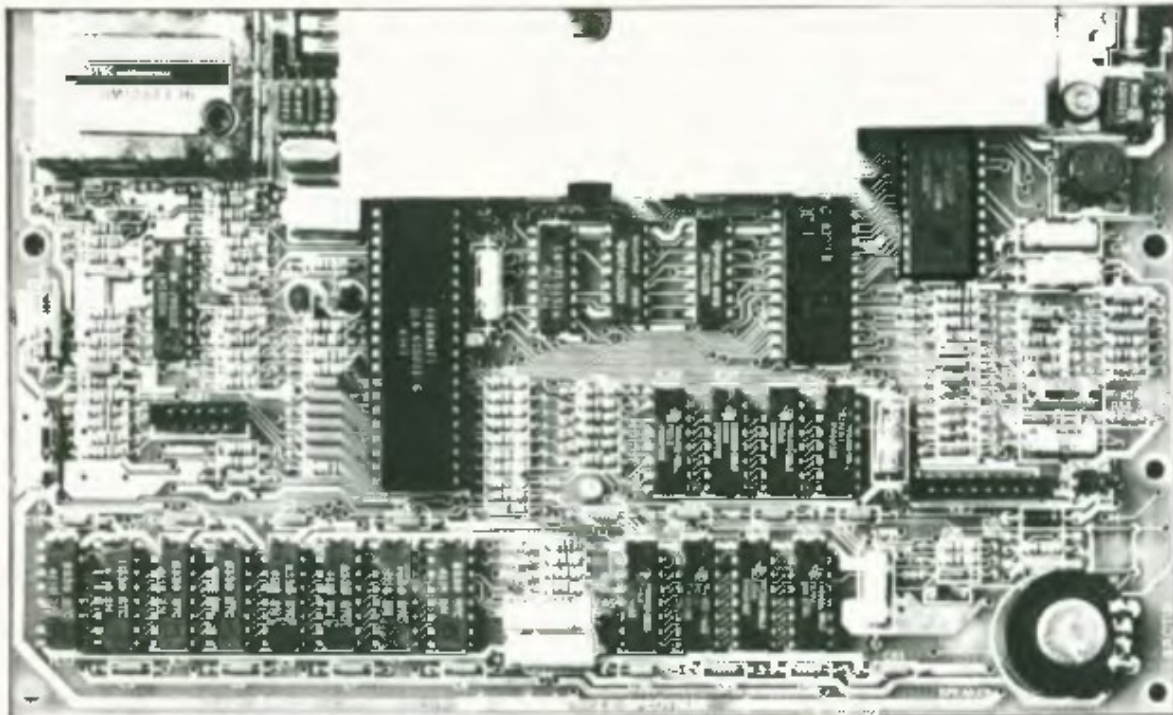




SSUE 2 BOARD



SSUE 3 BOARD



'LATE 5— PHOTO OF MAIN SPECTRUM BOARD  
WITH ROM OUTLINED



## 5. THE BASIC READ ONLY MEMORY

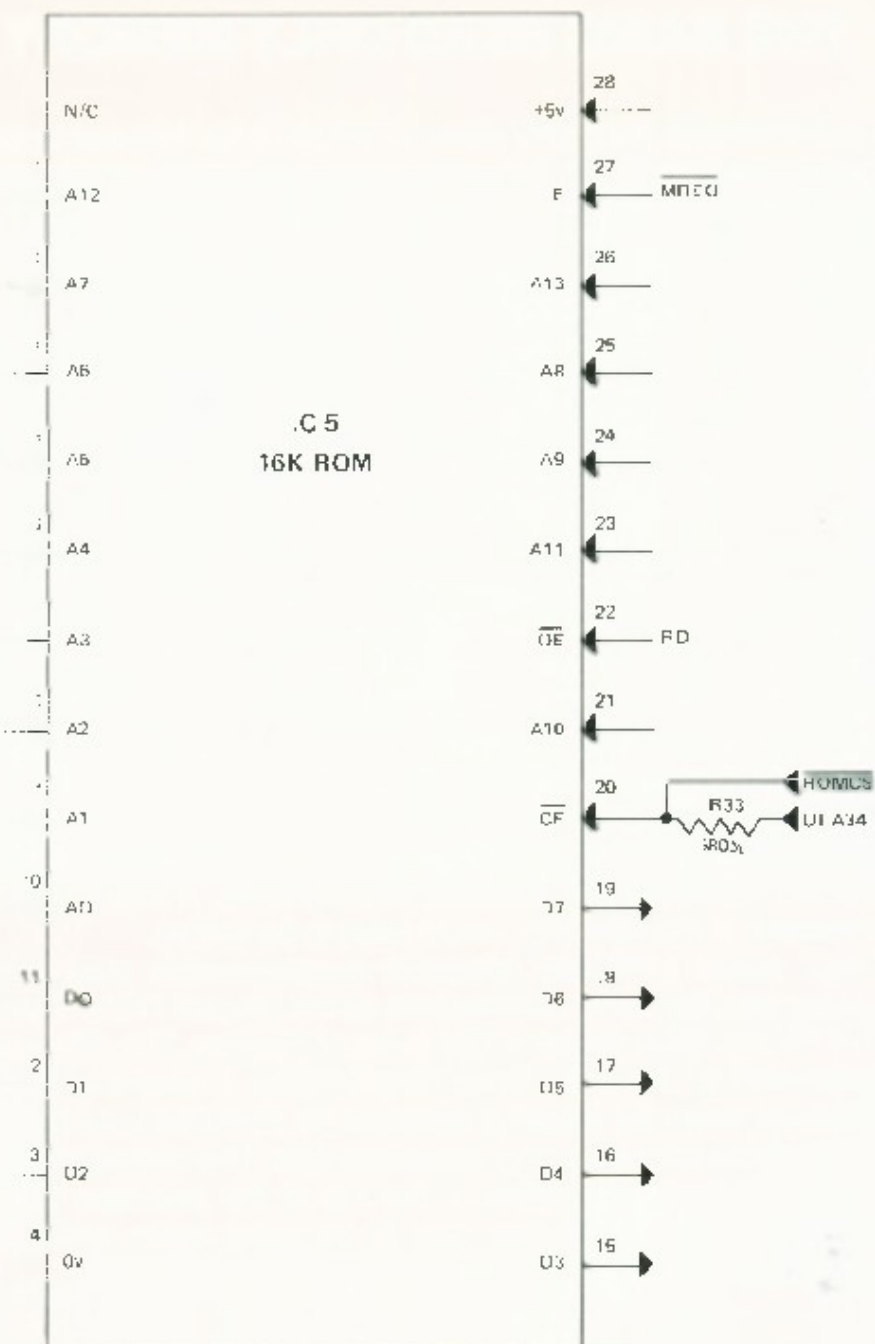
C5 is a 16K byte read only memory (ROM) chip. Housed in its 28 pin package, this chip is provided with 14 address lines, 8 data lines, two chip select pins, one output enable pin to enable data to be read by the CPU and of course the power supply connections. Its pin connection diagram is illustrated in fig. 5.

The ROM contains the program which tells the CPU how to run Sinclair BASIC. This program is embedded into the silicon structure of the chip during manufacture and cannot ever be changed. If this were not the case then BASIC would disappear whenever the power was switched off. Also, since you cannot modify the BASIC program nothing that you can do with software from the Spectrum keyboard can possibly destroy BASIC.

The ROM is positioned from address zero upwards. It has to be positioned here, because when the CPU is reset (when it is switched on), the CPU always runs the machine code program starting from address zero.

The ROMCS connection on the rear edge connector can be connected directly to +5 volts to disable the BASIC ROM. The ROMCS output from the ULA is connected via R33, and is therefore unable to pull the ROMCS signal low when the -5 volt connection is made. This might be useful for future add ons which would have their own ROM or RAM switched in instead. Different languages could then be run from the Spectrum in place of BASIC. Replacing the BASIC ROM in this way with a new chip requires a thorough understanding of the Spectrum hardware and software requirements. A totally new operating system or inclusion in the new chip would have to be written.

Chips called EPROMs (erasable programmable read only memories) are available with the same connections as the Sinclair ROM. The 27128 EPROM is such a device. It too contains 16K bytes of memory. The difference is that it can be programmed by the user. Once programmed, it retains all of the data just like a ROM, even when the power is switched off. EPROMs can be erased so that they can be reprogrammed, using ultraviolet light. This shines directly onto the silicon chip through a glass window in the top of the package. This facility is useful because it allows the same chip to be used again and again with different programs in it.



IG 5 BASIC READ ONLY MEMORY



## THE KEYBOARD

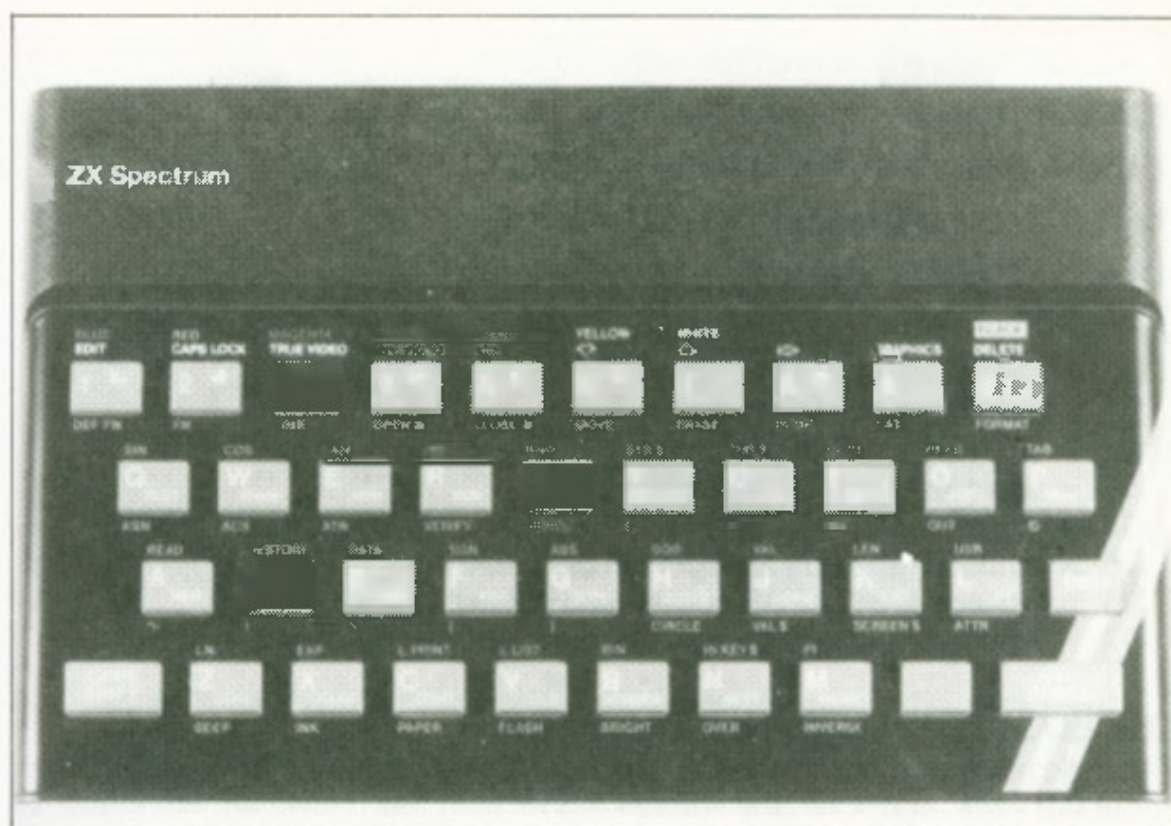


PLATE 6 — PHOTO OF KEYBOARD

The keyboard consists of a five by eight grid of wires, the crossing points of which can be connected by pressing the relevant key (see figure 6). Whenever the BASIC ROM program wishes to find out which key on the keyboard is pressed, it scans each of the eight rows in turn. Each row is selected by setting the relevant I80 address line to logic 0, keeping the remainder at logic 1. Each of the five columns is usually held high by one of the resistors R65-R69. If any of the keys on a particular row are depressed, the associated column input to the ULA will be pulled down to logic 0 instead of its usual logic 1. The key can therefore be read.

Consider for example that key 'D' is pressed. By doing an IN from an address with all lines at logic 1 except for A9, bit D2 of the input byte would be 0, whilst D0, D1, D3 and D4 would all be at logic 1. The keyboard is scanned 50 times per second to see if any of the keys have been pressed, so that everything you type should be noticed by the Spectrum. A keyboard scan is initiated by the ULA interrupting the CPU at the end of displaying each video frame. The CPU then reads the keyboard.

If you wish to scan the keyboard yourself for some reason, the following addresses can be used. Bear in mind that each of these numbers is simply used to set all of these address lines except for the one being used to scan, to logic level 1. Using this method of input, the reading of keys is no longer restricted to 50 times per second. It can be done as often (within limits) or as infrequently as desired.

- "N 32766" uses A15 to read the half row SPACE to B
- "N 49150" uses A14 to read the half row ENTER to H
- "N 57342" uses A13 to read the half row P to Y
- "N 61438" uses A12 to read the half row 0 to 6
- "N 63486" uses A11 to read the half row 1 to 5
- "N 64510" uses A10 to read the half row Q to T
- "N 65022" uses A9 to read the half row A to G
- "N 65278" uses A8 to read the half row SHIFT to V

n the byte read in:

- 0 = logic level at KBD 13 input
- 1 = logic level at KBD 12 input
- 2 = logic level at KBD 11 input
- 3 = logic level at KBD 10 input
- 4 = logic level at KBD 9 input
- 5 = unused (appears as logic 1 in byte read in)
- 6 = voltage level at EAR socket (see below)
- 7 = unused (appears as logic 1 in byte read in)

## THE LOGIC LEVEL AT THE EAR SOCKET

The logic level read in from the EAR socket will in general be 0 for voltages below about 0.6 volts, and 1 for any voltage higher than that. However, the threshold level between logic 0 and logic 1 is slightly different in the Issue 2 and Issue 3 models. The following table illustrates this difference. The voltages at the EAR port are set to four different values controlled by ULA control byte bits D4 and D3.

D4	D3	Voltage	Issue 3 N 57342	Issue 2 IN 57342
0	0	0.4v	191	191
0	1	0.7v	191	255
1	0	3.5v	255	255
1	1	3.8v	255	255

The value read in using IN 57342 uses address line A13 to read the half row of keys P, O, I, U, Y, and the values shown are with none of these keys pressed. The read in values are the same for the two Spectrums, except when D4=0 and D3=1 in the ULA control byte. The Issue 3 then registers the value at the EAR port (D6 in input byte) as a 0. The Issue 2 recognises it as a 1. As it happens, D4 and D3 normally are in this state, so if the keyboard is scanned by a user program (rather than the ROM), bit 6 in the input byte should ALWAYS be masked. Several commercial programs looked to see if a key had been pressed by checking for IN 57342 NOT EQUAL TO 255. This worked on Issue 2 machines, because the value was 255 without a key pressed. However, as soon as Issue 3's came on the market, such software stopped working because it hadn't masked out bit 6!

Details for constructing your own additional keyboard interface which will plug into the rear edge connector are given in Chapter 16.



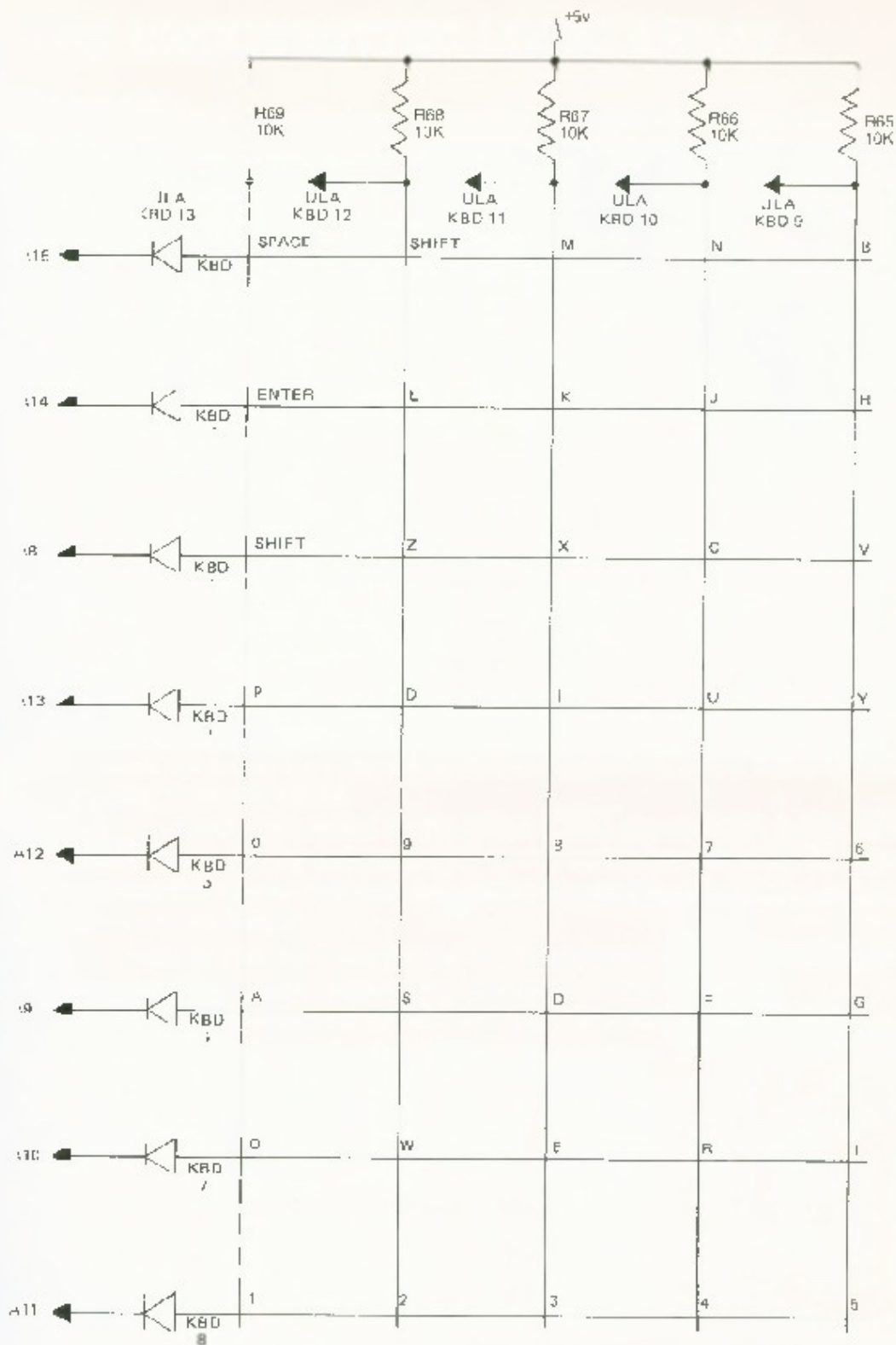
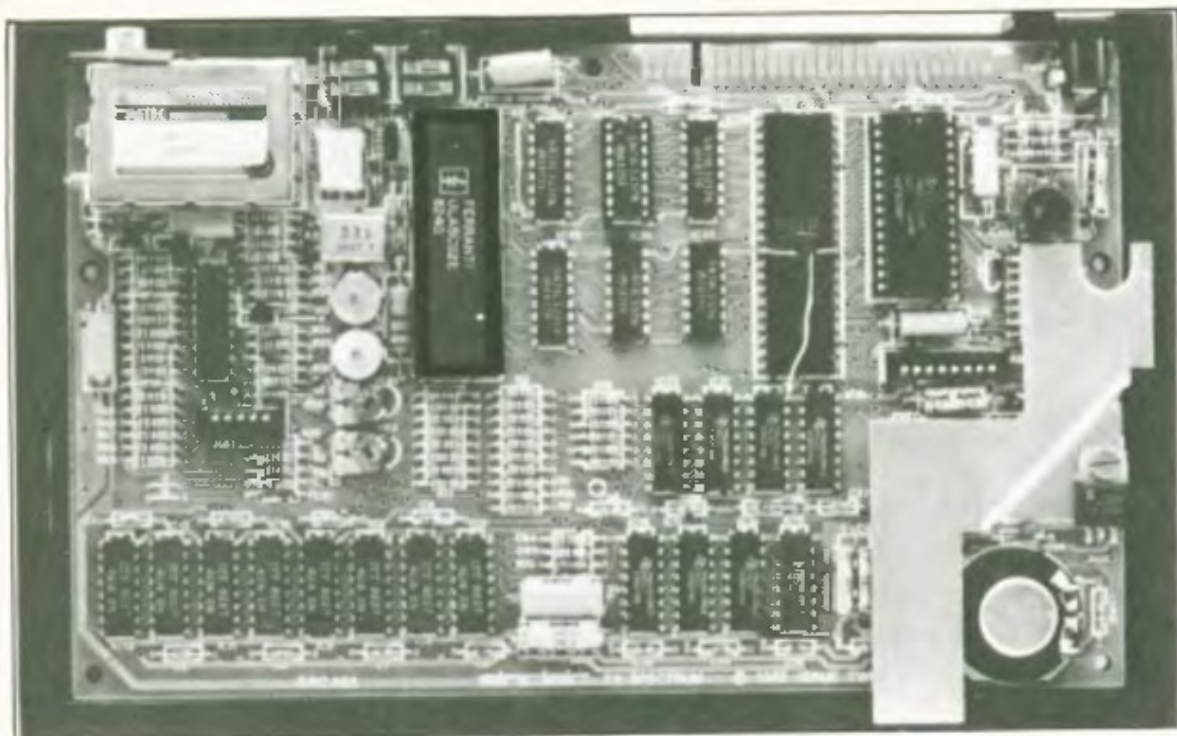
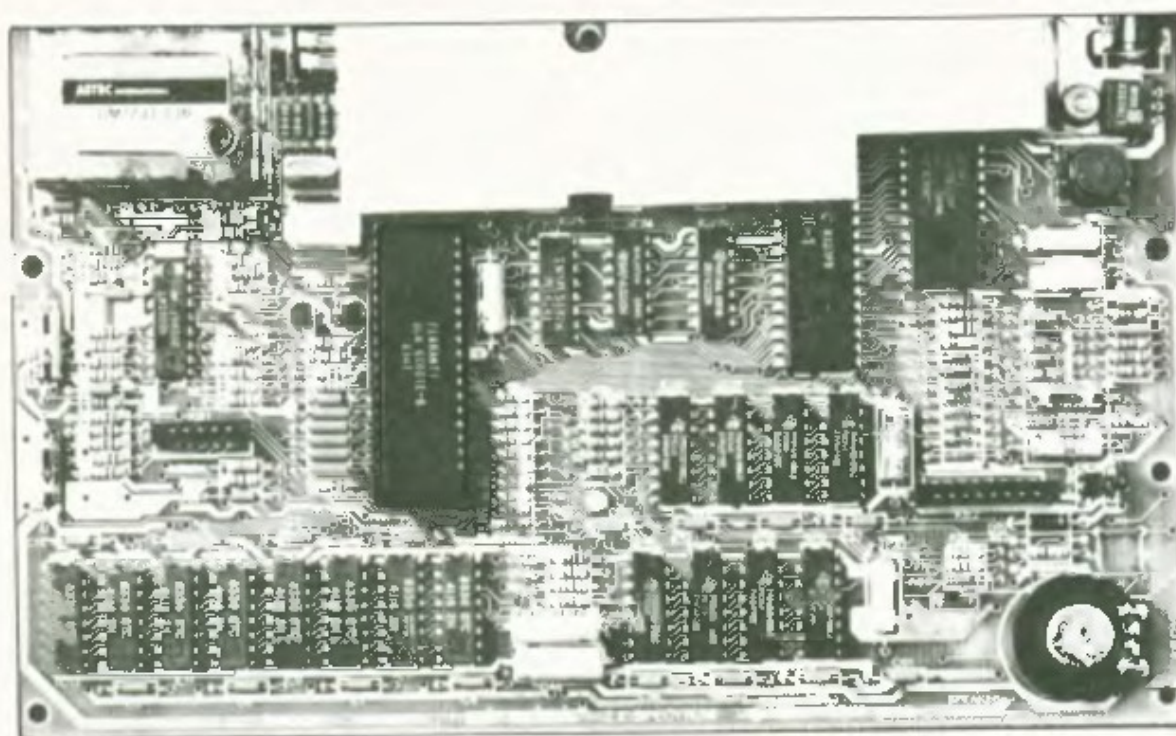


FIG 6 - SPECTRUM KEYBOARD

SSUC 2 BOARD



SSUC 3 BOARD



'LATE 7— PHOTO OF MAIN SPECTRUM BOARD  
WITH ULA OUTLINED



## 3. THE UNCOMMITTED LOGIC ARRAY

The ULA is Sinclair's special chip. It has been designed to replace lots of smaller logic chips which were used in older computers. The way in which it operates is defined when the chip is manufactured, and can never be changed with software. Its function is to take the heavy burden of input and output from the CPU. It performs the tedious output of information to the television all of the time when power is applied to the Spectrum. Apart from this major task, the ULA also deals with output to the buzzer and cassette plus input from the keyboard and cassette.

### ULA PIN DESCRIPTIONS

**DRAM A0 — A6** — DRAM A6 from the memory address multiplexer chips IC3 and IC4. These enable the ULA to determine which address is being selected by the CPU. Also used by the ULA to select the address of video data from the video memory. When used by the ULA as outputs in this way, they are able to override the output from the address multiplexers because of the set of series 330 ohm resistors.

**DRAM CAS** — dynamic RAM column address strobe (CAS) output. Used to latch in the column address for the dynamic RAM from the address multiplexer (see chapter 5 for more details).

**ROMCS** — ROM chip select output enables the 16K ROM chip IC5 whenever the CPU wishes to read from it. The ULA can monitor this with its A14 and A15 address inputs, so it can be outputting video information at the same time as the CPU is reading from the ROM.

**IORQGE** — input connected to the Z80A  $\overline{\text{IORQ}}$  pin via R27. If this  $\overline{\text{IORQGE}}$  connection is held high by connecting it to +5 volts, the Z80A  $\overline{\text{IORQ}}$  signal no longer reaches the ULA. This can be useful for adding extra I/O devices. See chapter 15 for more details.

**RAS** — used as row address strobe on the dynamic RAM chips. (see chapter 5 for more details about refreshing dynamic memories). This pin is also connected to the Z80A refresh pin via R57 (330 ohms). The memories can then be refreshed by the CPU during the video field sync time, when regular accessing by the ULA stops for about 5 ms.

**KBD9 — KBD13** — inputs from the Spectrum keyboard. See chapter 7 for further details.

**B** — blue — yellow colour difference output.

**R** — red — yellow colour difference output.

**Y** — luminance and sync outputs for the video.

**D0 — D7** — 8 bit bidirectional databus. Connected directly to the video memory databus and via 470 ohm resistors to the main system databus. This enables asynchronous operation of the ULA accessing video RAM and the CPU accessing the rest of memory. The computer therefore runs faster than it would if the ULA stopped it every cycle.

**CLK** — 3.5 MHz clock output to the Z80A CPU. This clock may occasionally be stopped by the ULA to prevent the CPU from accessing the video RAM when the ULA is using it.

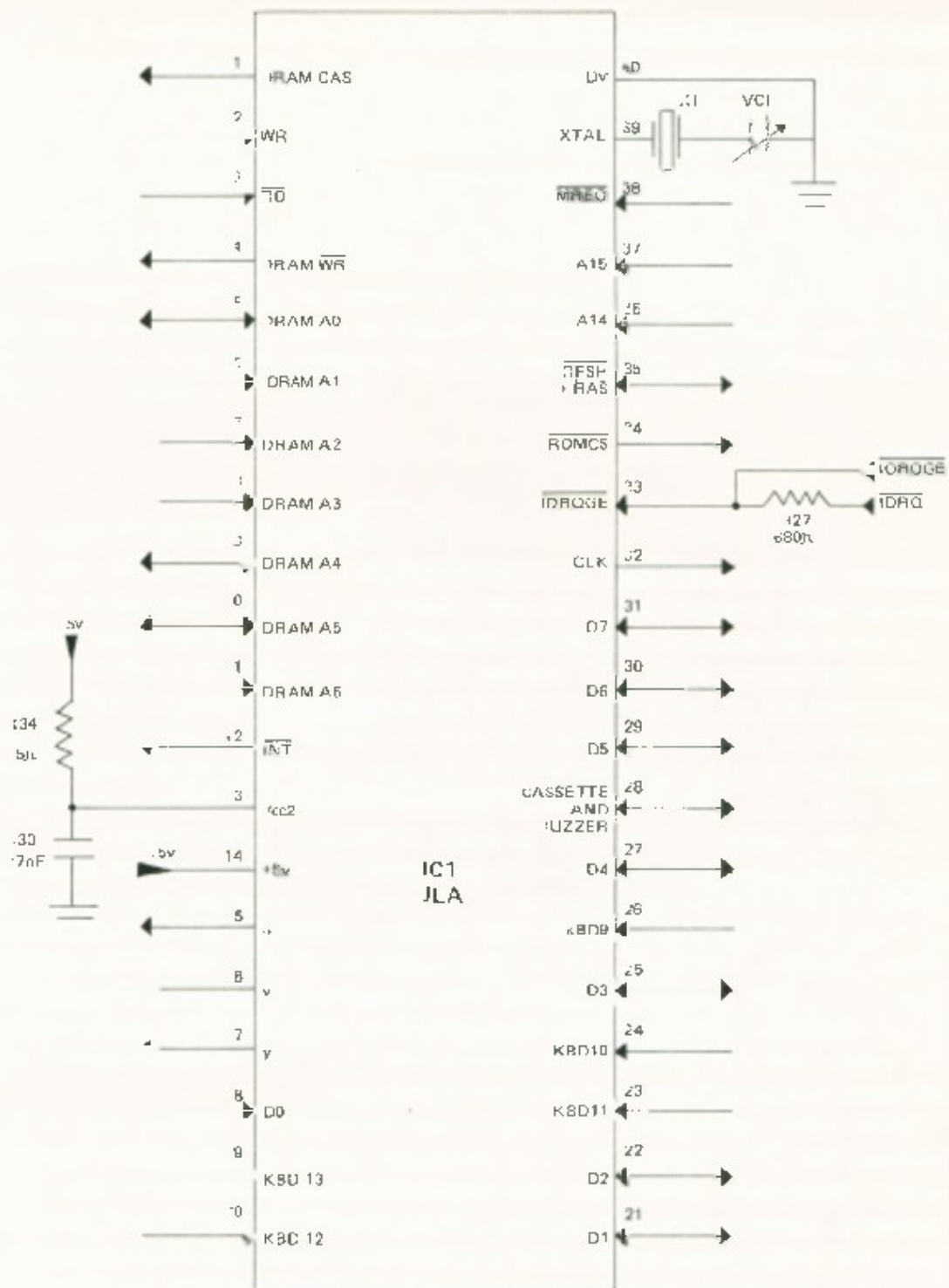
**WR** — tells the ULA when the CPU is writing to some device. If that device happens to be the ULA then the ULA will use this signal to latch the incoming data.

$\overline{RD}$  — tells the ULA when the CPU is reading from a device. If the device is the JLA, data will be output onto the databus for the CPU.

$\overline{WEQ}$  — this input tells the ULA that the address bus now contains a valid address for a read or write operation. This signal is required to distinguish between a memory or I/O operation.

$\overline{INT}$  — interrupt to the CPU operates 50 times per second. Upon receiving this interrupt, the CPU increments a 2 byte counter in memory and scans the keyboard to see if any keys have been pressed.





NOTE — VC1 IN ISSUE 2 IS REPLACED BY C73 IN ISSUE 3

FIG 7a — THE UNCOMMITTED LOGIC ARRAY PIN CONNECTIONS

## THE TRANSISTOR PATCH IN ISSUE 2

Issue 2 Spectrums have TR6 soldered across the Z80A chip. The connections to this transistor are shown in figure 7b. The effect of this patch is to pull the ORQGE input to the ULA high whenever address line zero (A0) is at logic 1. The ULA will therefore only be selected when A0 and IORQ from the Z80A are in the logic 0 state together. In Issue 3 Spectrums, transistor TR6 has been incorporated on the printed circuit board. It operates in exactly the same manner as in issue 2 Spectrums.

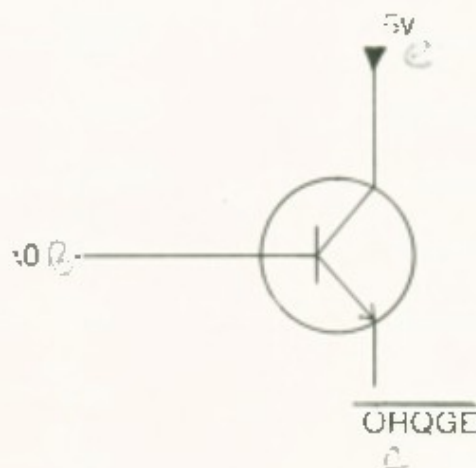


FIG 7b — THE TRANSISTOR PATCH CIRCUIT

## VIDEO OUTPUT

One of the major problems on most home computers is that of producing the video display. Especially for a colour display of the resolution used in the Spectrum, data has to be copied from the video memory to the display at a continuous and fast rate. This poses problems when the CPU wants to read from the video memory at the same time as the ULA. Two devices cannot address different places in a memory chip simultaneously.

Most other computers use one of two methods to resolve this. In the first method the CPU has priority. This produces 'snow' on the screen, representing video information which was not displayed because the CPU was using the memory. Snow is annoying but the CPU runs at full speed. The second method gives the video circuit priority during a television field scan. This requires that the CPU should only be able to operate during the field sync time. Using this method does eliminate snow, but the computer operates very slowly. Neither of these methods is satisfactory.

Our Spectrum uses a very ingenious way to get over the problem. Let us assume that the ULA is accessing video memory. The CPU can simultaneously access the ROM or extra 32K of RAM without any bus contention occurring. The ULA and video memory have their address and data buses separated from the rest of the system by 330 ohm and 470 ohm series resistors respectively. Normally therefore, there are two separate systems operating independently of one other. The ULA outputting video, the Z80A operating BASIC. But then, the CPU may wish to access the 16K of RAM which the ULA is using, because this contains all of the BASIC system variables as well as the video memory. At this



point, the ULA realises what is about to happen (by monitoring A14 and A15), so it hastily stops the CPU clock. The Z80A doesn't notice this because its only way of measuring time is by assuming that its clock input is constant. The ULA can then let the CPU have access to the memory for a few hundred nanoseconds when there is a brief gap in video output.

How does this novel design feature affect any programs which you may wish to write? In BASIC, it will always appear totally transparent to the user, but if you run a machine code program from the affected 16K of RAM, the timing of routines will not be constant. Normally this would not be important, however, routines with a critical timing loop in them, such as a BEEP routine, will not operate properly. The BEEP in BASIC does work properly because the machine code that operates it is running from the ROM.

### A BIT ABOUT INTERRUPTS

If you start writing machine code routines, you may wish to use vectored interrupts. Whenever the CPU is interrupted (eg by a piece of your external hardware), you can make it run a routine whose address can be found at a position in memory pointed to by a 16 bit pointer. This 16 bit pointer is made up from the CPU I register contents and an 8 bit byte supplied by the interrupting device. The I register defines A8 — A15 in this case. Assume now that your interrupt routine address is held in the 16K of RAM used by the ULA. The I register will therefore contain a decimal number between 64 and 127 (A15 = 0, A14 = 1). During every instruction cycle of the CPU, a refresh will occur. Refresh puts out the I register contents onto A8 — A15 and operates MREQ. This combination of signals confuses the ULA into expecting a read to or write from the video memory by the CPU. This doesn't occur so the ULA gets confused. So confused in fact that it omits some video output, causing snow on the screen!

Try running this small BASIC program to set the I register so that you can see the snow for yourself. Remember that values of I between 64 and 127 create snow.

```
10 CLEAR 32499
20 INPUT "Enter the value for I register ";v
30 POKE 32500,62 :REM LD A,v
40 POKE 32501,v
50 POKE 32502,237 :REM LD I,A
60 POKE 32503,71
70 POKE 32504,201 :REM RET
80 LET a =USR 32500
90 GO TO 20
```

Line 10 stops BASIC using memory above address 32499. Lines 30 — 70 put a machine code program into memory. You will be prompted for the value required in the I register. The machine code program then sets the I register when it is called from BASIC at line 80. Return from the machine code jumps to line 90, causing the whole BASIC program to be repeated.

## CLOCKS

The ULA generates its own master clock. This master clock frequency is held constant at 14 MHz by the crystal X1. The 14MHz master clock is divided by 2 to produce the correct video dot frequency of 7MHz. Division by two again reduces the frequency to 3.5MHz which is fed to the Z80A. This 3.5MHz clock is not constant and can be stopped for short periods by the ULA to override the CPU during video memory accesses.

## KEYBOARD AND CASSETTE INPUTS

The keyboard is dealt with in chapter 7. In summary, if you read in a byte from I/O address 254, bits D0 — D4 will hold the logic level at the KBD13 — KBD9 inputs to the ULA. These bits are set to 0 if a key is pressed and to 1 if it isn't. D6 holds the input level at the EAR input from your cassette recorder.

## BUZZER, CASSETTE & BORDER COLOUR OUTPUTS

A control byte can be sent to the ULA by writing to output address 254. Of the eight bits in the control byte, only D0-D4 are used. Bits D0-D2 control the BORDER colour, bit 3 is the MIC output bit and bit 4 is the buzzer output bit.

### BORDER COLOURS AND VALUES FOR BITS 0 — 2

D2	D1	D0	Colour
0	0	0	black
0	0	1	blue
0	1	0	red
0	1	1	magenta
1	0	0	green
1	0	1	cyan
1	1	0	yellow
1	1	1	white

In fact, it can be seen that D0 controls the BLUE output, D1 controls the RED output and D2 controls the GREEN output. All other colours are combinations of these basic primary colours.

### BUZZER AND MIC OUTPUT BITS 3-4

D4 (buzzer)	D3 (mic)	Voltage at ULA pin 28
0	0	0.4v
0	1	0.7v
1	0	3.5v
1	1	3.8v



Figure 7c and figure 7d illustrate the cassette interface and buzzer circuits for the two Spectrums. Looking at the above table, it will be seen that the MIC output bit only generates a voltage change of about 300mV when it is flipped between logic 0 and logic 1. This signal is used to save programs on cassette. However, the voltage level is insufficient to overcome the 0.7 volt drop across D9 (and D10 in issue 2's). The buzzer therefore doesn't operate. When the buzzer output bit is flipped between logic 0 and logic 1, it generates a voltage change of about 3V. This large voltage change is sufficient to operate the buzzer. The buzzer in an issue 3 Spectrum is driven via transistor TR7 from the +9V DC supply, so it doesn't present any additional load to the +5V regulated supply.

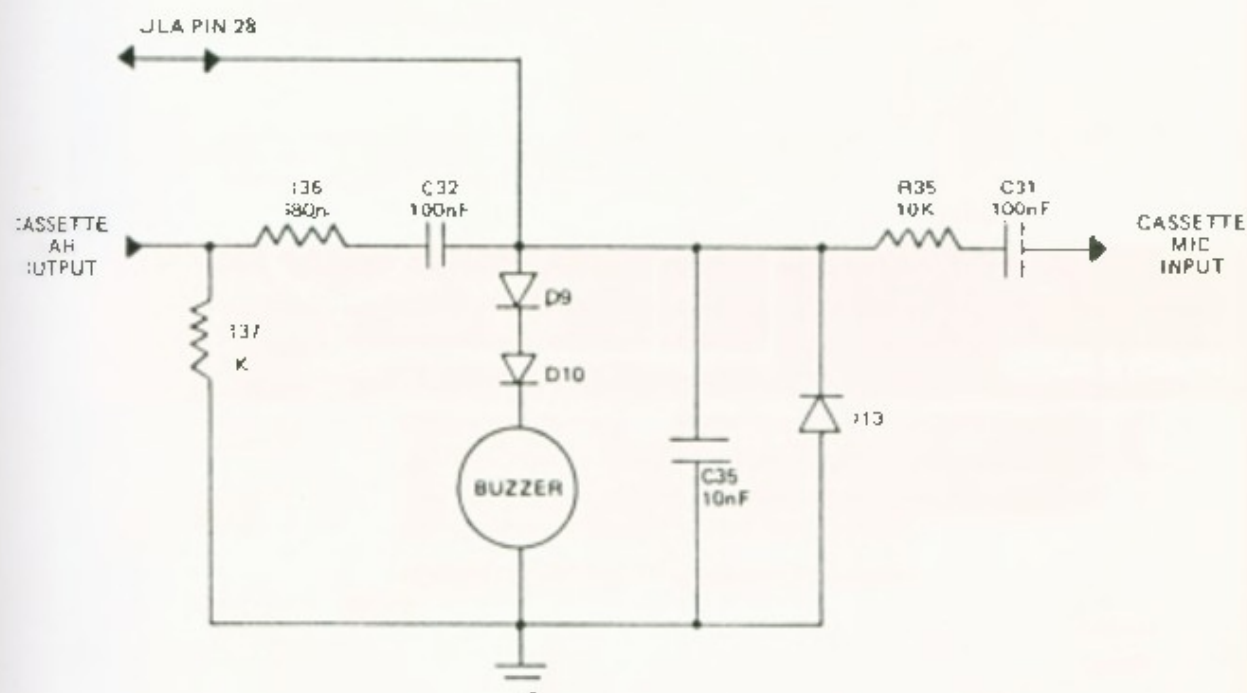


FIG. 7c — ISSUE 2 CASSETTE INTERFACE AND BUZZER CIRCUIT

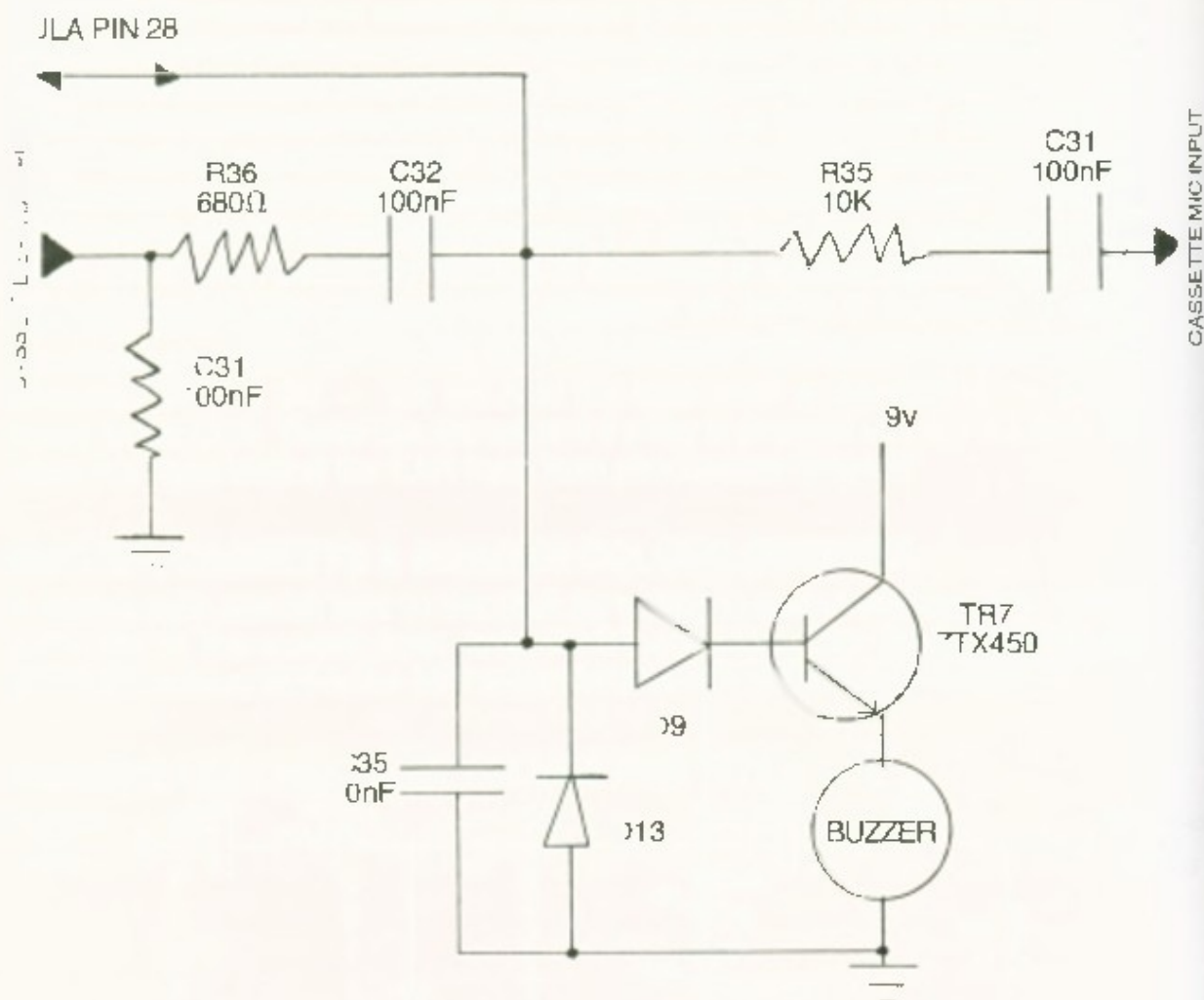


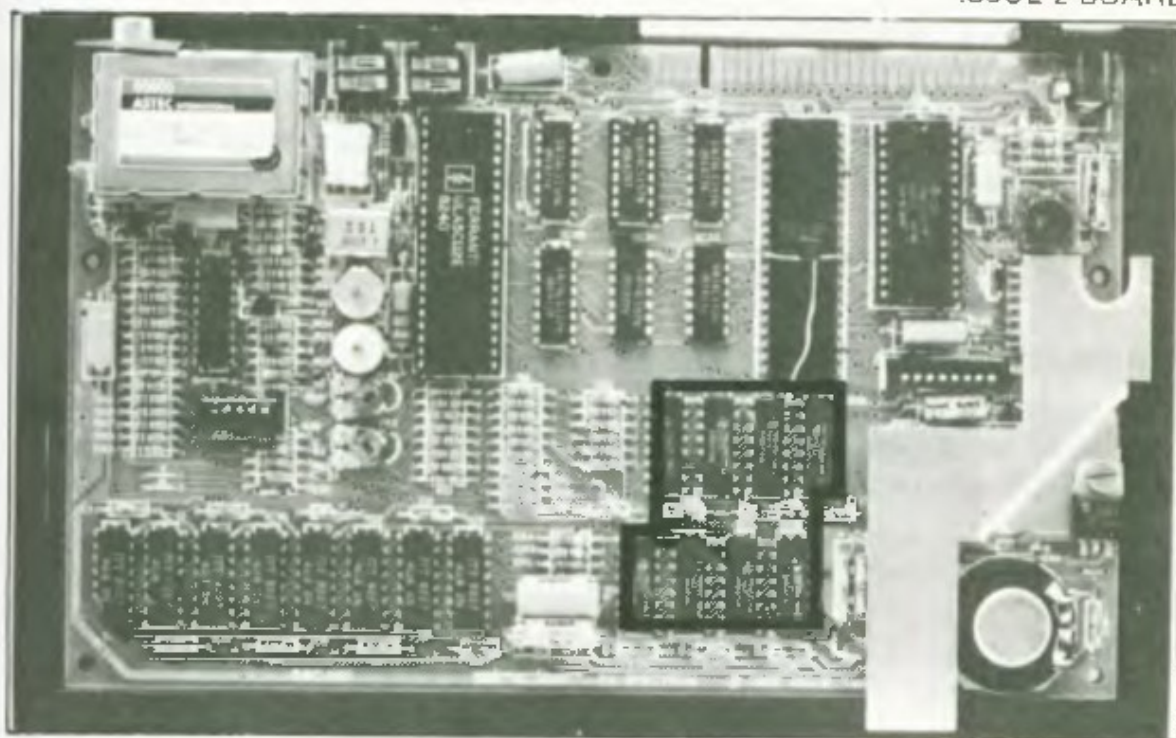
FIG. 7d — ISSUE 3 CASSETTE INTERFACE AND BUZZER CIRCUIT

## AMPLIFYING THE BUZZER

The sound output from the Spectrum's internal buzzer is normally not very loud. For certain games or other applications, it may be advantageous to connect the Spectrum up to an amplifier. This is very easy, because the Spectrum sound signal is present on both the MIC and EAR sockets at the rear of the computer. All that is required is a standard 3.5mm jack plug, a length of two way audio cable and a suitable plug to the amplifier being used. Connect the two plugs to opposite ends of the cable. For an amplifier with a high impedance input, plug the 3.5mm jack into the Spectrum's MIC socket. For an amplifier with a low impedance input, plug the 3.5mm jack into the Spectrum's EAR socket.



ISSUE 2 BOARD



SSUE 3 BOARD

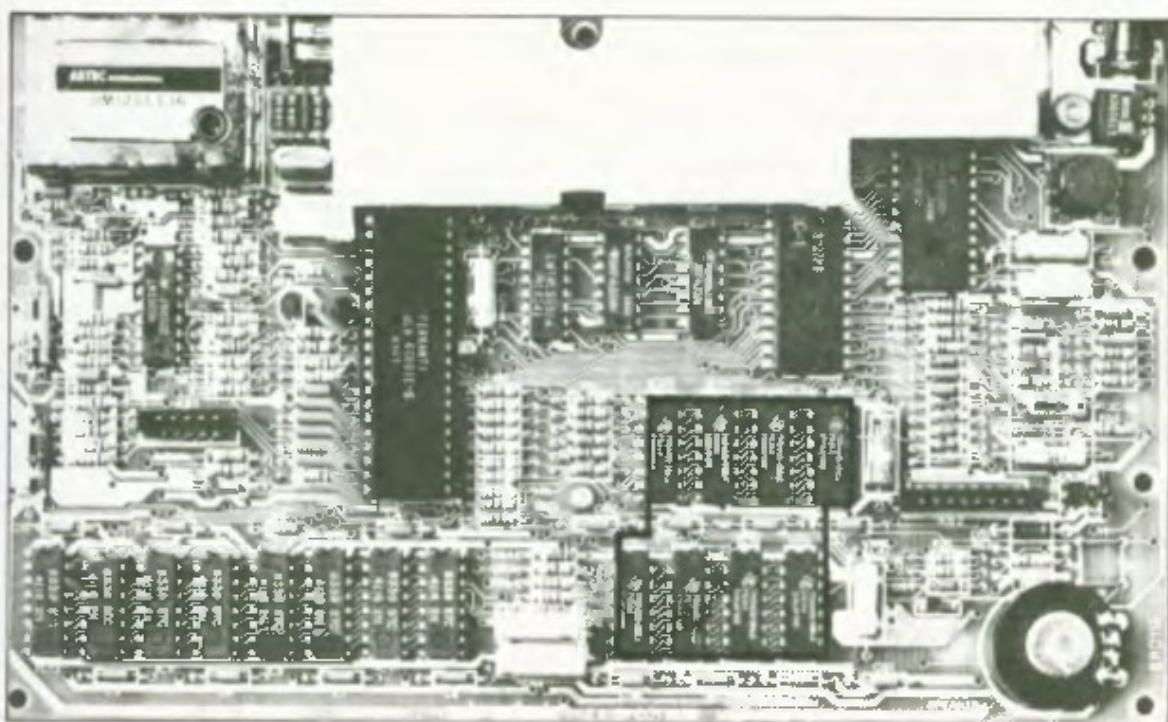


PLATE 8 — PHOTO OF MAIN SPECTRUM BOARD  
WITH MEMORY EXPANSION SOCKETS OUTLINED

## 2. THE MEMORY EXPANSION SOCKETS

The Sinclair Spectrum can be purchased in two different versions. One of these contains 16K of user RAM, and the other contains 48K. This chapter explains first of all how a 16K Spectrum can be upgraded to a 48K Spectrum, and there then follows a detailed description of how the additional memory circuit operates.

Since upgrading an Issue 3 Spectrum enables the use of chips which cannot be used on a Issue 2 Spectrum, the type of Spectrum should be checked before purchasing any chips.

### UPGRADING AN 'ISSUE 2' SPECTRUM

The upgrading procedure for Issue 2 Spectrums consists of plugging twelve integrated circuits into sockets on the printed circuit board and soldering one wire link. The following integrated circuits are required:

C15-22	8 of 4532 32K $\times$ 1 dynamic memory chips (see text)
C23	1 of 74LS32
C24	1 of 74LS00
C25-IC26	2 of 74LS157

The 4532 memory IC's will probably be the most difficult to obtain. Luckily, several mail order companies can supply these, either individually, or as a complete 'Spectrum memory upgrade kit'. A current copy of any good computing magazine should contain advertisements by some such company.

The 4532 chips used in Issue 2 Spectrums are 64K RAMs from Texas Instruments which failed to operate to specifications. Usually, several bits in one half of the chips are permanently fixed to a logic 1 or 0. The bad half of the chip is not used. Two versions are therefore available corresponding to different operational halves. These are designated the 4532L in which the lower 32K bits function correctly, and the 4532H in which the upper 32K bits function correctly.

To select the particular type of 4532 which is used, a wire link is provided on the main circuit board. Note that only Texas Instruments chips should be used. This is because Texas 4532's are split in half using the ROW address A7. Most other manufacturers' chips use A15 (COLUMN address A7) to select the operational half of the chip, so they must NOT be used in the Issue 2 Spectrum. If any difficulty is experienced in obtaining 4532's, it is possible to use any manufacturers' 64K $\times$ 1 chips instead. The 64K chips will probably be more expensive, but it will not be necessary to solder in the wire link.



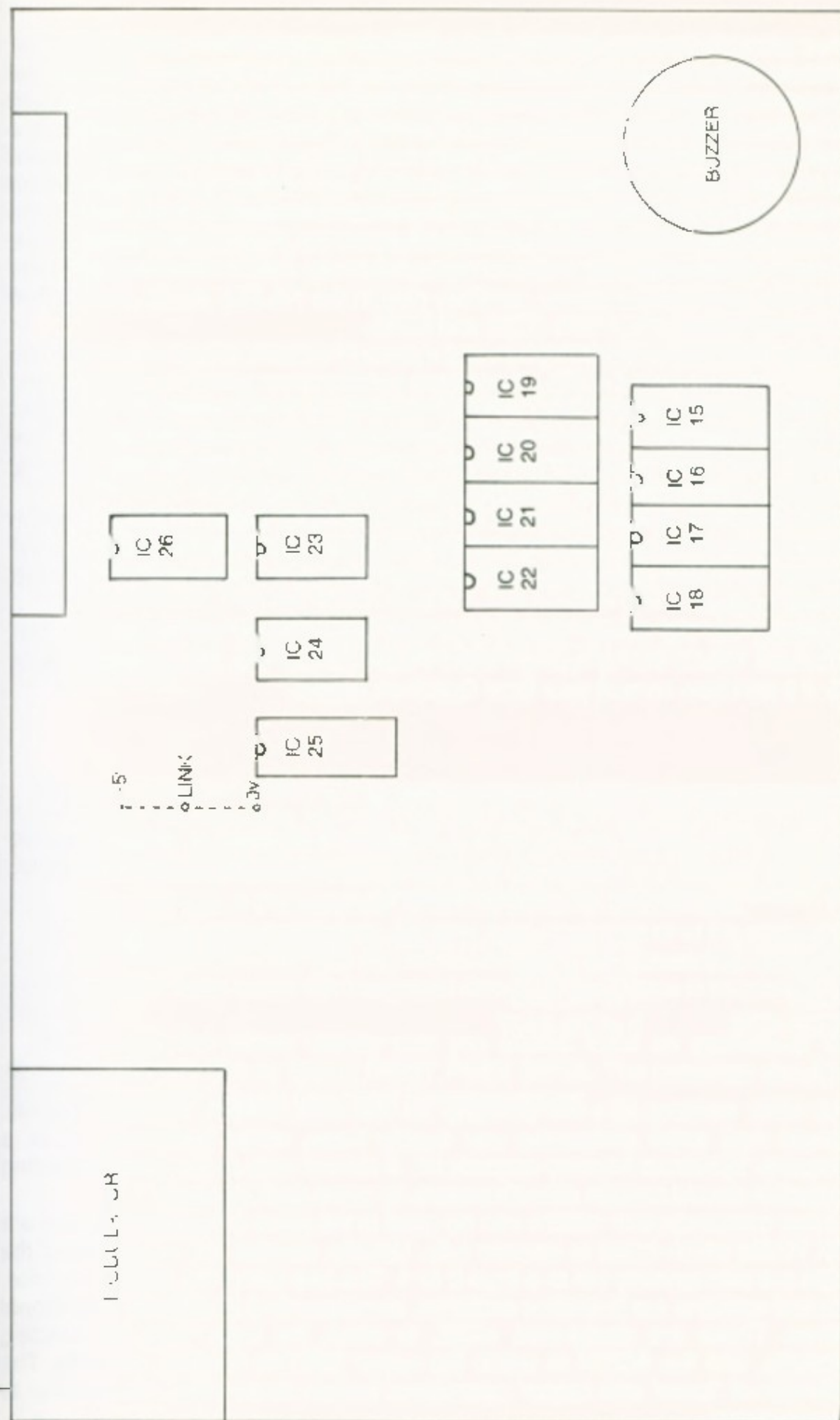


FIG. 8a — ISSUE 2 BOARD SHOWING POSITION OF MEMORY EXPANSION SOCKETS

You should now refer to the layout diagram in figure 8a. This illustrates the positions of the expansion sockets on an Issue 2 board. Carefully unscrew the five retaining screws on the underside of the Spectrum. The keyboard should then come away from the circuit board and base. Using the layout diagram as a guide, start by inserting the memory chips IC15 to IC22 into their sockets. Ensure that all of the pins on the integrated circuits are vertical and straight before attempting to insert them into sockets. The small indentations at the ends of the chips correspond to the indentations marked on the diagram. It is ESSENTIAL that ALL of the chips are inserted the correct way round. Now that all of the memories have been inserted, carefully plug in the 74LS32 (IC23), the 74LS00 (IC24) and the two 74LS157s (IC25 and IC26).

Using a fine tipped soldering iron and a piece of stripped single strand wire, the link can now be made. If 4532Ls have been installed then solder the wire between the central hole (marked "link" in figure 8a) and the hole marked 0V. If 4532Hs have been installed, then connect the wire between the central hole and the hole marked +5V. Take great care not to let the solder run onto any other connections, since that may damage the computer when power is applied.

The installation is now complete. Go back and check that all of the chips are in the correct sockets and pointing in the correct direction. Also check that none of the pins have been bent under the integrated circuits in case there is no contact with the socket.

Connect up the power supply and you should have an operational 48K Spectrum. Check this by typing PRINT PEEK 23733. The response should be 55. If it isn't then consult your local Spectrum expert.

## UPGRADING AN 'ISSUE 3' SPECTRUM

The upgrading procedure for Issue 3 Spectrums consists of plugging twelve integrated circuits into sockets on the circuit board and soldering in two wire links. The following integrated circuits are required:

IC15-IC22	8 of 4532 32K×1 dynamic memory chips (see text)
IC23	1 of 74LS32
IC24	1 of 74LS00
IC25-IC26	2 of 74LS157

The 4532 memory IC's will probably be the most difficult to obtain. Luckily, several mail order companies can supply these, either individually or as a complete 'Spectrum memory upgrade kit'. A current copy of any good computing magazine should contain advertisements by some such company.

Most manufacturers' 4532's can be used in the Issue 3 Spectrum. These are 64K RAMs which failed to operate fully. Usually, several bits in one half of the chips are permanently set to a logic 0 or 1. The bad half of the chip is not used. Several versions are therefore available, corresponding to different operational halves. These are designated 4532L in which the lower 32K bits function correctly, and the 4532H in which the upper 32K bits function correctly. The definition of 'lower' and 'upper' operational halves varies from manufacturer to manufacturer.



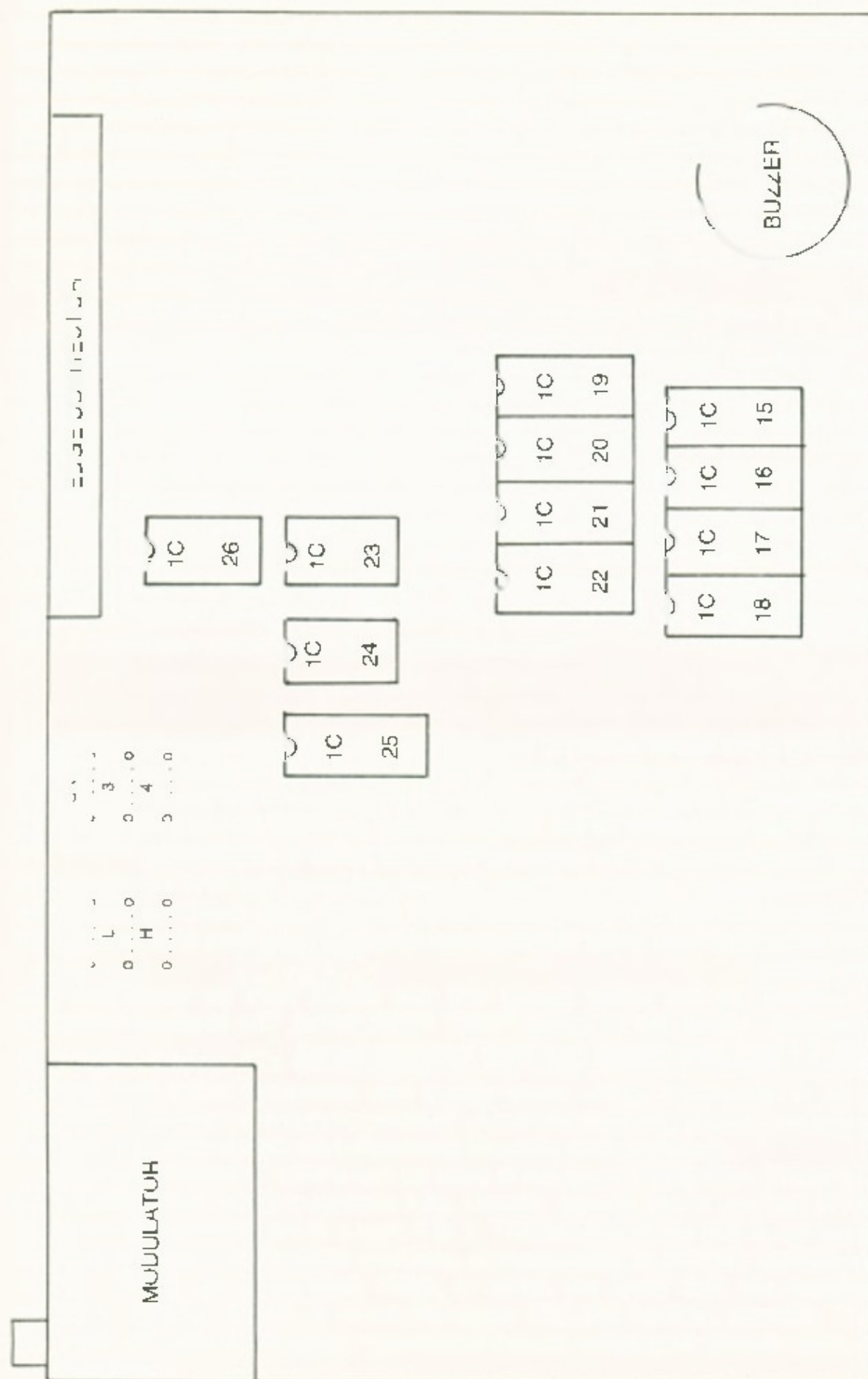
To select the particular type of 4532 which is used, a block of contact pads which can be connected by wire links is provided on the main circuit board. The connections to this link block, together with various options for common types of memory chips are illustrated in figure 8c. Texas Instruments 4532's are split in half using the ROW address A7. Most other manufacturers' chips, such as those produced by OKI use A15 (COLUMN address A7) to select the operational half of the chip. If any difficulty is experienced in obtaining 4532's, it is possible to use any manufacturers' 64K x 1 dynamic RAMs. The 64K chips will probably be more expensive, but either the 'L' or 'H' link pattern for that manufacturer's chip can be selected.

You should now refer to the layout diagram in Figure 8b. This illustrates the positions of the expansion sockets on an Issue 3 board. Carefully unscrew the five retaining screws on the underside of the Spectrum. The keyboard should then come away from the circuit board and base. Using the layout as a guide, start by inserting the memory chips IC15 to IC22 into their sockets. Ensure that all of the pins on the integrated circuits are vertical and straight before attempting to insert them into sockets. The small indentation at one end of the chip corresponds to the indentations marked on the diagram. It is **ESSENTIAL** that **ALL** chips are inserted the correct way round. Now that all of the memories have been inserted, carefully plug in the 74LS32 (IC23), the 74LS00 (IC24) and the two 74LS157's (IC25 and IC26).

Using a fine tipped soldering iron and two pieces of stripped single strand wire, the links can now be made. Various valid link patterns for different types of 4532's are illustrated in figure 8c. Choose the correct pattern for the type of chips being used. Take care not to let the solder run onto any other connections, since that may cause damage to the computer when the power is reconnected.

The installation is now complete. Go back and check that all of the chips are in the correct sockets and facing in the correct direction. Also check that none of the pins have been bent under the integrated circuits in case there is no contact with the socket.

Connect up the power supply and you should have an operational 48K Spectrum. Check this by typing PRINT PEEK 23733. The response should be '55. If it isn't, then consult your local Spectrum expert.



## Fig. 5b — 5bUE5BÚÁHU5HOWING POSITION OF MEMORY EXPANSION SOCKETS



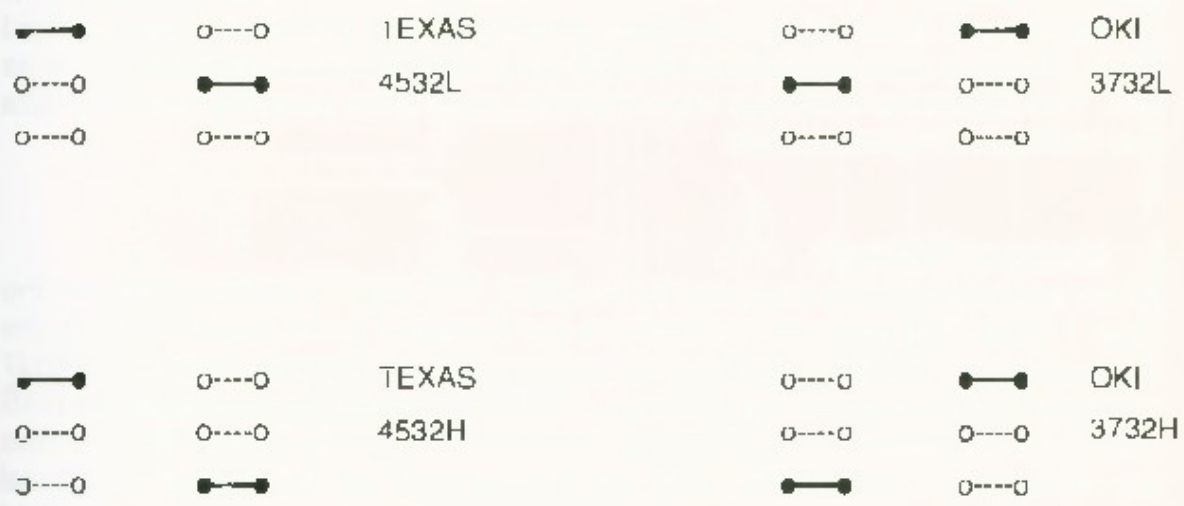
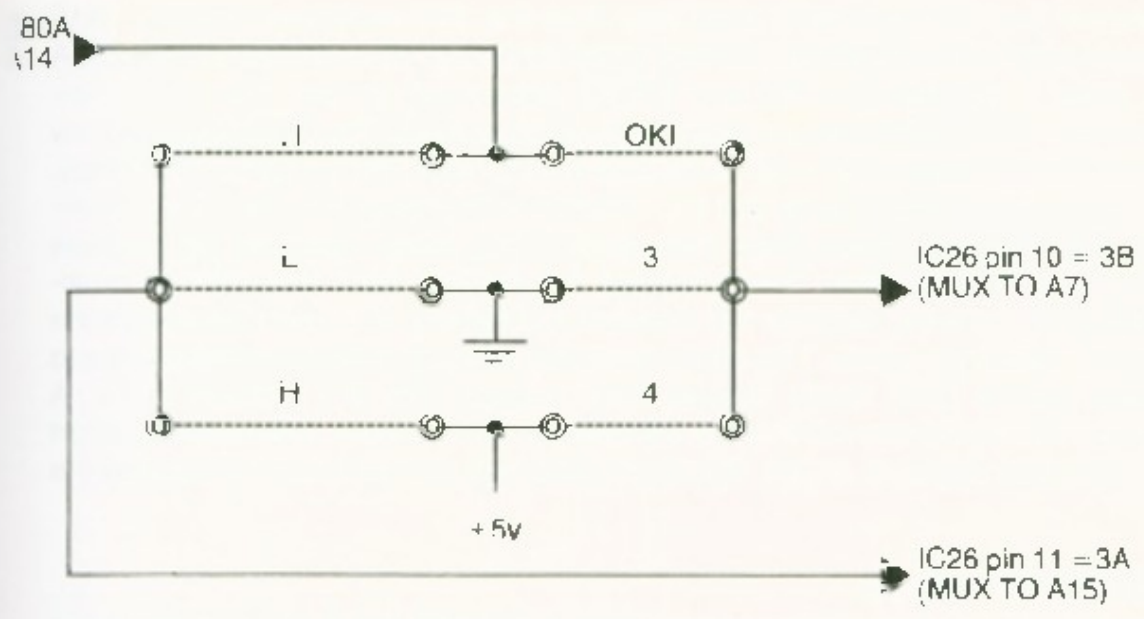


FIG. 8c — ISSUE 3 MEMORY TYPE SELECTION LINKS

# XPANSION MEMORY CIRCUIT DESCRIPTION

## 1. MEMORY READ

Whenever the CPU wishes to read data from the 32K expansion memory, A15 will go to logic 1 to indicate an address in the range 32768 — 65535. As the Z80A read cycle is initiated both RD and MREQ signals will go low. The high to low transition of MREQ is buffered by IC23a to operate RAS. The RAS line going low causes the addresses A0 — A7 to be latched into the eight dynamic RAM chips. RAS going low is delayed by the RC network comprising R71 and C63. Eventually, the RAS low signal filters through to IC23c which operates the address multiplexers "select" line. This changes the outputs from the address multiplexers to A8 — A14. Another RC network comprising R70 and C64 delays the low going signal yet again. This delay gives the address multiplexers a chance to stabilise before CAS finally latches their outputs into the memory chips. A complete 15 bit address will then be latched into the memory. After a further short delay the data appears on the Spectrum data bus so that it can be read by the CPU.

## 2. MEMORY WRITE

When the CPU wishes to store some data in this 32K block of memory, A15 is again set to logic 1. The write cycle commences with WR and MREQ going low. Addresses are latched into the dynamic memories in the same way as a read cycle. The only difference is that the WR pin on the dynamic memories is low as well. The memories therefore know that they must store data from the CPU data bus instead of outputting data.

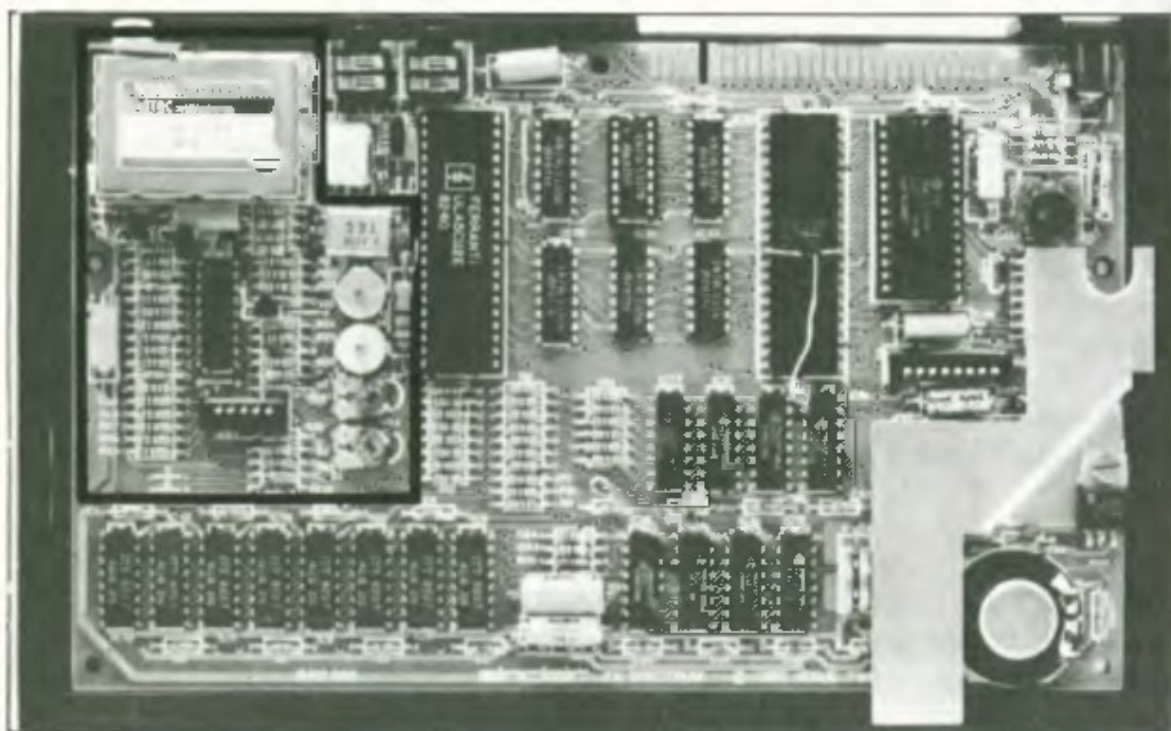
## 3. Memory Refresh

When a refresh cycle is executed by the Z80A, WR and RD remain inactive in the logic 1 state. The address multiplexers are therefore set to direct A0-A7 to the memory array. The Z80A puts the number of the row to be refreshed on A0-A6 (7 bits allow 128 different row addresses). MREQ then goes low to operate the RAS line on the memories. This is all that is required for a refresh. Each of the 128 rows are refreshed at least once every two milliseconds by this method. Continual refreshing by the CPU was not required for the video memory except during field synchronisation pulses to the television. This is because the ULA is accessing the video memory sequentially at each of the 128 row addresses during video data output.

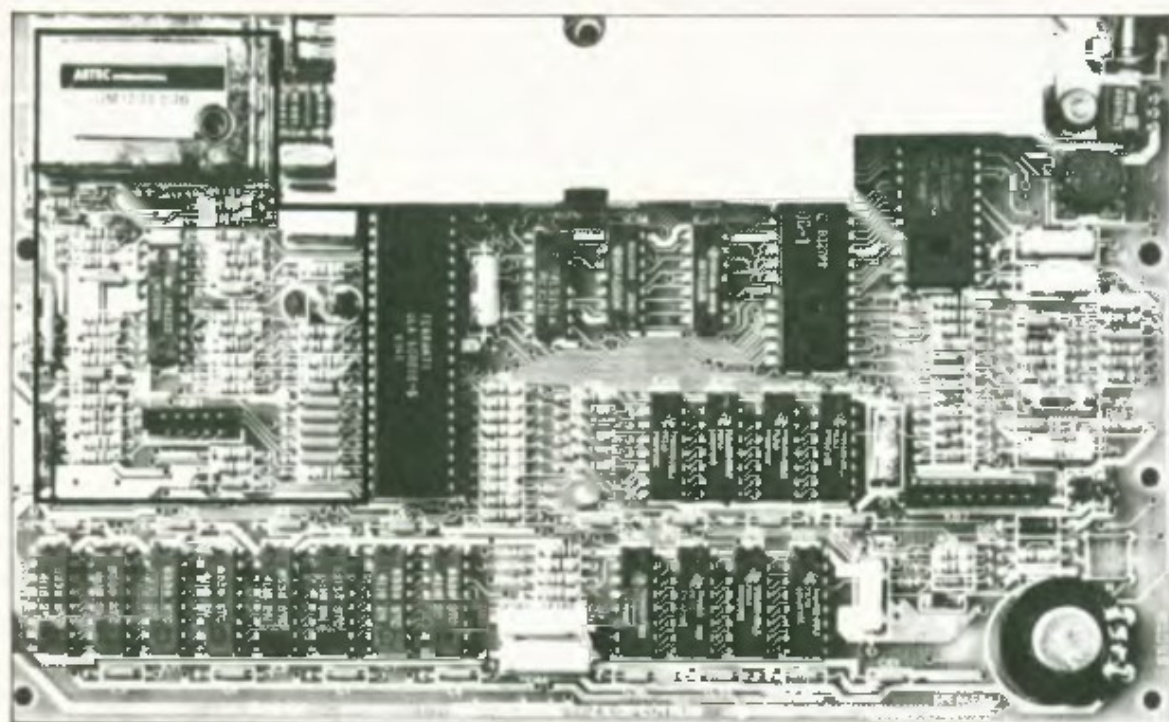




SSUE 2 BOARD



SSUE 3 BOARD



'LATE 9 --PHOTO OF MAIN SPECTRUM BOARD  
OUTLINED VIDEO CIRCUIT

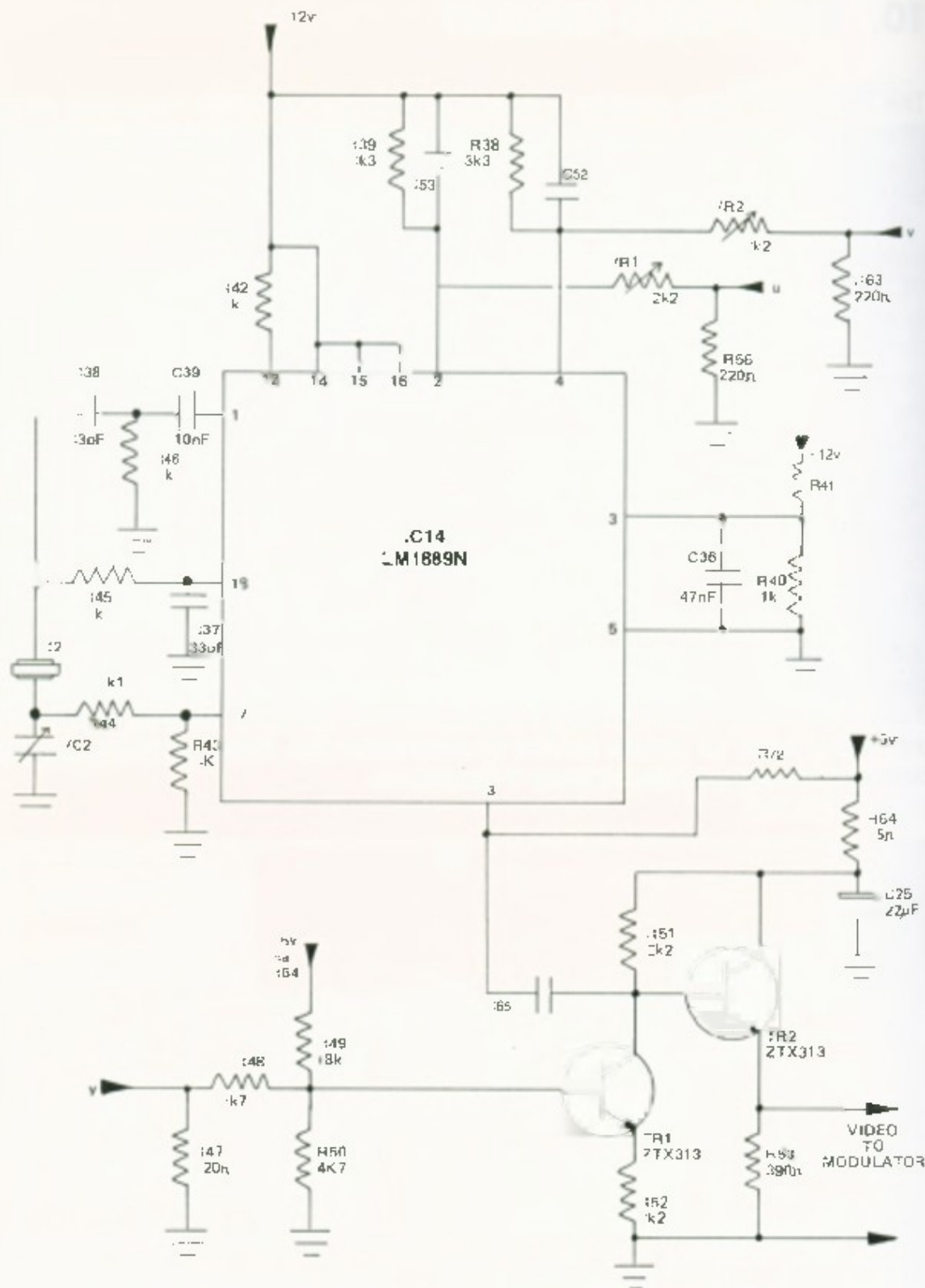


## 0. THE VIDEO CIRCUIT

The Spectrum video circuitry is based around the LM1889N integrated circuit produced by National Semiconductor. The video circuit diagrams for Issue 2 and Issue 3 Spectrums are shown in Figure 9a and Figure 9b respectively. The LM1889N accepts two colour difference signals (U = blue-yellow, and V = red-yellow) from the ULA. These are combined to produce a single colour output signal. It is more economic to use colour difference signals rather than separate RED, GREEN and BLUE signals. Only two sets of circuitry are required instead of three associated with RGB signals. The colour is mixed with the incoming composite video sync. and luminance (Y from the ULA) to produce the composite colour video signal. After being buffered by the emitter follower circuit using TR2, the video signal is fed into the video modulator. This enables the video information to be displayed on an ordinary colour television.

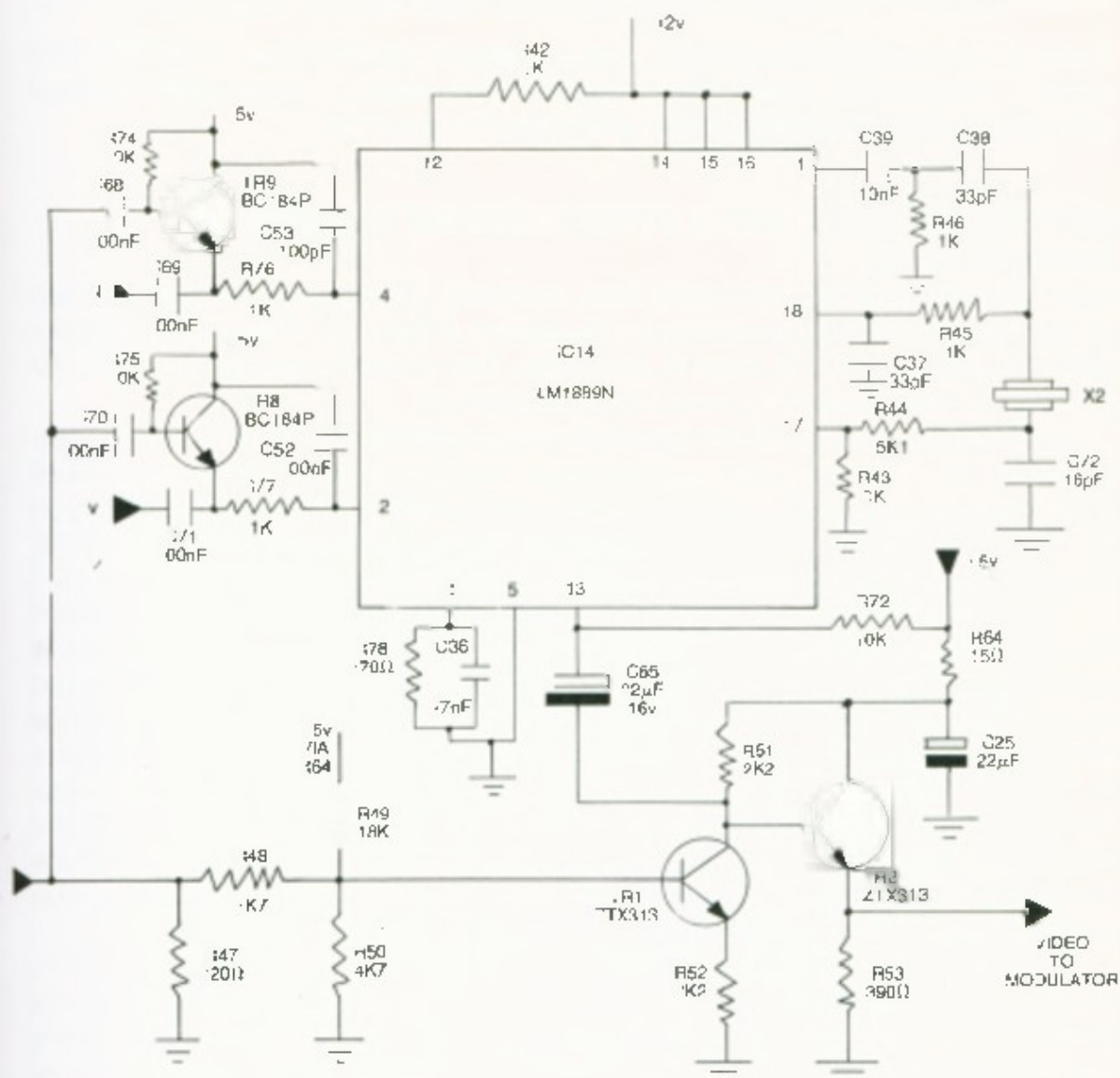
It is unfortunate that RGB video was not used, because some colour monitors require these signals. However, it is still possible to improve the rather poor quality display obtained on a television. For this, a monitor with a composite video input is required. Chapter 20 explains how such a monitor can be connected to your Spectrum.

Looking at figures 9a and 9b, it will be seen that several adjustments are available on Issue 2 Spectrums. The two adjustment resistors VR1 and VR2 alter the relative amplification of the red-yellow and the blue-yellow signals. By varying these adjustments, you can vary the output colour quality or the grey scales. See the next chapter for details on how to accomplish this. Note that NO adjustable components are provided in Issue 3 Spectrums.



IG. 9a — ISSUE 2 VIDEO CIRCUIT DIAGRAM





IG. 9b — ISSUE 3 VIDEO CIRCUIT DIAGRAM

# 1. TUNING THE ISSUE 2 VIDEO CIRCUIT FOR A BETTER QUALITY DISPLAY

It should be noted that this chapter is only relevant to Issue 2 Spectrums. The four variable components which were present inside Issue 2's have been replaced with fixed components which cannot be varied in Issue 3's.

## IMPROVING THE MOVING STRIATIONS ON THE SIDE OF CHARACTERS

Many Spectrum computers suffer from annoying striations on the side of video characters. These are caused by an interaction of the ULA 14MHz clock with the video display which it is used to generate. The striations can be improved by altering the clock frequency very slightly using VC1.

To do this, you should fill the Spectrum screen with some text. Any old program listing will do. Then turn the computer over so that it is resting on the keyboard. If you have an early version of the Spectrum, a small hole should be present on the right hand side. A small screw head should be visible inside the hole. If your Spectrum has not got a hole then it will be necessary to open the case. To do this, remove the 5 retaining screws that are visible on the underside of the Spectrum and lift off the keyboard as far as you can. Locate and remove the small retaining screw on the component side of the board near the middle. The circuit board should now be free from the bottom of the case. Turn the circuit board upside down. You should see a small hole with a screw head inside on the right hand side of the board. Using a small screwdriver carefully turn this screw a little way clockwise then back a little way anticlockwise. This should give you a feel for the effect of the adjustment. You should then be able to adjust the setting to produce the most pleasing display possible. Note that turning the screw through one complete revolution will set the adjustment back to its original value.

Sometimes it will help to readjust the channel setting on your television. Unfortunately, as the computer warms up, the crystal expands and its frequency changes. Bad striations may therefore reoccur whenever the temperature of the computer changes a lot. You should therefore let it warm up a bit before trying any adjustments. If the striations do become bad again, the only solution is to readjust the frequency as before using VC1.

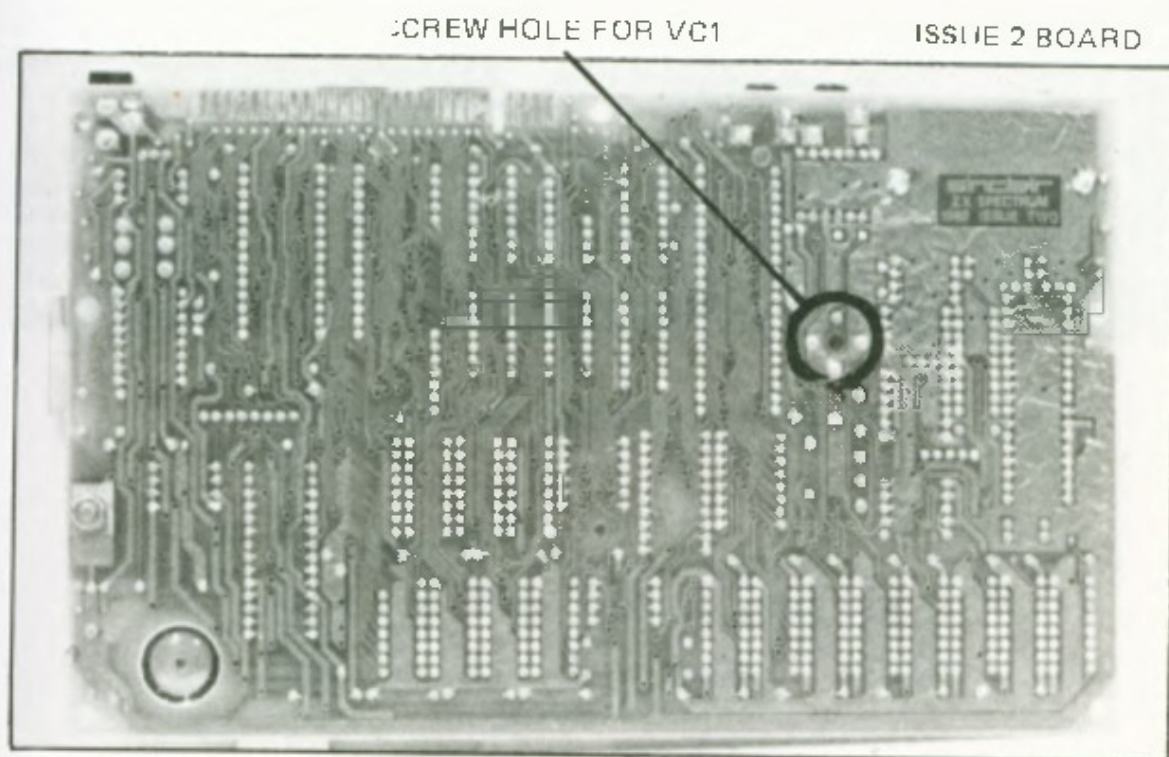
The easiest way to vary the colour is with the colour control on your television. Similarly for grey scales, adjust the brightness and contrast on your TV. The following paragraph explains how to vary the colour settings inside the Spectrum.

This ONLY applies to a colour (or grey scale) CHANGE. If there is no colour at all then refer to chapter 13.

To adjust the colour setting (or grey scale), you will need to open up your Spectrum by removing the 5 retaining screws which are accessible from the underside of your computer. Having unscrewed the case you should put a display showing all of the colours and shades onto the screen. The program at the start of Chapter 16 of your BASIC manual is suitable for this. Now you can carefully adjust VR1 and VR2 using a screwdriver. The positions of these two adjustment presets are shown in Appendix D and in Fig. 9c. Watch the colours on the screen

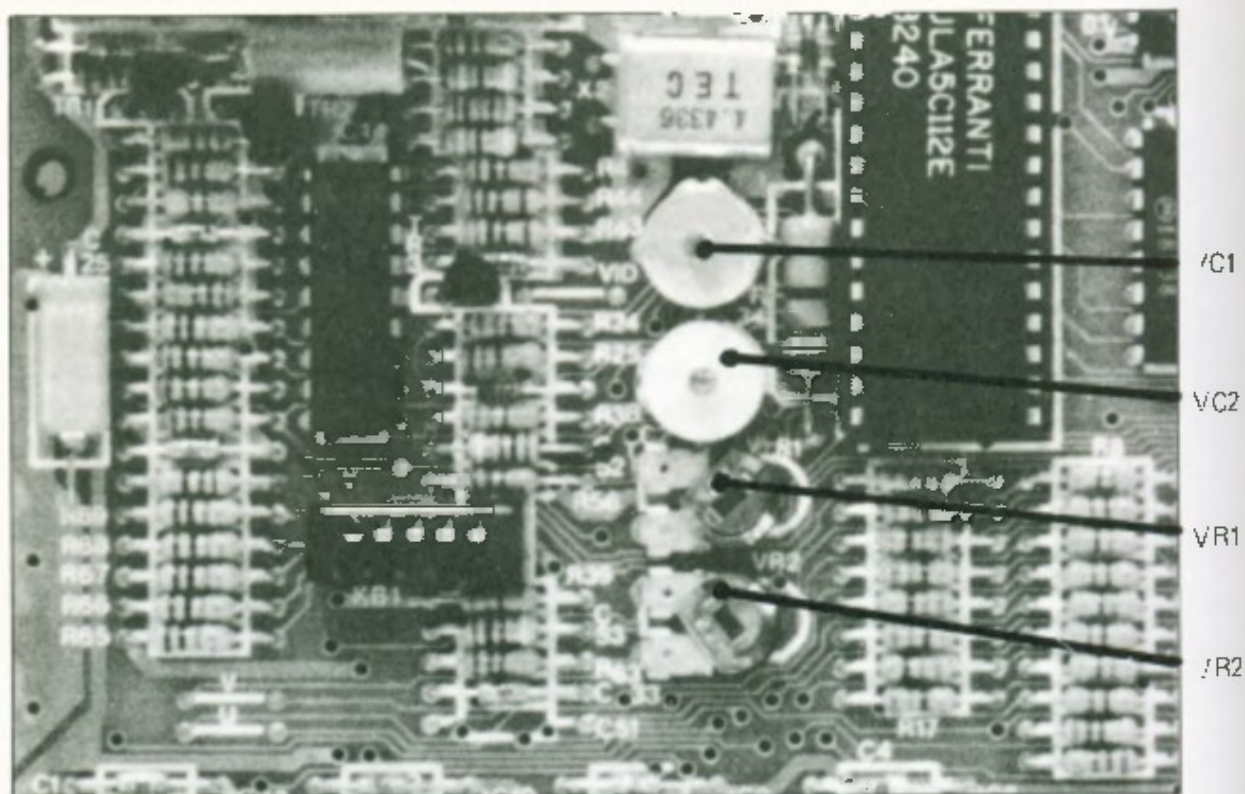


as you vary the settings. This will give you a feeling for the effect of each control. VR1 varies the red-yellow amplitude and VR2 varies the blue-yellow amplitude. VR1 has to be varied in conjunction with VR2 to vary the green level. How you eventually set the colour (or grey scale) is purely a matter of personal preference.



LATE 10 — PHOTO OF SCREW HEAD FOR VC1

## CHANGING THE COLOUR OR GREY SCALE





11	28B	28A	NO CONNECTION
9	27B	27A	A10
USACK	26B	26A	AE
COMCS	25B	25A	RFSH
4	24B	24A	WI
5	23B	23A	+12volts (UNSMOOTHED)
6	22B	22A	+12volts
7	21B	21A	WAIT
TEST	20B	20A	bvolts
SUSPQ	19B	19A	WR
1	18B	18A	RD
1	17B	17A	ORQ
2	16B	16A	MPED
VIDEO	15B	15A	KALT
ivolts	14B	14A	NMI
ORIDGE	13B	13A	INT
13	12B	12A	D4
12	11B	11A	D3
11	10B	10A	D5
10	9B	9A	D6
1LK	8B	8A	D2
1vo ts	7B	7A	D1
1vo ts	6B	6A	D0
1LOT	5B	5A	SLD1
1vo ts	4B	4A	ND CONNECTION
1vo ts	3B	3A	D7
112	2B	2A	A13
114	1B	1A	A16



FIG 10a - THE EDGE CONNECTOR

## 2. THE EDGE CONNECTOR

This chapter contains a contact by contact description of the signals available on the rear edge connector. Some hints and ideas to help you design your own circuits are included as well. For some simple experiments which you can do see chapter 14.

The connections designated 28A and 1A are marked on the upper side of the Spectrum board. 28B is under 28A and 1B is under 1A. Fig 10a illustrates the rear view of the Spectrum showing the correct orientation of the edge connector. Note that it is shown upside down in the BASIC manual!

### SIDE A CONNECTIONS

**A A15** of the address bus. This can be used as an output from the Spectrum to select various external devices in conjunction with the rest of the address bus. It can also be used as an input to the Spectrum from any external devices which take over control of the CPU buses using the BUSREQ signal.

**A A13** of the address bus

**A D7** of the data bus. This bidirectional 8 bit bus can be used to transfer information to or from the Spectrum.

**A** no connection

**A** slot to locate the edge connector in the correct position.

**A D0** data bus

**A D1** data bus

**A D2** data bus

**A D6** data bus

**0A D5** data bus

**1A D3** data bus

**2A D4** data bus

**3A** INT connected to the Z80A interrupt line, and via R26 (680R) to the ULA INT pin. This could be used by an external device to request an interrupt from the Spectrum, or if connected to +5 volts this will prevent the ULA from interrupting the Z80A CPU every 20 ms. See chapter 14 for an experiment using INT.

**4A** NMI to the Z80A non-maskable interrupt input. Normally held at logic 1 by 28(10K) connected to +5 volts. If taken low by an external device the Z80A will be forced to jump to address 102 decimal (66Hex) and start executing the machine code from there. See chapter 14 for an experiment using an NMI.

**5A** HALT from the CPU HALT output — indicates that the CPU has executed a software HALT instruction. The CPU waits for an interrupt from an external device before continuing to execute the program. See chapter 14 for a simple experiment using a HALT.

**6A** MREQ from the CPU memory request output — indicates that the address bus now holds a valid address for a memory read or write operation. This signal is also operated during a memory refresh.

**7A** IORQ connected directly to the Z80A input/output request output. Indicates that the lower half of the address bus holds a valid I/O address for an I/O operation. The upper half of the address bus holds the contents of the CPU A register if IN A,n or OUT n,A are used in a machine language program. n appears in the lower half of the address bus. If a register indirect I/O operation occurs, the



CPU C register appears on A0 — A7 and the CPU B register appears on A8 — A15. From BASIC, a full 16 bit I/O address can be specified. This appears on A0 — A15 when IORQ is active.

**8A RD** the Z80A read output indicates that the CPU wishes to read data from memory or an I/O device. The addressed device should use this signal to put the relevant data onto the data bus.

**9A WR** the Z80A write output indicates that the CPU data bus holds data to be stored at the addressed location.

**20A** — 5 volts power supply line. See the power supply circuit for details.

**21A WAIT** connected to the Z80A wait input, held high via R29 (1K5) to +5 volts — this input can be used by slow external devices to make the CPU wait until they are ready to transfer their data. Care should be taken not to operate this line for long periods of time (ie more than 1ms) because dynamic memory will not be refreshed during wait states. The program or data could be corrupted and lost if WAIT is used for too long.

**22A** +12 volt power supply output (see power supply circuit for more details).

**23A** +12 volts NOT minus 12 volts as labelled in the Spectrum manual! This pin is in fact connected to the collector of TR 4 and is therefore an unsmoothed +12 volt supply. You should not use this as a +12 volt supply directly. The +12 volt supply from 22A should normally be used. A circuit to convert it to -12 volts is given in chapter 3.

**4A M1** from the Z80A machine cycle one output — indicates that the CPU is currently getting the opcode for the next instruction to be executed from memory.

**5A RFSH** memory refresh signal from the Z80A. During a memory refresh the CPU R register appears on A0 — A7 and the CPU I register appears on A8 — A15. This can have interesting consequences if I has a value between 64 and 27. See chapter 8 for full details.

**6A** A8 address bus

**7A** A10 address bus

**8A** no connection

## SIDE B CONNECTIONS

**1B** A14 address bus

**2B** A12 address bus

**3B** +5 volts logic chip supply

**4B** -9 volts unregulated DC supply from the mains adaptor.

**5B** slot for locating the edge connector in the correct position.

**6B** 0 volts connection for the power supplies. Two connections are supplied because this line carries the sum of the other

**7B** 0 volts power supply currents. You should connect both to your external circuit to prevent either connection being overloaded.

**3B CLK** the 3.5 MHz clock signal from the ULA. Can be used to synchronise the operation of several Z80A support chips with the Z80A CPU. Note that the ULA may stop this clock for an odd cycle now and again if the CPU is accessing the first 6K of RAM. See chapter 8 for more details.

**3B** A0 address bus

**0B** A1 address bus

**1B** A2 address bus

**2B** A3 address bus



**3B IORQGE** connects to the **IORQ** input of the ULA chip. It is also connected to the **IORQ** output from the Z80A via R27 (680R). If **IORQGE** is connected to +5 volts the ULA chip will not receive its **IORQ** signal from the Z80A. This could be useful for expanding the number of devices available for user I/O. A7 could be used to disable the **IORQGE** signal when it is low. You would then be able to use A0 — A6 in any combination to address any one of 128 I/O devices. A circuit to do this together with an experiment using **IORQGE** is given in chapter 14.

**4B 0v** another zero volts connection intended for use in conjunction with the video signals.

**5B VIDEO** Video signals from the Spectrum. These are not always connected to their designated signals on Issue 2 Spectrums.

**6B Y** Chapter 20 explains how to connect up a video monitor to these signals.

**7B V**

**8B U**

**9B BUSRQ** connected to the Z80A bus request input and held high by R30 (1K) to +5 volts. This can be used by external devices to request the use of all the CPU buses. Control is handed over after the current machine cycle is completed. The Z80A signals to the external device when its buses are available by taking **BUSACK** low.

**10B RESET** connected to the Z80A reset pin. This line is provided with a simple RC delay, at power on, to allow the whole computer to reach an operational state before the CPU tries to do anything. A simple **RESET** button can be connected between this pin and 0 volts (see fig 10b). When this switch is operated the CPU will reset in the same way as at power on. The difference is that you do not have to disconnect the power supply and then reconnect it. A **RESET** switch would be extremely useful to anyone who replaces the **BASIC** ROM with a machine code monitor, because memory would not then be erased whenever a reset had to be performed.

**11B A7** address bus

**12B A6** address bus

**13B A5** address bus

**14B A4** address bus

**15B ROMCS** connects directly to the ROM chip select pin and via R33 (680R) to the ULA. If you connect **ROMCS** to -5 volts the 16K **BASIC** ROM will disappear from the Spectrum memory. Obviously you must replace this with another program in some external memory at switch over, otherwise the computer will crash!

**16B BUSACK** the Z80A bus acknowledge signal tells an external device that it now has full control of the Z80A buses. It is used in conjunction with the **BUSRQ** signal.

**17B A9** address bus

**18B A11** address bus



## 13. FAULT DIAGNOSIS

The Spectrum is a complicated device and there are several faults which could occur. Most faults tend to be of a simple nature such as a loose connection, bad tuning or a blown fuse. It is these simple type of faults which this chapter aims to isolate. For more complex repairs, the computer should be returned to Sinclair or a qualified repair company.

### SYMPTOM: No display at all

Follow these instructions in sequence:

- a. Connect the video lead from the Spectrum to the television.
- b. Switch on the TV.
- c. Connect the Spectrum power supply unit to the mains.
- d. Connect the supply lead to the Spectrum.
- e. Press the 'ENTER' key on your Spectrum and keep it pressed. Can you hear a faint clicking coming from the buzzer? If you cannot then go to 'g'.
- f. If you get here then most of your Spectrum is working. Try tuning your television to match the channel output from the Spectrum. If you still do not get a display on the TV then go to 'j'.
- g. Check that your mains power pack lead is correctly inserted into the Spectrum.
- h. Check the fuse in the mains plug to the ZX power supply.
- i. If you have access to a multimeter check that there is a  $\pm 9$  volts between the 0v and  $\pm 9$  connections on the rear edge connector. If there is not any voltage then thoroughly check all power supply connections. There is almost certainly a break in connection somewhere.
- j. If your Spectrum still isn't working then seek professional aid.

### SYMPTOM: No colour on the display

Follow these instructions in sequence:

- a. Make sure that you are actually meant to be displaying colour. The program at the start of Chapter 16 in the BASIC manual will put all available colours on the screen.
- b. Check that the colour control on your television set is adjusted to give colour. At one end of the scale you will only get black and white pictures.
- c. Check that the TV channel you are using is exactly tuned to your Spectrum. If it is not you will get a poor quality display with little or no colour.
- d. If you get as far as this, it is quite probable that the colour crystal in your Spectrum is not tuned to the colour crystal in your television. On Issue 3 Spectrums, it is not possible to tune this, so the computer will have to be repaired professionally. On Issue 2 Spectrums however, you may try to retune the Spectrum. First unscrew the five retaining screws under your Spectrum so that you can move the keyboard. The adjustment to be varied is VC2. Look at the component layout diagram in Appendix D to find it. Now rotate VC2 slowly using a small screwdriver (clockwise or anticlockwise). You should find a point at which colour returns to your display. If after a complete turn of VC2 the colour hasn't returned, you will have to get the Spectrum repaired professionally.

#### **SYMPTOM: Random graphics on the screen**

When power is connected to the Spectrum a display of random graphics in assorted colours appears. There is no copyright message present. If this occurs with no external circuits connected to the Spectrum, your Spectrum will have to be repaired professionally. The fault is normally caused by bad power supply regulation or a fault in one of the chips. The reduction in regulation of the power supplies caused by extra circuits being added can sometimes cause this problem as well. You should therefore disconnect any external circuits and try providing them with an external power supply.

Long leads to external circuits can also cause problems. There are two solutions to long leads. You can either shorten the leads, or if this is impossible, incorporate some buffer integrated circuits.

If this problem occurs even with no peripherals attached, the fault is most probably with the power supply. You can verify this by listening to the Spectrum when it is switched on. A small buzzing sound can be heard if the power supply is functioning properly.

It is advised that you return your Spectrum to Sinclair Research if this is the fault with your Spectrum. If you are feeling adventurous, however, you may take your Spectrum's life in your own hands and try replacing TR4 with a small NPN transistor. If this does not rectify the fault, then try replacing TR5 with a small PNP transistor.



## 4. EXPERIMENTING WITH THE EDGE CONNECTOR

This chapter provides lots of practical information which will be required by those who want to connect their own circuits up to the Spectrum. It should be read in conjunction with chapter 12 which explains each of the edge connector signals on a pin by pin basis. To illustrate how some of the signals can be used there are some small circuits and associated test programs.

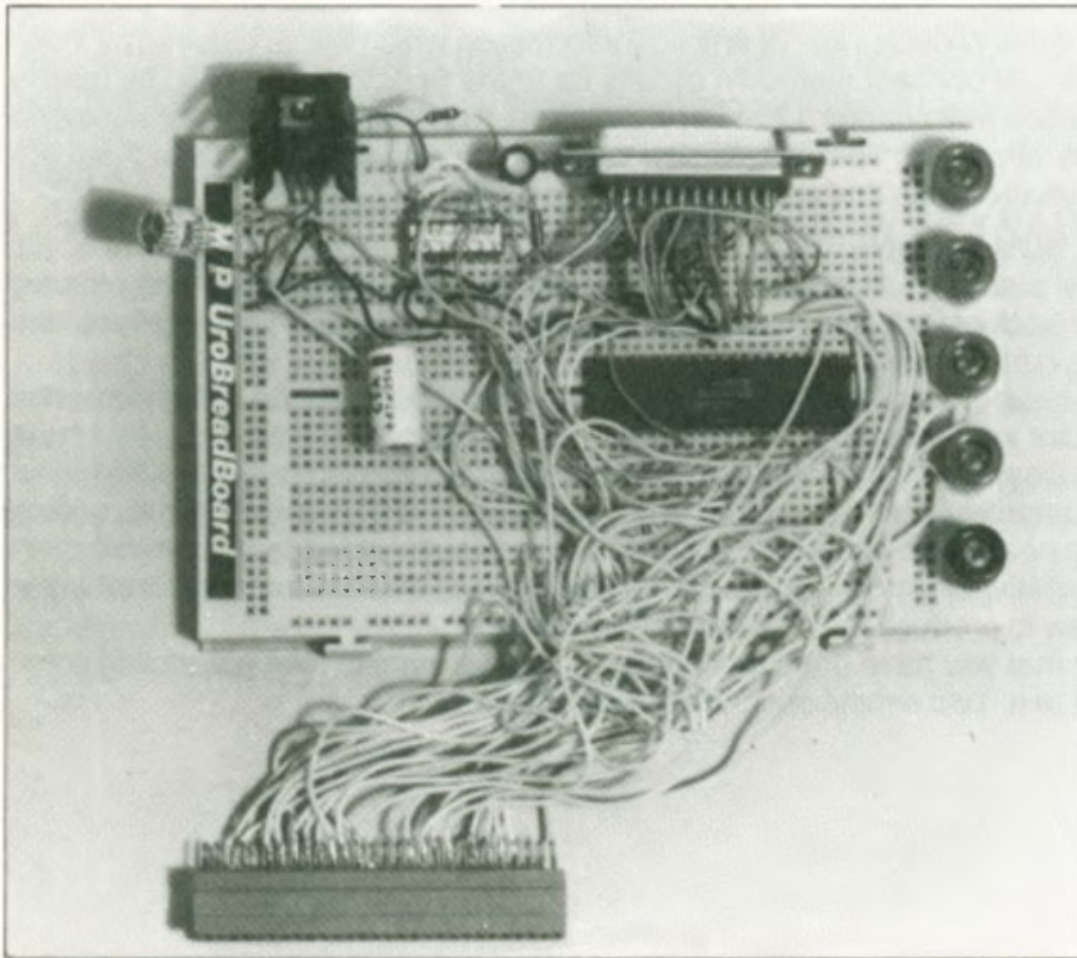
### WHAT YOU WILL NEED

The first piece of hardware which you must have is a suitable connector to plug into the back of your Spectrum. A connector will allow you to solder wires onto the various contacts very easily.

The type of connector required is a 28 way double sided 0.1" pitch edge connector with a key in position 5. The key is there to locate the connector in the correct position. You should be able to purchase a connector like this from your local computer/electronics store or by mail order. If you have any difficulty getting one it is possible to use a standard 43 way 0.1" double sided connector with a key in any position from 5 — 20. The extra contacts can then be carefully cut off with a hacksaw to produce the correct size of connector.

Now that you have got your connector you will want to start connecting some circuits to it. Two simple options are available:

## . Breadboard system



IG 10h – BREADBOARD SYSTEM

Many different types of breadboards are available. Most modern ones have metal contact strips with about 5 contacts per strip. These strips are moulded into a plastic baseboard. The leads to components such as integrated circuits, resistors, capacitors etc can be pushed through holes in the plastic to make contact with the metal strips underneath. Connections between components can then be made with lengths of single core insulated wire. These pieces of wire have about 5mm of insulation stripped from each end. The wires then push into the relevant holes on the breadboard.

This type of construction for circuits is very quick and easy. Once you have finished with a circuit it can be unplugged and a new one can be constructed. A breadboarding system is illustrated in fig 10h. This shows the prototype PIO addition and an extra +5 volt power supply. Single strand wires were soldered onto the Spectrum edge connector at one end. The other end is left free to connect to anywhere on the breadboard.



## 2. Vero strip board system

A veroboard is a ready made circuit board with horizontal copper connection strips on one side. Holes are drilled through the board spaced 0.1" apart on a square grid pattern. This type of board is useful for experimental work where a more permanent circuit than breadboarding can offer is required. Component leads are inserted through the holes and soldered onto the copper strips. So that useful circuits can be made, the copper strips will often have to be broken. These breaks remove unwanted connections between component leads. A special 'zero spot face cutter' can be used to make breaks in the tracks. A more readily available alternative is to use a 1/6" drill. To make a break, place the drill into a hole on the copper side of the board. Revolve it in the cutting direction whilst pressing it into the board. Stop when a break in the copper strip has been made.

If you use veroboard it is highly advisable to use integrated circuit sockets. This will make the chips easy to remove in the event of failure and will eliminate the possibility of them overheating when they are being soldered. The big advantage of veroboard is that you have a permanent circuit. The disadvantage is that it cannot be modified as easily as a breadboard, and used veroboards may have to be thrown away. A circuit constructed on a veroboard is shown in fig 12b.

Whichever of the above two types of prototyping board you use, you should always try to keep the connection wires as short as possible. If leads attached to edge connector contacts are too long, then the extra capacitive load may prevent the Spectrum from working at all. Long leads sometimes make the computer crash fairly frequently until it has warmed up.

Adding a few chips onto the Spectrum, such as the designs in this book, presents no problems. The Z80A can provide sufficient line driving capability itself. If you decide to design lots of additional circuits then some form of buffering will have to be added. If you are designing circuits of this complexity then it shouldn't prove too difficult to design this buffering yourself using the information given in the rest of this book. Refer to some of the books in appendix B for help.

We now come to some simple circuits which you can build. Many only consist of a simple switch and resistor.

## RESET SWITCH

The circuit is illustrated in fig 10b. You simply connect a push button between  $\overline{\text{RESET}}$  and 0v on the edge connector. When you press the button the RESET line is pulled down to 0v to reset the Z80A. Upon releasing the button, C27 charges via R31 until the voltage reaches logic 1 level. The Z80A then starts to run the machine code program starting from address 0 (usually in the Sinclair ROM). Using this button simulates a "power on reset" and consequently programs which were in the memory are deleted.

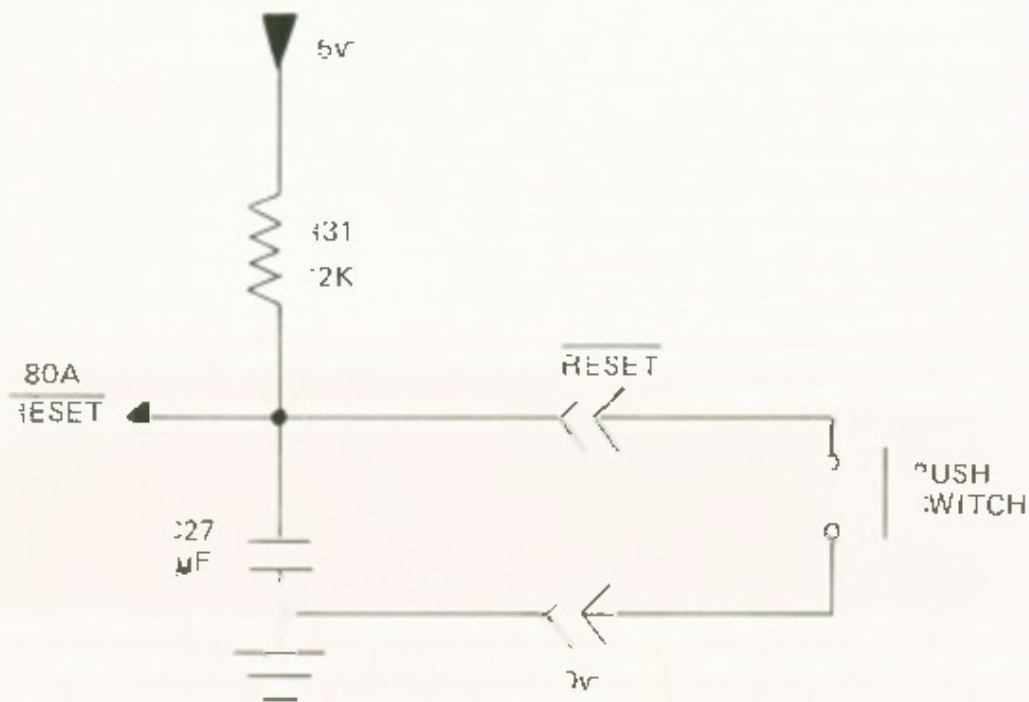


FIG 10b - RESET SWITCH ADDITION



## HALT EXPERIMENT

The  $\overline{\text{HALT}}$  line is an output from the Z80A chip. When active (output = logic 0) it indicates that the CPU has executed a software HALT instruction and is awaiting an interrupt from an external device. Whilst the CPU is halted it executes NOP's (NO operations which do nothing). These ensure that the memory refresh mechanism continues to operate irrespective of the duration of the HALT.

The circuit for this experiment is shown in fig 10c. Any small LED (light emitting diode) will do. Now try running this little program.

```
0 CLEAR 32499
10 POKE 32500,118 :REM HALT
20 POKE 32501,201 :REM RET
30 LET a =USR 32500
40 GO TO 40
```

Line 10 stops BASIC using memory above 32499. Lines 20 and 30 set up a machine code program to execute a HALT then return control to BASIC (after an interrupt has been received from the ULA). Line 40 jumps to the small machine code program. You should notice that the LED lights when you run the program. In fact it is switched on when the program first gets to line 40. Then the ULA sends an interrupt to the CPU which turns off the HALT line. The program goes back to line 40 and turns the HALT on again, until the next interrupt is received from the ULA. You can't actually see the LED flashing on and off because it is flashing at 50 Hz (the ULA interrupts the CPU 50 times every second).

## NMI TEST

The circuit for this test is shown in fig 10d. It simply requires a push switch between the  $\overline{\text{NMI}}$  input on the edge connector and the 0 volt line. If you press the button to generate an  $\overline{\text{NMI}}$  (non maskable interrupt) then, no matter what the CPU was doing previously, it will now start to run the interrupt service routine at address 66Hex (102 decimal). This in fact will reinitialise BASIC and so it has the same effect as pressing the RESET button.

## INT AND IORQGE TESTS

In these tests we are going to use the circuits in fig 10e and fig 10f. These effectively connect the appropriate lines on the edge connector directly to +5v. The 100 ohm resistor is there to reduce the current taken from the power supply and can be omitted.

Now enter the following short program. Leave both switches for  $\overline{\text{INT}}$  and  $\overline{\text{IORQGE}}$  open whilst doing this.

```
0 CLS
10 PRINT AT 0,0; PEEK 23672 - 256*PEEK 23673
20 GO TO 20
```

When you run the program you will see an incrementing number at the top of your screen. This shows how many times the CPU has been interrupted since it was switched on. Now operate the  $\overline{\text{INT}}$  switch. The clock stops. Try to BREAK your BASIC program. Nothing happens! All is not lost however. Switch the  $\overline{\text{INT}}$  switch off again and the counter restarts where it left off. You can also use the keyboard again.

Why did this happen? Well, the way in which the counter and keyboard operate relies upon interrupts. Once every 50th of a second the ULA activates the  $\overline{\text{INT}}$  line. This causes the CPU to jump to a little bit of machine code in the BASIC

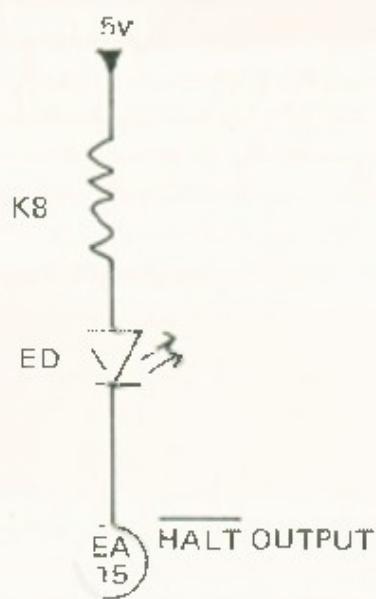


FIG 10c - HALT TEST



FIG 10d - NMI TEST

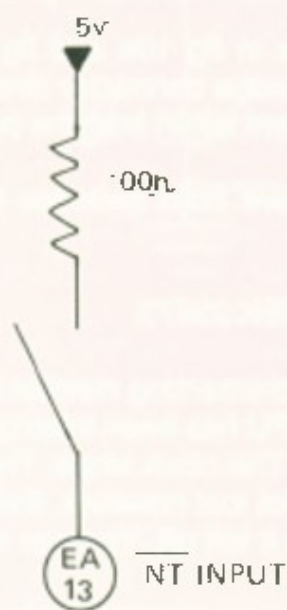


FIG 10e - INT TEST

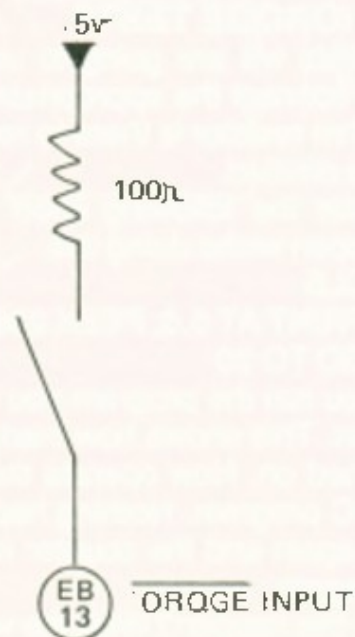


FIG 10f - IORQ TEST



IOM. The purpose of this routine is to scan the keyboard to see if a key has been pressed and to increment a 3 byte counter in memory. By connecting INT to +5 volts, you stop the ULA interrupt signal reaching the CPU. The CPU doesn't ever scan the keyboard or increment its counter unless it receives an interrupt. Therefore none of the keys on the keyboard operate and the counter stops.

Now enter this one line program.

```
0  FOR t = 1 TO 30: BEEP 0.01,t: NEXT t: GO TO 10
```

If you run it you will hear a BEEP of rising pitch from the buzzer. Try operating the INT switch. It has no effect and the tone continues. INT doesn't affect output from the CPU to the ULA. It only affects inputs from the ULA to the CPU. Switch INT off again. Now try operating the IORQGE switch. The BEEP suddenly stops. Yet again the keyboard has become totally unuseable, but for a totally different reason. Can you see why?

All I/O which is done by the ULA relies upon the ULA being able to detect whenever the CPU wants to send it information or get information from it. The ULA uses the address lines, RD, WR and IORQGE to test if the CPU is trying to communicate. By connecting IORQGE to +5 volts, you are preventing the Z80A IORQ signal from reaching the ULA. No communication between the two devices can therefore occur. The 3 byte ON time clock does continue to run because the ULA is still generating interrupts for the CPU 50 times per second.

## 5. ADDING A Z80A PIO CHIP

This parallel input/output chip enables the Z80A CPU to communicate with the outside world. In this chapter a brief explanation of how the PIO works, how to program it, and how to connect it up to the rear edge connector will be given. It is not intended to give a complete and fully detailed description of the PIO chip. This information can be readily obtained from specialised books on Z80 interfacing. Chapter 18 contains details of an 8 channel analogue voltage to digital output converter as a practical example of a useful device which can be run from the PIO. One of its many uses is for reading the X and Y coordinates from a joystick input.

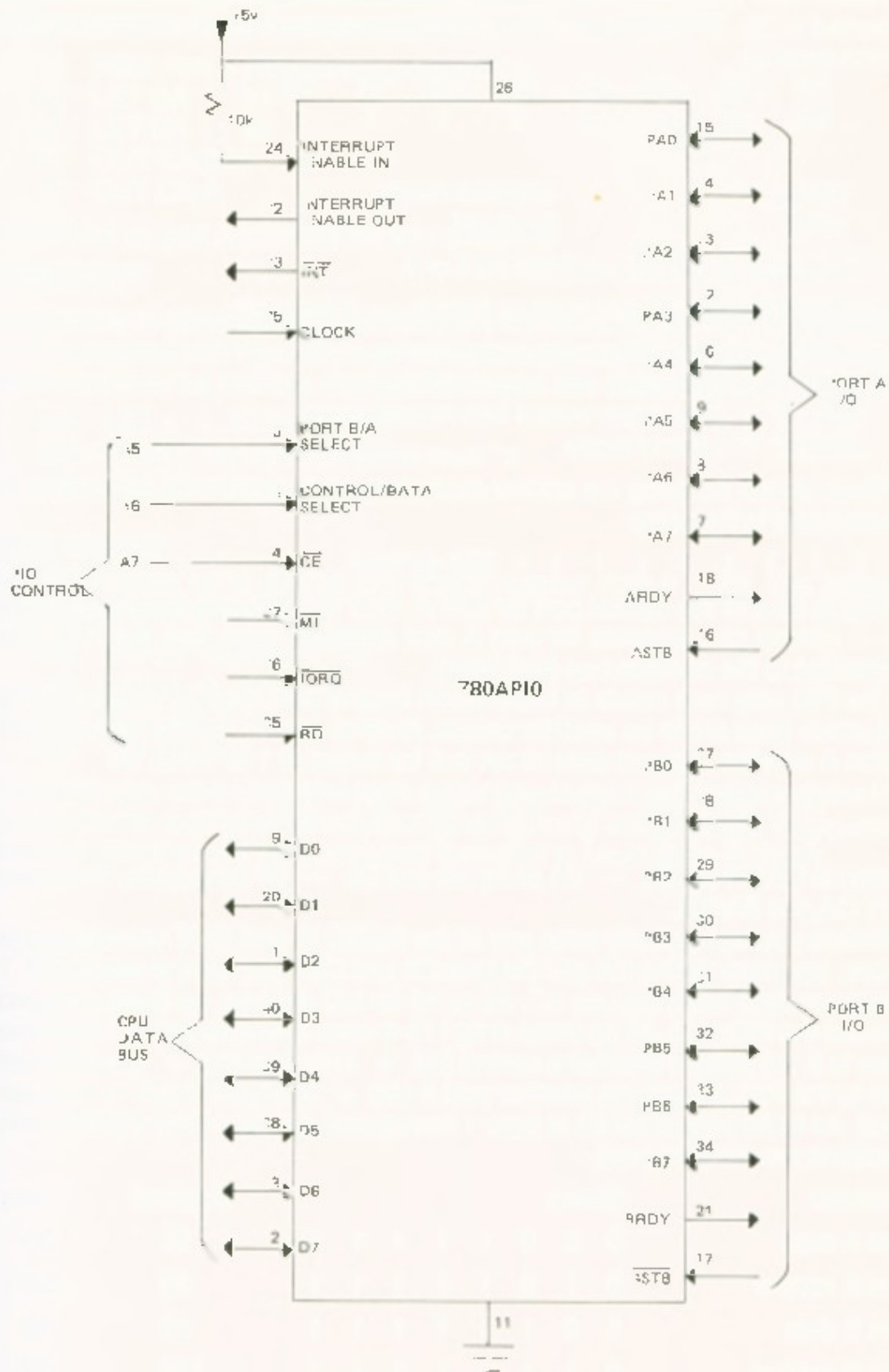
The Z80A PIO chip has been designed specifically for use with the Z80A CPU chip. If you look at the pin connections in fig. 11a you will notice lots of familiar signals. There is the databus for fully bidirectional communication with the CPU, the clock,  $M1$ ,  $\overline{IORQ}$ ,  $\overline{RD}$  and  $\overline{INT}$  signals all being connected directly to their equivalent pins on the edge connector. There are also two pins for the "interrupt enable input" and "interrupt enable output" signals. These are only of relevance when interrupts are being used. The basic use of interrupts are to force the CPU to run a specified piece of machine code program. For example, the Spectrum might be running BASIC and also be under use as a central heating controller. The CPU doesn't want to waste time keeping a check on the house temperature (from a signal from a thermostat) because this would slow down the BASIC. We would therefore use interrupts. If the temperature gets too high or too low anywhere in the house the thermostat circuit would interrupt the CPU which is then forced to run the central heating program. When this is completed it can go back to BASIC and forget about the heating until it is interrupted again.

Looking at the other connections you will see that there are two I/O ports. These are designated port A and port B to distinguish between them. Each port has 8 I/O lines ( $PA0 - PA7$  and  $PB0 - PB7$ ) and two handshake lines ( $ARDY$ ,  $ASTB$  and  $BRDY$ ,  $BSTB$ ). In this book only the eight I/O lines will be used. The other signals are used in BYTE data transfer modes between input/output devices (usually of different computers).

In order to illustrate the PIO chip programming there follows a practical example with explanation. The object is to define  $PA0$  to  $PA1$  as inputs and  $PA2$  to  $PA7$  as outputs. It will then be seen how these can be used by the CPU to operate the switches and lamps circuit. This example uses port A, however port B can be used in exactly the same way by sending control words and data words to port B instead. The circuit shown in fig 11b should be built on the same board as the PIO chip. The 74LS05 integrated circuit is used to operate the light emitting diodes (LED's) because the PIO chip outputs cannot supply enough current. Virtually any type of LED can be used in this circuit.

If you have a 16K Spectrum then the Spectrum's own +5 volt power supply should be sufficient to operate all of the circuits in this book. 48K Spectrum owners may find that their -5 volt power supply just cannot provide sufficient current for their extra 32K of memory plus the additional circuits given here. If you find this to be the case with your Spectrum, an additional +5 volt power supply will be required. A suitable design is discussed in chapter 3.





IG 11a - Z80A PARALLEL INPUT/OUTPUT CHIP

Before the PIO chip can begin to operate as we desire, it must be told what it is expected to do. This is carried out by writing a control word to it. The format of a control word is:

17	D6	D5	D4	D3	D2	D1	D0
M1	M0	x	x	1	1	1	1

mode word                      signifies mode to be set.

not used by the PIO so it can be 0 or 1.

The mode word can be any number from 0 — 3 inclusive.

mode 0 = byte output with handshaking  
 mode 1 = byte input with handshaking  
 mode 2 = byte input/output with handshaking  
 mode 3 = control mode

Since we will be using the control mode, the mode word must be set to 3. The control word is therefore:

1 1 1 1 1 1 1 = 255 decimal

This is sent to the control port for port A. OUT CA,255 can be used in BASIC. CA = control port address for port A and depends upon how the PIO chip is connected to the address bus (see later).

The PIO chip now needs to know which lines of port A are to be used for input and which lines are to be used for output. Another control word is therefore written to port A with the relevant data bit set to 0 for output or 1 for input. So

17	D6	D5	D4	D3	D2	D1	D0
1	0	0	0	0	0	1	1

Written to CA will set D0 — D1 as inputs and D2 — D7 as outputs. You can use OUT CA,3 in BASIC.

Before you can actually write any of the BASIC IN and OUT commands given above, you must define CA and DA plus CB and DB if you want to use port B. The Spectrum itself does not use A7, A6 or A5 for I/O devices, so these are used to operate the PIO chip. A7 as wired in fig 11a is used to enable the PIO chip when it is set to 0. A6 selects whether a control (A6 = 1) or data (A6 = 0) word is being sent to the PIO. A5 selects whether port A (A5 = 0) or port B (A5 = 1) is being used. A4 to A0 should be left at logic 1 throughout, so that ordinary Spectrum I/O functions are not disrupted.

Selected	Binary	Decimal	Token
Port A data	0 0 0 1 1 1 1 1	31	DA
Port A control	0 1 0 1 1 1 1 1	93	CA
Port B data	0 0 1 1 1 1 1 1	63	DB
Port B control	0 1 1 1 1 1 1 1	127	CB



## EXPERIMENTS WITH THE LED's and SWITCHES

Always start your BASIC programs with:

```
0 REM Port A initialisation program
10 LET DA = 31
20 LET CA = 93
30 OUT CA,255
40 OUT CA,3
```

This will set up port A correctly for your program.

### SOME INPUT

It is possible to generate each of the binary numbers 00, 01, 10, 11 equivalent to 0—3 in decimal with the switches in different positions. Try the following program loop to see how the number displayed changes with different switch positions.

```
00 LET x = IN DA
20 PRINT "My input switches read ";x
30 GO TO 100
```

### SOME output

The LED's allow us to display any binary number from 0—63. The following program will do this for you:

```
00 INPUT "Number? ";A
10 LET A = A * 4
20 OUT DA,A
130 GO TO 100
```

Now that you have seen how to do some simple input and output with the PIO, you could write a program which counts up in binary automatically. By operating one or the switches the program could then start to count down in binary.

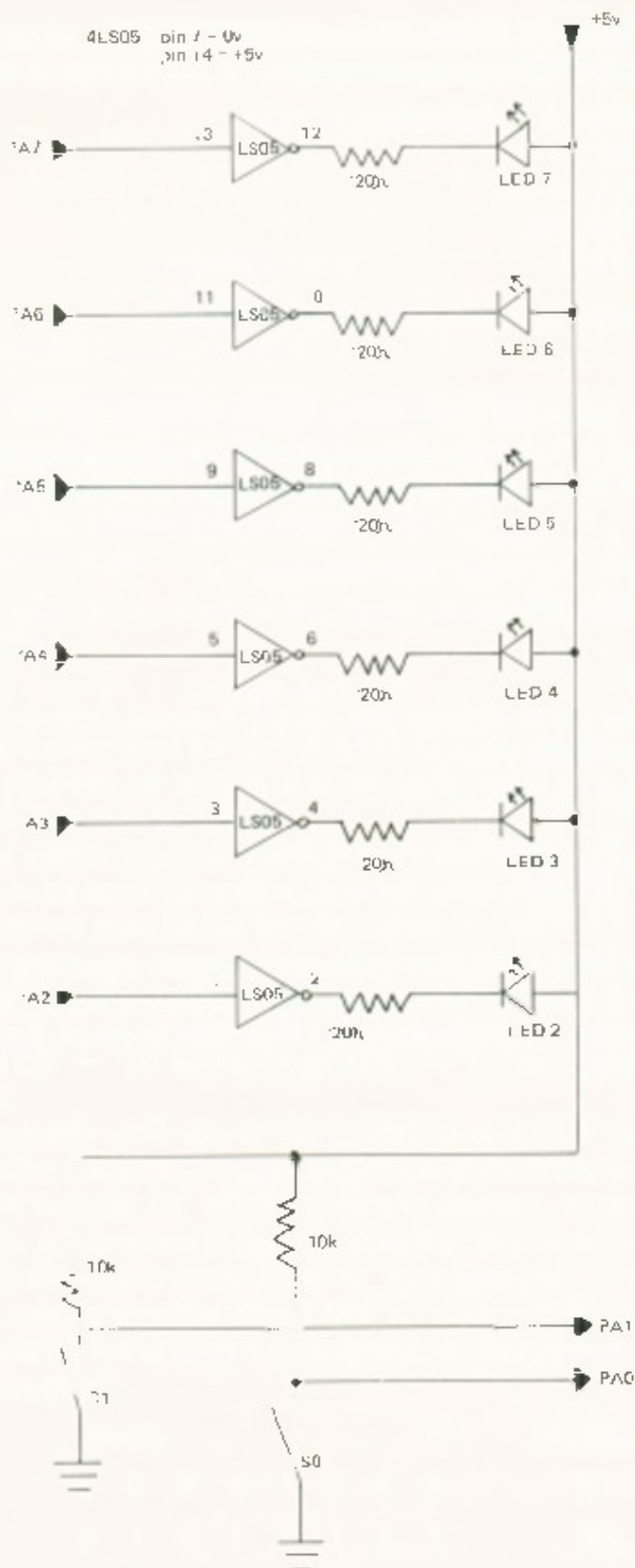


FIG 11b - LAMPS AND SWITCHES EXPERIMENT





## 16. ADDING YOUR OWN KEYBOARD

### INTRODUCTION

This chapter explains in detail how you can build a keyboard interface which will plug directly into the rear edge connector. It will operate in parallel with the original Spectrum keyboard so that two keyboards can be used simultaneously. Chapter 17 explains how you can use this keyboard interface for Sinclair compatible joysticks. The possibilities are enormous. You could add a numeric keypad (an old calculator keypad for example), a hexadecimal keypad for machine code programming, or a complete full size keyboard.

### CIRCUIT DESCRIPTION

You should refer to the circuit diagram illustrated in fig 12a. The keyboard connections EKBD1 — EKBD13 shown on this circuit diagram are equivalent connections to KBD1 — KBD13 shown in fig 6. The signals IORQGE, A0, and RD are all combined so that the common inputs to NOR gates IC1b, and C2a — IC2d are only low when these signals are all low. This extra keyboard interface is therefore selected whenever the ULA (port 254) is addressed to be read from. If none of the keys are pressed then the inputs to the NOR gates are pulled high by resistors R2 — R6. Outputs to the NOR gates are low and are inverted by IC3 to give a high on the data bus lines D0 — D4. Since IC3 hex buffers have 'open collector' outputs, the ULA can easily set any of D0 — D4 at logic 0 if any of the Spectrum keyboard keys are pressed. Now imagine that KBD10 is connected to EKBD6. If you look at fig 6 you will see that this indicates that the 'F' key has been operated. When a read occurs with A9 low the input to IC2d is pulled low. The output to IC2d goes high and D3 is pulled low by IC3d. The CPU can then read D0 — D4 directly from its data bus and register that the 'F' key has been pressed.

### PARTS LIST

#### Resistors

- R1 4K7
- R2 — R6 10K

all 5% ¼ watt

#### Capacitors

- C1 22uF 6 volt electrolytic
- C2 0.1 uF disc ceramic

#### Semiconductors

- IC1 74LS02
- IC2 74LS02
- IC3 74LS05

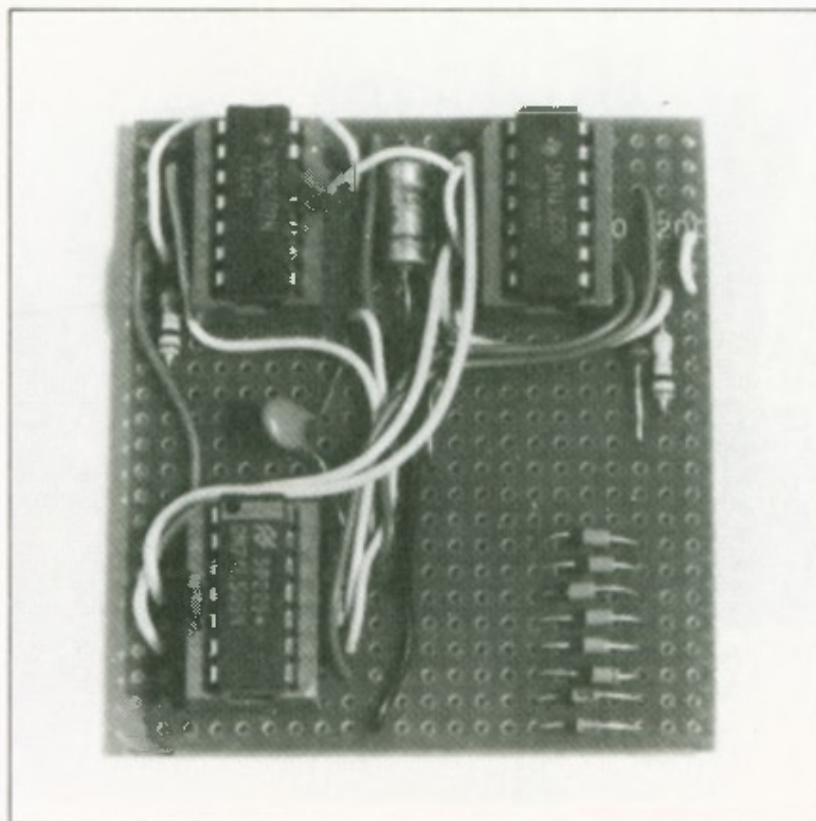
- D1 — D8 IN4148 diodes

#### Miscellaneous

- 1 off 14 pin DIL IC sockets
- 18 way Spectrum edge connector
- 1.3" x 2.6" (or larger) piece of 0.1" pitch copper strip veroboard
- wire and solder



## CONSTRUCTION DETAILS



**FIG 12b — KEYBOARD INTERFACE BOARD**

This circuit can be built on breadboard or veroboard. The component arrangements for the veroboard design are illustrated in fig 12b. Sockets should be used for the three integrated circuits so that they are not damaged by overheating during soldering. Make sure that you break all of the copper strips as shown in the diagram. A 1/6" drill is suitable for this (see chapter 14 for some hints on construction). The circuit must be carefully wired up as in the full circuit diagram in fig 12a. The completed circuit board showing all of the components soldered into position, together with the interconnection wires is shown in fig 12b. The tips of the arrows on diodes D1 — D8 in the circuit diagram are normally represented by a band on the correct end of the components.

You will now want to connect some form of keyboard up to the circuit. The next chapter explains in detail how to connect up some joystick game controls. Alternatively, there are lots of full size keyboards available which could be connected up. The major problem here is that all of the keyboards tend to be wired in a different way. It is not therefore possible to give exact details for converting different keyboards in a book of this nature. The basic requirement is that the keyboard is rewired to conform to the Sinclair keyboard shown in fig 6. For example, the 'I' key must be connected between EKBD4 and EKBD11. The '7' key must be connected between EKBD5 and EKBD10 and so on. If this rewiring of a keyboard does not appeal to you, several large mail order companies will supply individual keyboard switches. It is fairly easy to mount these on a large piece of veroboard on a 5 × 8 key grid.

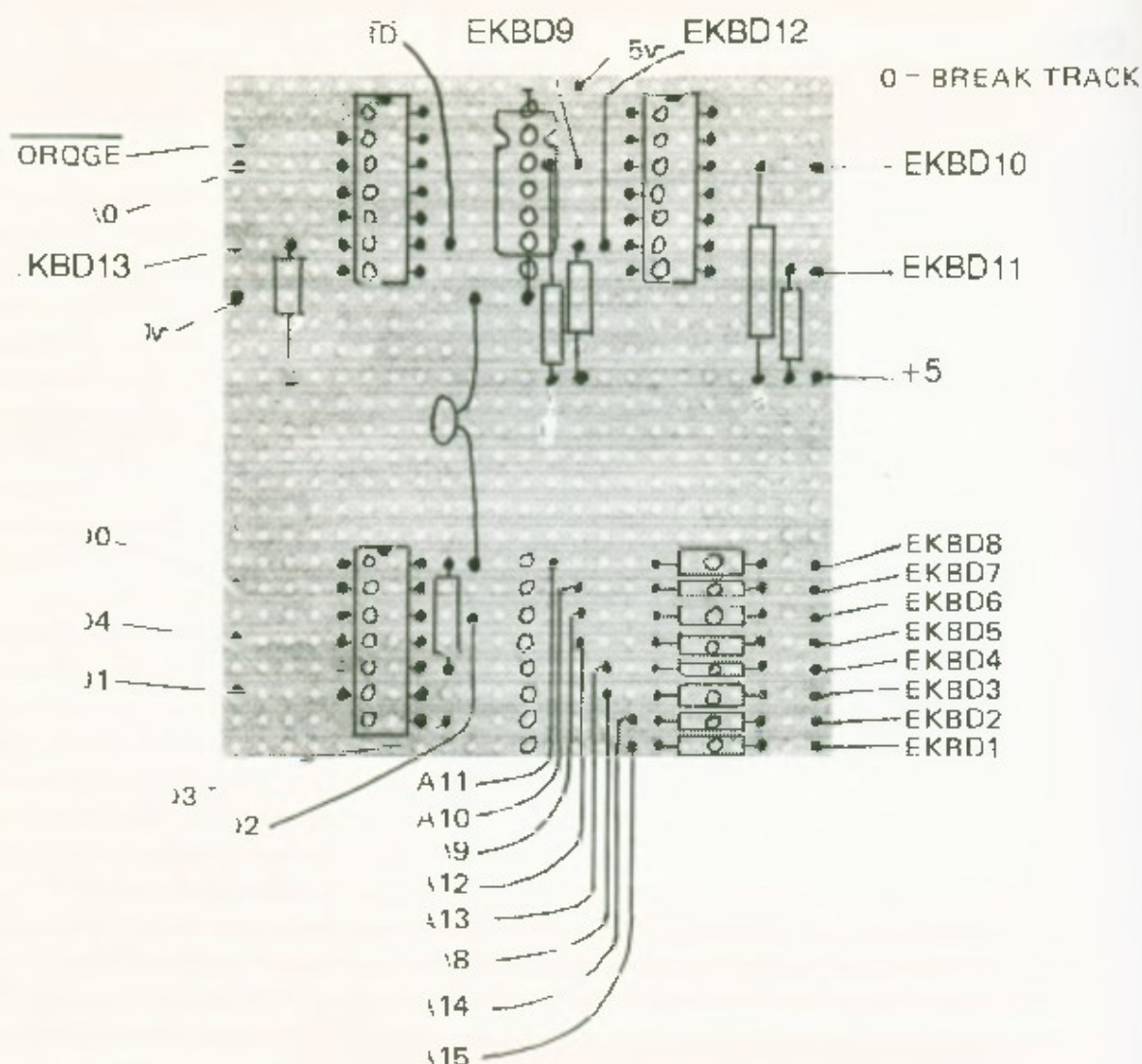
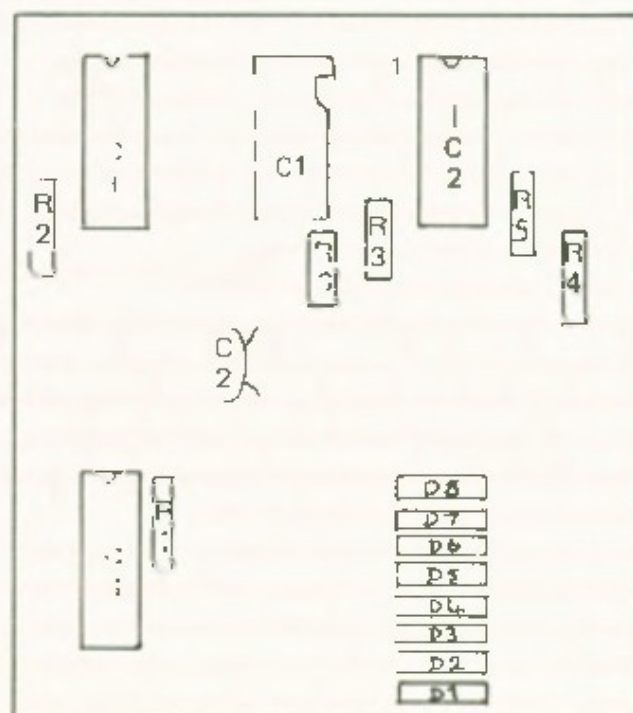
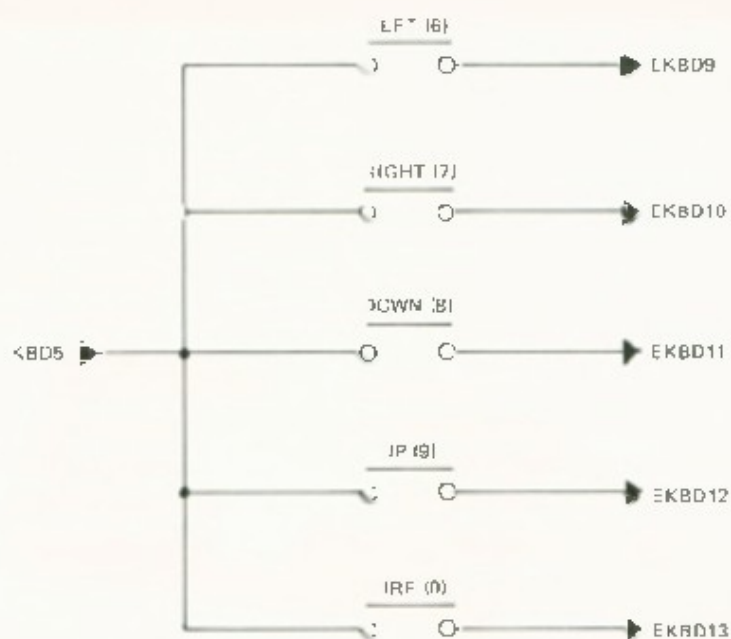


FIG 12b - ADDITIONAL KEYBOARD/JOYSTICKS  
COMPONENT LAYOUT DIAGRAM

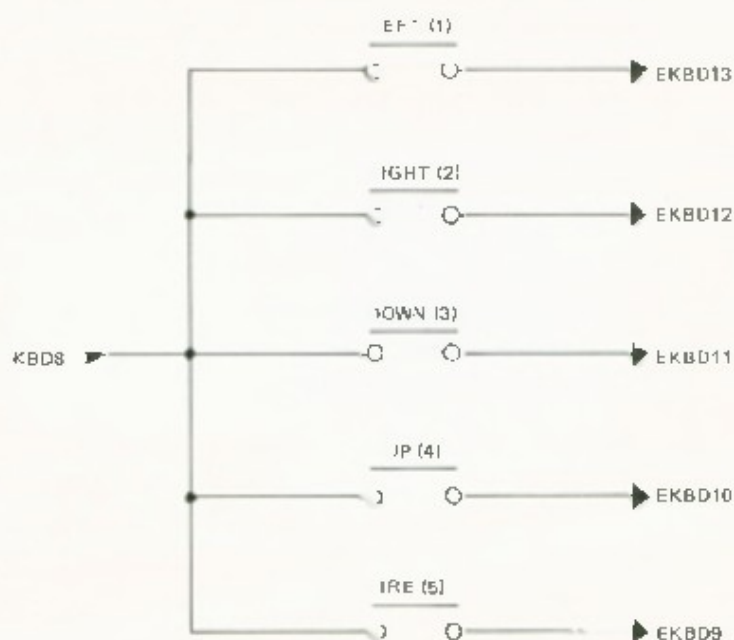


NOTE - Connections  
between components must  
be made as in the circuit  
diagram





IG 12c - JOYSTICK 1 CONNECTIONS



IG 12d - JOYSTICK 2 CONNECTIONS

## 7. SINCLAIR COMPATIBLE JOYSTICKS

A joystick, in this context, means some form of hand held device used to give movement instructions to programs. Sinclair joysticks will have a LEFT, RIGHT, UP, DOWN and FIRE function. The hand held unit could consist of a lever which operates one of four switches when pushed forwards, pulled back or moved from side to side. An alternative small hand held unit with a compact arrangement of pushbuttons is shown in fig 12e. The outer four buttons give the movement commands and the central one is a fire button.

The Sinclair joysticks appear from software to be identical to the top row of keys on the Spectrum keyboard, both from BASIC and machine code programs. Joystick 1 will appear on input port 61438 and joystick 2 on port 63486. The functions of the joysticks correspond to the number keys as follows:

Key	Function	
1	LEFT	
2	RIGHT	
3	DOWN	JOYSTICK 2
4	UP	
5	FIRE	
6	LEFT	
7	RIGHT	
8	DOWN	JOYSTICK 1
9	UP	
0	FIRE	

In order to add this control box to your Spectrum, the keyboard interface circuit of chapter 16 is required. Only the diodes D8 and D5 are used. The other diodes can be omitted.

The unit illustrated in fig 12e uses 5 push button switches for the control box.

The internal arrangement of buttons soldered onto a piece of veroboard is shown in fig 12f. The exact arrangement and type of buttons is left to the constructor. Internally the control box (es) must be wired in accordance with fig 12c (for joystick 1) and fig 12d (for joystick 2). The connections to the keyboard interface are clearly shown on these diagrams. If you are able to obtain a ready made joystick control box which makes contacts as in fig 12c and fig 12d then this can be connected to the keyboard interface circuit instead of a completely homebrew circuit.

All that you need now is some software to use with your joysticks. Any Sinclair joystick compatible software can be used. There follows a program named 'APOLLO' to enable you to experiment with the joystick control. Joystick 2 or the keyboard keys 1 — 5 are used.



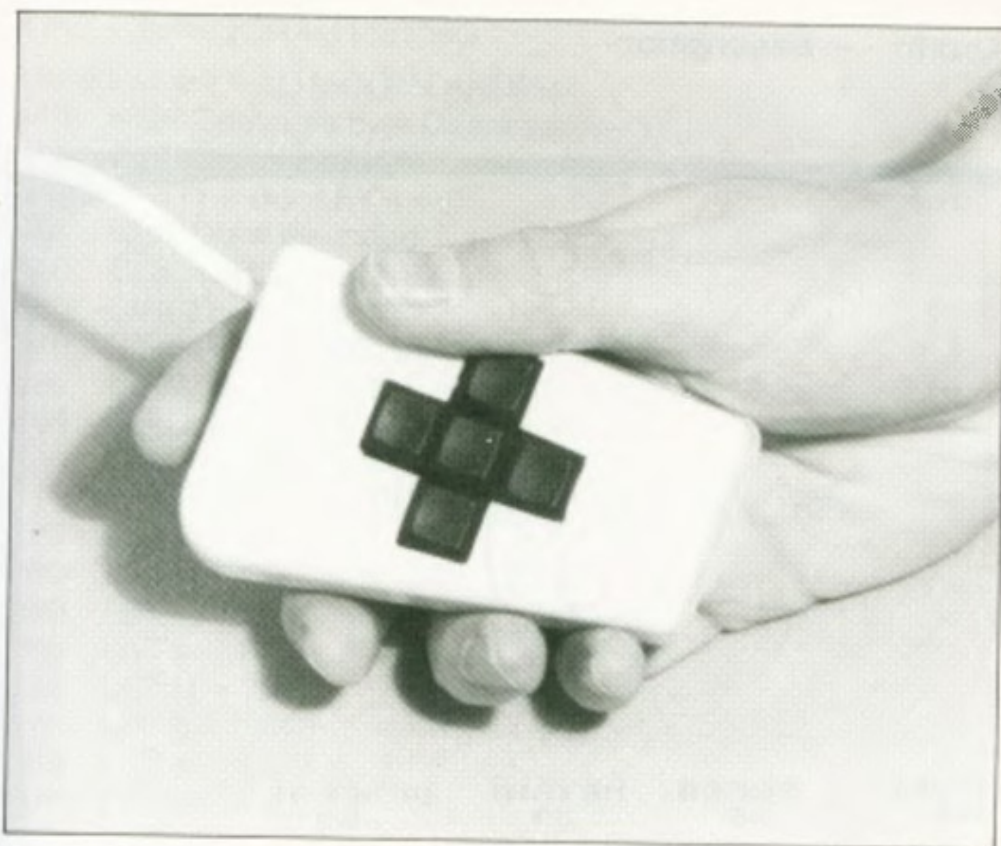


FIG 12e — JOYSTICK CONTROL IN HAND

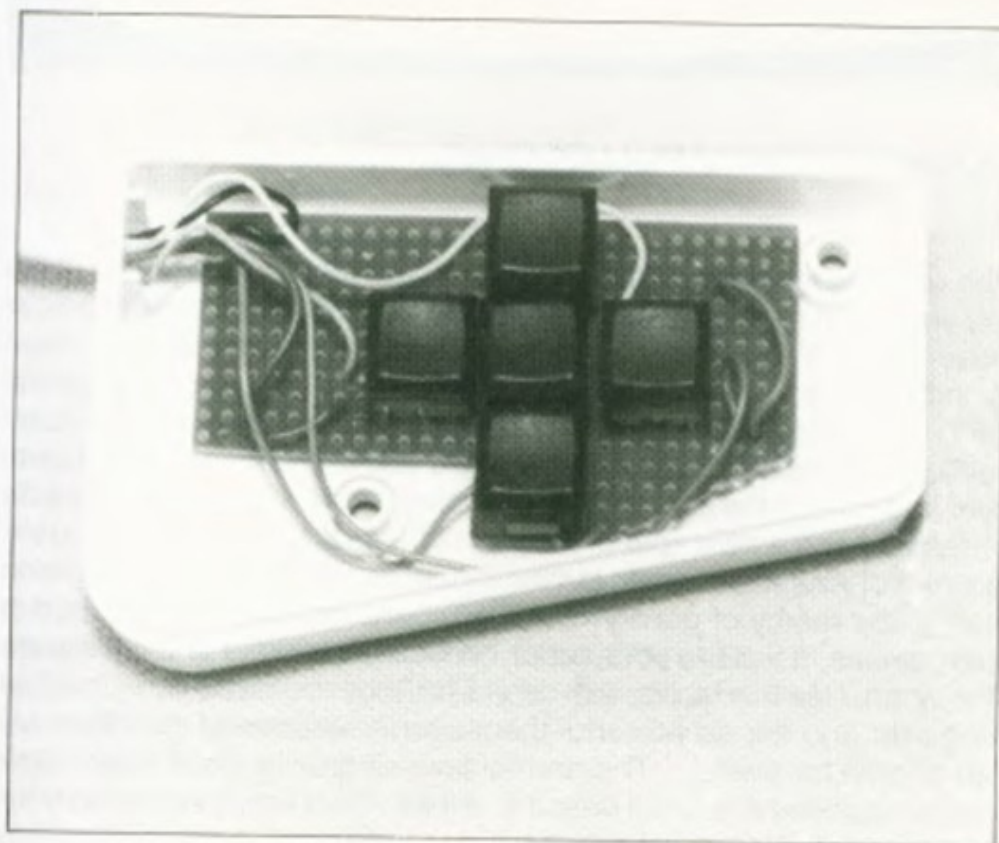
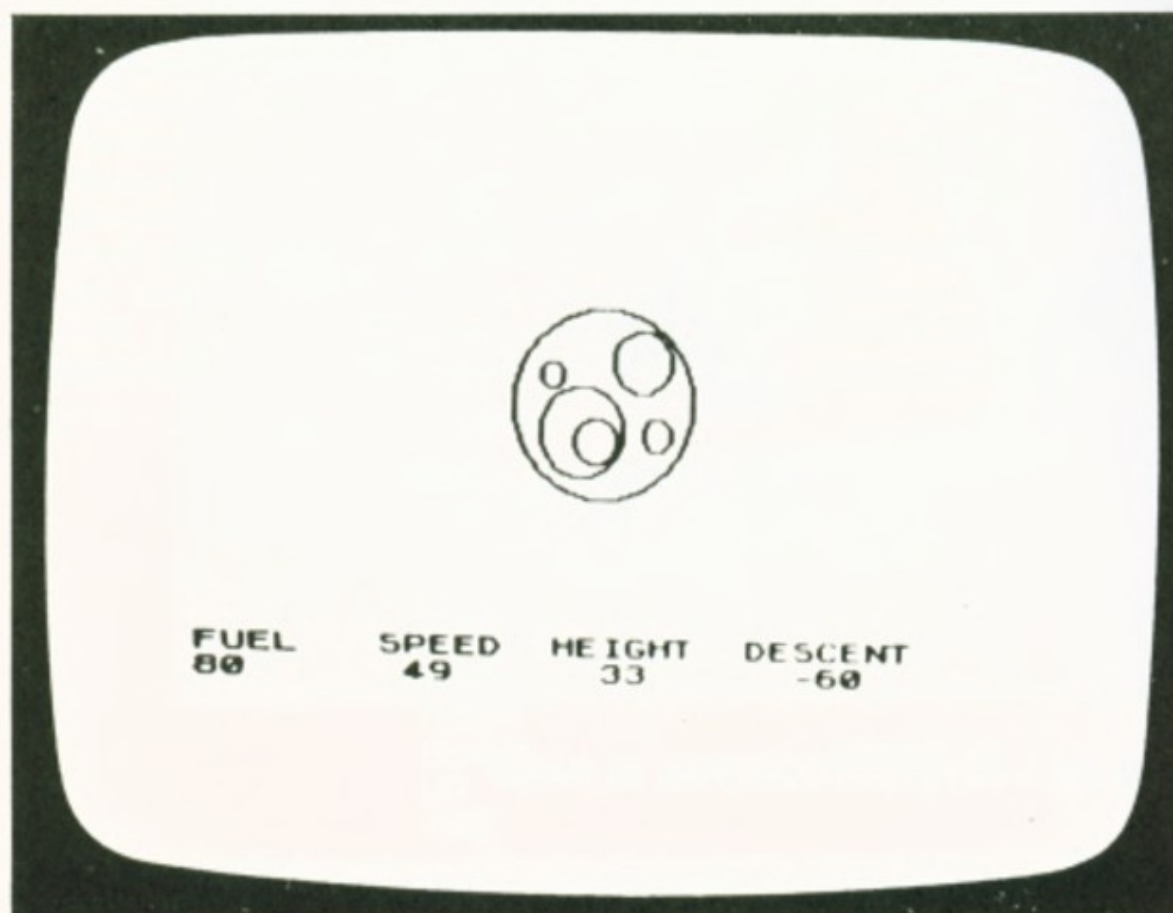


FIG 12f — INTERNAL LAYOUT OF HAND HELD JOYSTICK

## Program Apollo — description



IG 12d — DISPLAY FOR "APOLLO" GAME

The aim of the game is to successfully land a lunar module on the surface of the moon. Initially you are in a steady circular orbit. The joystick controls your thruster rockets. These apply thrust away from the moon (UP), towards the moon (DOWN), to increase tangential velocity (RIGHT) and to decrease tangential velocity (LEFT). You only use fuel when operating a thruster. A continuously updating readout of remaining fuel, speed, height above the lunar surface, and rate of descent appears on the screen. The objective is to land on the surface with a low rate of descent and a slow speed. If either of these is too large then you will crash. In the event of your fuel running out the thrusters will no longer operate and you will be left to the mercy of gravity! The FIRE button should only be used in absolute emergencies. It will fire your super boost retro rockets and accelerate away from the moon. Use this facility with care. The super boosters consume fuel at an alarming rate and are so powerful that several seconds of operation will boost you out of orbit for ever . . . The normal laws of gravity are obeyed. Slow down your orbital speed and you will drop into a lower orbit which increases your speed again. Applying full thrust towards the lunar surface will push you closer to the moon, but your orbital speed increases rapidly so that centrifugal acceleration throws you out further into space. Happy landings.



# APOLLO PROGRAM LISTING

```

00 REM APOLLO MOON LANDER
10  REM Copyright by A C Dickens
20  REM December 1982
30  LET r1 = 30 : LET s = 0
40  REM Draw the moon
50  CLS : CIRCLE 127, 87, r1
60  CIRCLE 110, 96, 4
70  CIRCLE 140, 100, 10
80  CIRCLE 125, 75, 7
90  CIRCLE 145, 77, 5
00  CIRCLE 120, 78, 14
10  PRINT AT 20, 0; "FUEL [ ] [ ] [ ] SPEED [ ] [ ] HEIGHT [ ] [ ] DESCENT"
20  LET x = 127
30  LET y = 157
40  LET r = 70
50  LET v = 1
60  LET kt = 70
70  LET p = 0.095
80  LET w = 1/70
90  LET am = 70
00  LET p1 = p/3
10  LET u = 0
20  LET k = 0
30  LET fu = 100
40  LET fl = 0.2
50  REM Get joystick input
60  IF fu = 0 THEN LET e = 31 : GO TO 380
70  LET e = IN 63486 - 224
80  LET f = INT (e/16) : REM FIRE key
90  LET e = e - f*16
00  LET f1 = INT (e/8) : REM LEFT key
10  LET e = e - f1*8
20  LET f3 = INT (e/4) : REM DOWN key
30  LET e = e - f3*4
40  LET f2 = INT (e/2) : REM RIGHT key
50  LET e = e - f2*2
60  LET f4 = INT e : REM UP key
70  LET f1 = (1 - f1) * p
80  LET f2 = (1 - f2) * p1
90  LET f3 = (1 - f3) * p
00  LET f4 = (1 - f4) * p1
10  IF f = 0 THEN LET fu = fu - 10 : LET f1 = 5 * p
20  IF f1 + f2 + f3 + f4 > 0 THEN LET fu = fu - f1
30  REM Calculate next position
40  LET f = (kt/r - v*v)/r
50  LET am = am + (f2 - f4) * r
60  LET u = u + (f1 - f3 - f)
70  LET r = r + u
80  LET v = am/r

```

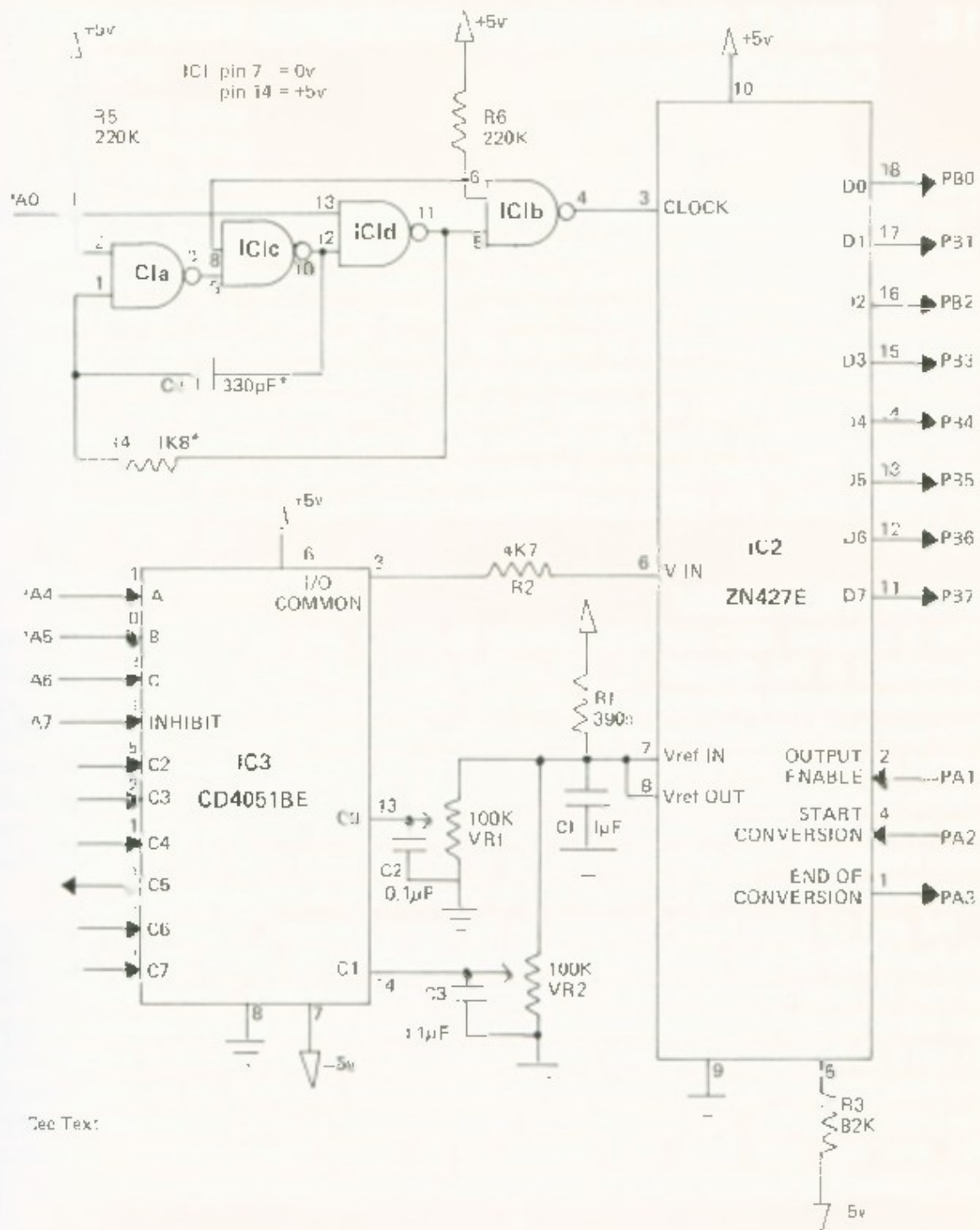
```

390 LET j = v/r
300 LET k = k + j
310 REM Clear previous spaceship
320 PLOT INVERSE 1: x,y
330 PLOT INVERSE 1: x, y+1
340 PLOT INVERSE 1: x-1,y
350 LET x = 127 + r*SIN k
360 LET y = 87+r*COS k
370 IF x < 225 AND x >= 1 AND y >= 0 AND y <= 174 THEN GO TO 710
380 LET x = 0
390 LET y = 0
400 GO TO 750
410 REM Draw new spaceship
420 PLOT x, y
430 PLOT x-1,y
440 PLOT x,y+1
450 IF fu < 0 THEN GOSUB 950
460 LET h = r-r1
470 PRINT AT 21, 0; INT (fu*10)/10; " □ □ ";
480 PRINT AT 21, 8; INT (v*100); " □ □ ";
490 PRINT AT 21, 16; INT h; " □ □ ";
500 PRINT AT 21, 24; INT (u*100); " □ □ ";
510 IF h <= 0 THEN GO TO 830
520 GO TO 350
530 REM At lunar surface
540 IF ABS v > 0.04 THEN GO TO 900
550 IF ABS u > 0.15 THEN GO TO 900
560 CLS
570 PRINT AT 20, 0; "SAFE LANDING - WELL DONE!"
580 FOR x = 0 TO 300: NEXT x
590 GO TO 100
600 REM Impact with the moon
610 CLS
620 PRINT: FLASH 1; "TOO FAST >>> YOU CRASHED!"
630 PAUSE 200
640 GO TO 100
650 REM Out of fuel
660 PRINT AT 20, 0; FLASH 1; "EMPTY";
670 LET fu = 0
680 RETURN

```

Note: □ indicates a space.





See Text:

IG 13a – 8 CHANNEL ANALOGUE TO DIGITAL CONVERTER CIRCUIT

## 8. EIGHT CHANNEL ANALOGUE TO DIGITAL CONVERTER

### Introduction

You will appreciate from reading earlier chapters in this book that the computer deals in digital quantities. This leaves only two options for each bit of information. It must be either 1 or 0. In the real world, quantities which we may wish to measure can often take a wide range of values. Some examples are blood pressure, temperature, light intensity, or volume of sound. These parameters can usually be expressed as a variable voltage by using a special transducer. An analogue to digital converter (ADC) will accept as its input a variable voltage (say 0 — 2.55 volts) and provide a digital output which can be read by a computer. In this circuit an 8 bit ADC has been used, and consequently its output can vary from 0 — 255 in integral steps. The circuit can therefore measure down to an accuracy of 0.01 volts.

### Construction details

The 8 channel ADC connects directly to port A and port B of the PIO chip described in the chapter 15. The author used a 25 way D type connector to facilitate removal of devices from the PIO. The unit in use is shown in fig 13c. You can see the D type connector protruding from the side of the box. The component layout on 0.1" pitch veroboard is shown in fig 13b. Make the breaks in the copper tracks as shown before soldering in the components. It is recommended that all of the integrated circuits should be plugged into sockets and not soldered directly onto the board. This will make it easier to replace faulty chips and eliminates the possibility of damaging the chips due to overheating during soldering.

R4 and C4 have been asterixed on the layout diagram. These are the timing components for the clock circuit. The photograph of the circuit board in fig 13d shows R4 as a variable resistor in series with a fixed resistor. This enables you to set the oscillator to its maximum of 600KHz, however it is not normally necessary to run it at this speed because BASIC is so slow. It is also difficult to ensure that the frequency is not above 600KHz unless you have access to an oscilloscope or frequency counter. You are therefore recommended to omit the variable resistor R4 and use one fixed 1K8 resistor in its place. Interconnection between components should be made in accordance with the circuit diagram in fig 13a.



## COMPONENTS LIST FOR THE 8 CHANNEL ADC

### RESISTORS

- R1 390 ohms
- R2 4K7 ohms
- R3 82K ohms
- R4\* 1K8 ohms (or 330 ohms in series with a 2K2 preset — see text)
- R5 220K ohms
- R6 220K ohms

all ¼ watt 5%

### CAPACITORS

- C1 1 uF
- C2 0.1 uF
- C3 0.1 uF
- C4 330 pF
- C5 22 uF 6v

### INTEGRATED CIRCUITS

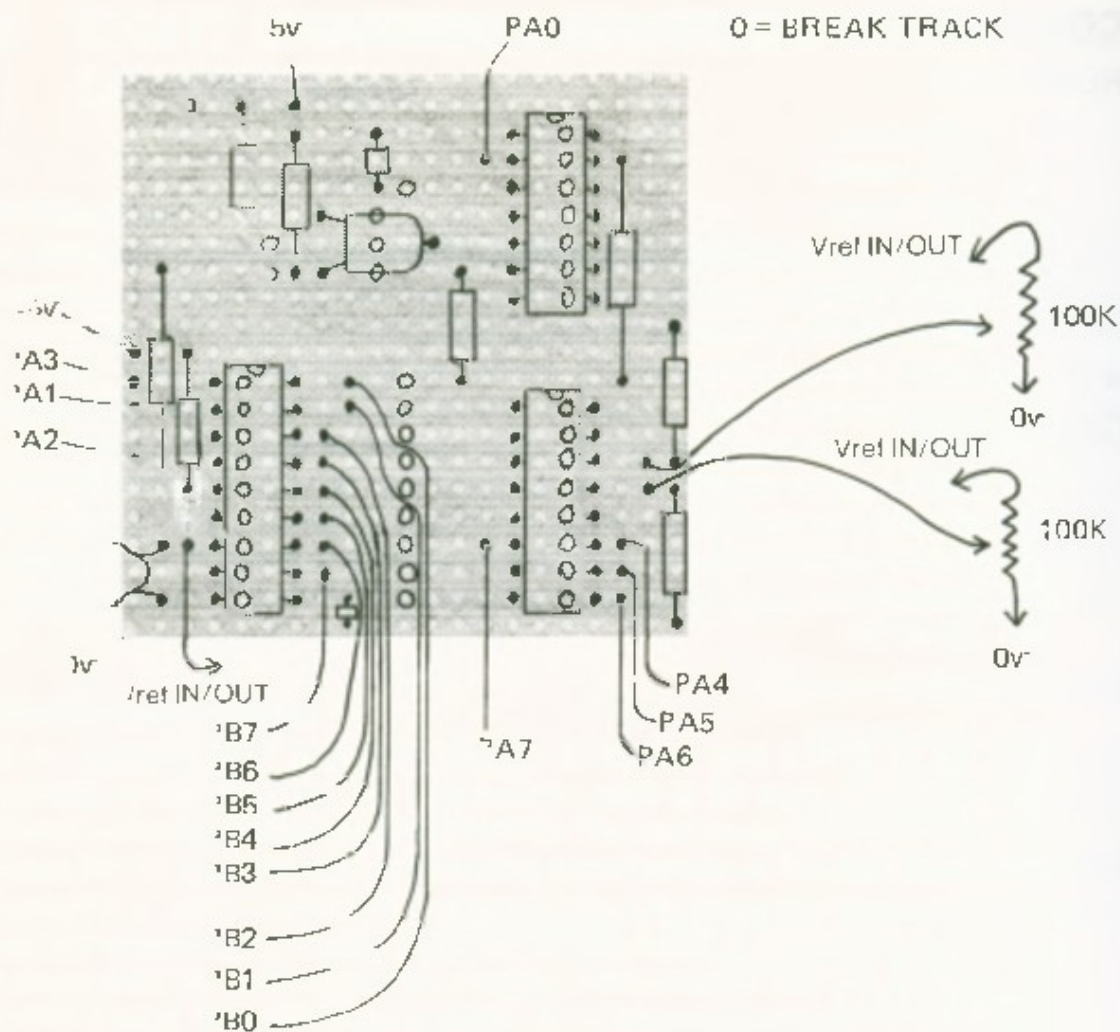
- C1 CD4011BE
- C2 ZN427E
- C3 CD4051BE

### MISCELLANEOUS

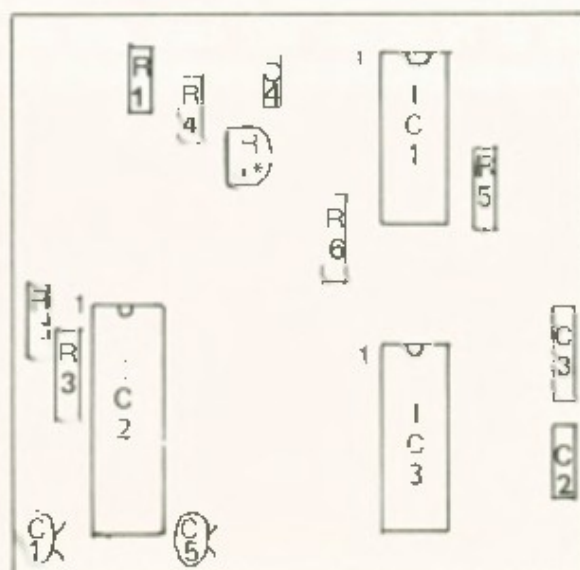
- R1,VR2 100K ohm X and Y axes joystick
- 1.1" copper strip board
- sockets for IC's
- case
- connector to PIO

Note — if you have any difficulty obtaining a suitable 2 axis joystick, it is available from:

Maplin Electronic Supplies,  
P.O. Box 3,  
Hayleigh,  
SSEX SS6 2BR



IG 13b -- 8 CHANNEL ADC COMPONENT LAYOUT DIAGRAM



NOTE -- Connections between components must be made as in the circuit diagram



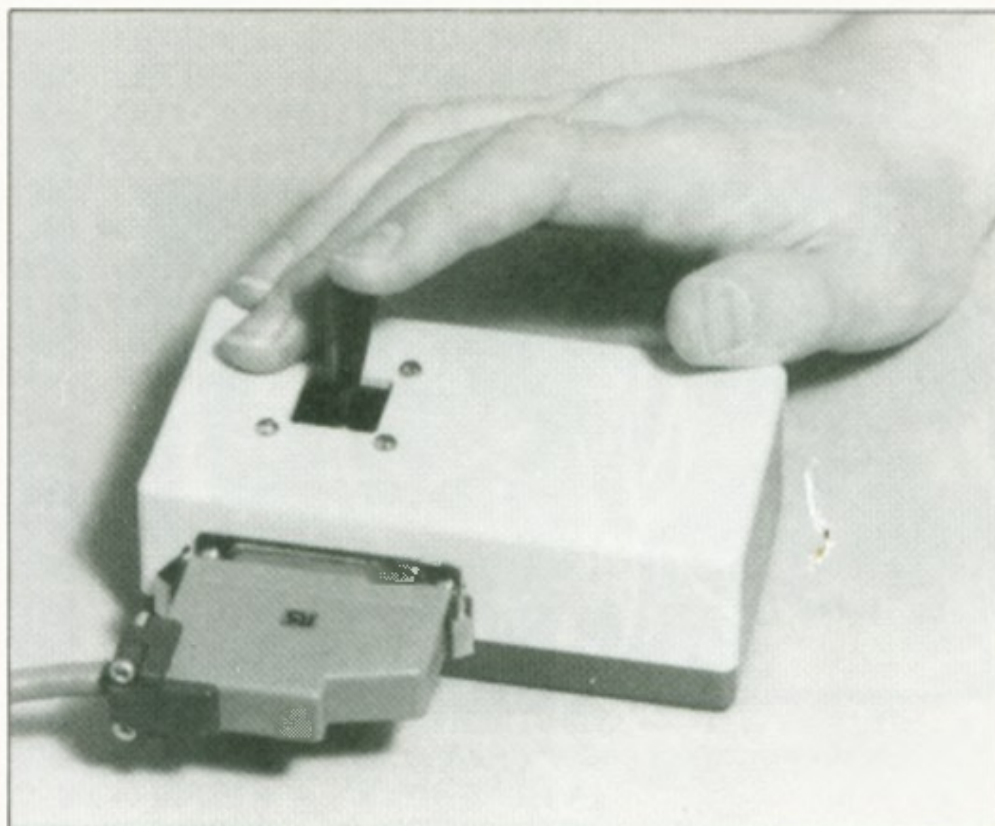


FIG 13c – 8 CHANNEL ADC IN USE WITH A JOYSTICK

### HOW IT WORKS

The heart of the circuit is the analogue to digital converter chip (IC2). Its objective is to accept a voltage at its  $V_{in}$  input and convert it to an 8 bit digital output. To do this it requires several signals from the PIO chip plus a clock. The clock is similar to the Z80A clock in your Spectrum, but operates at a lower frequency. On each clock pulse the chip performs the next bit of the conversion.

The frequency of the clock is about 500 kHz, and it is turned on and off by PA0. Each analogue to digital conversion therefore takes about 20  $\mu$ s. The conversion is initiated by PA2 operating the start of conversion pin and PA0 switching on the clock. IC1 together with its associated resistors and capacitor forms the clock circuit. When the conversion is complete, IC2 sends an end of conversion signal to PA3. This indicates that the output data byte can then be read. PA1 operates the output enable pin. Port B of the PIO can then be used to read in the 8 bit data byte from the ADC.

In the sample BASIC drawing program, PA0, PA1, and PA2 are all operated simultaneously. The end of conversion signal will be finished in the time it takes BASIC to complete the coordinate read routine. If machine code were to be used then it would be necessary to use the end of conversion signal. This is because machine code operates so much faster than BASIC. The ADC may not have completed the conversion, and therefore it must be checked.

The ADC chip is fairly expensive, so if we wish to measure analogue voltages from more than one source, it is better if we can arrange this with only one ADC

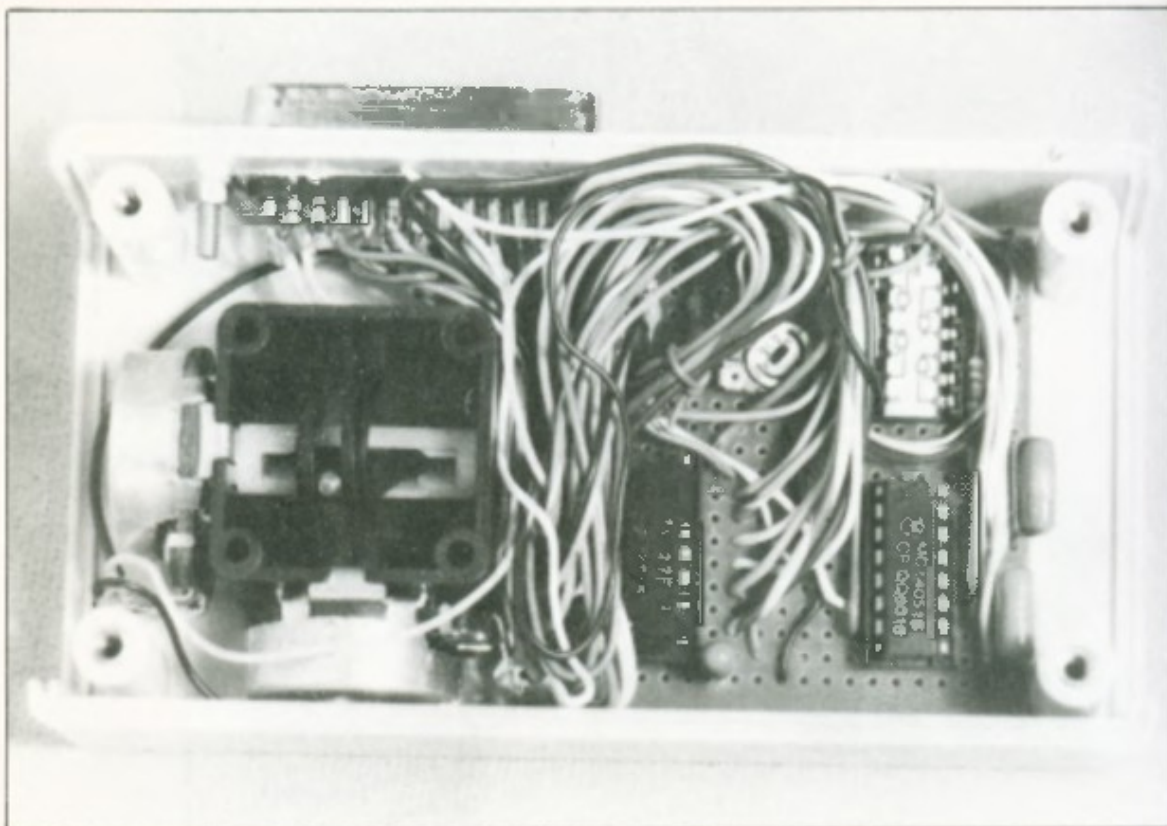


FIG 13d - 8 CHANNEL ADC INTERNAL LAYOUT

chip. This is where IC3 comes in useful. This is a 1 to 8 line analogue multiplexer chip. An address from 0 to 7 can be applied to it via PA4, PA5 and PA6. This enables any one of the 8 inputs to be connected via the multiplexer to the common output. If we select address 0 for example, the input voltage applied at I0 will appear at the I/O common pin. The chip is conceptually similar to an 8 way pole switch. Changing the address to the chip is like changing the position of the switch.

The BASIC drawing program will operate with the X and Y axes of the joystick connected channel to 0 and channel 1 respectively. Lines 10 to 90 set up the PIO port so that port B is all input and port A is all output except for PA3. Consider lines 140 to 170, which read in the X coordinate. First of all, port A outputs are set to zero. This stops the clock, selects multiplexer channel 0, and disables the ADC chip. Then in line 160, the clock is started simultaneously with the start of conversion signal to the ADC chip. By the time that line 170 reads in the X coordinate from port B, the conversion is complete.

The BASIC drawing program itself is very easy to use. There are two modes, 'D' and 'S', each entered by pressing the appropriate key on the Spectrum keyboard. In 'S' mode a small flashing dot can be moved around the screen with the joystick, but it doesn't leave any mark. If 'D' is pressed then the moving dot leaves a line behind it. You can draw pictures in this way. Typing 'S' again stops the drawing until you press 'D'. To start a new drawing press 'N'.



## DRAWING PROGRAM LISTING

```
0  REM JOYSTICK DRAWING PROGRAM
10  REM
20  REM SET UP THE PIO PORTS
30  LET DA = 31: LET CA = 95
40  LET DB = 63: LET CB = 127
50  OUT CA.BIN 11111111
60  OUT CA.BIN 00001000
70  OUT CB.BIN 11111111
80  OUT CB.BIN 11111111
90  LET A$ = "S"
100 LET X1 = 0: LET Y1 = 0
110 REM SET UP THE SCREEN FOR PLOTTING
120 BORDER 7: PAPER 7: INK 0: CLS
130 REM READ X COORDINATE
140 OUT DA.BIN 00000000
150 OUT DA.BIN 00000111
160 LET X = IN DB
170 REM READ Y COORDINATE
180 OUT DA.BIN 00000000
190 OUT DA.BIN 00010111
200 LET Y = IN DB*175/255
210 REM D=DRAW, S=STOP DRAW, N=NEW PICTURE
220 IF INKEY$ = "D" THEN LET A$ = "D"
230 IF INKEY$ = "S" THEN LET A$ = "S"
240 IF INKEY$ = "N" THEN GOTO 100
250 IF A$ = "D" THEN GOTO 350
260 REM FLASH DRAWING POINT
270 PLOT OVER 1: X,Y
280 REM GENERATE DELAY
290 PAUSE 10
300 PLOT OVER 1: X,Y
310 LET X1 = X
320 LET Y1 = Y
330 GO TO 140
340 REM DRAW
350 PLOT X1, Y1
360 DRAW X-X1, Y-Y1
370 GO TO 320
```

## 9. A CENTRONICS PARALLEL PRINTER INTERFACE

### INTRODUCTION

The Spectrum has been designed to operate with a variety of Sinclair peripherals. Two of these can be used to obtain printed output on paper. The first is the ZX Printer, which produces output on narrow strips of aluminised paper. Such output is normally sufficient for program listings, but unsuitable for most professional applications such as word processing. High quality printers for such applications do exist. They are generally provided with one of two standard printer interfaces - Centronics parallel or RS232 serial. The Zx Interface 1 peripheral allows RS232 printers to be connected directly to the Spectrum. However, most modern printers are equipped with Centronics parallel interfaces as standard, and no Sinclair interfaces are yet available to allow such printers to be connected directly to the Spectrum. This chapter explains how to make a suitable Centronics parallel printer interface using just one Z80A PIO chip. A comprehensive interface program is also provided, for use with a 16K or 48K Spectrum.

### CIRCUIT DESCRIPTION

In order to construct the parallel printer interface, the Z80A PIO chip must be connected to the Spectrum rear edge connector. This is explained in Chapter 15. Having wired up the Z80A PIO chip to the Spectrum, it is then necessary to connect it up to the printer. The connections should be made as illustrated in Figure 14a. Some form of multicore cable will be required to connect between the PIO and printer plug. The type used will be dependent upon the Centronics plug employed to connect to the printer. Two common types exist: an insulation displacement type and an individual wire type. Either type will plug into the mating socket on the printer. The former is designed for use with ribbon cable (strands of wire moulded side by side in the form of a ribbon). This type is very easy to use, because no soldering is required at the printer end of the interface cable. The second type is designed to have individual wires soldered onto the terminals inside the connector. The type used will be determined by availability and/or personal preference.



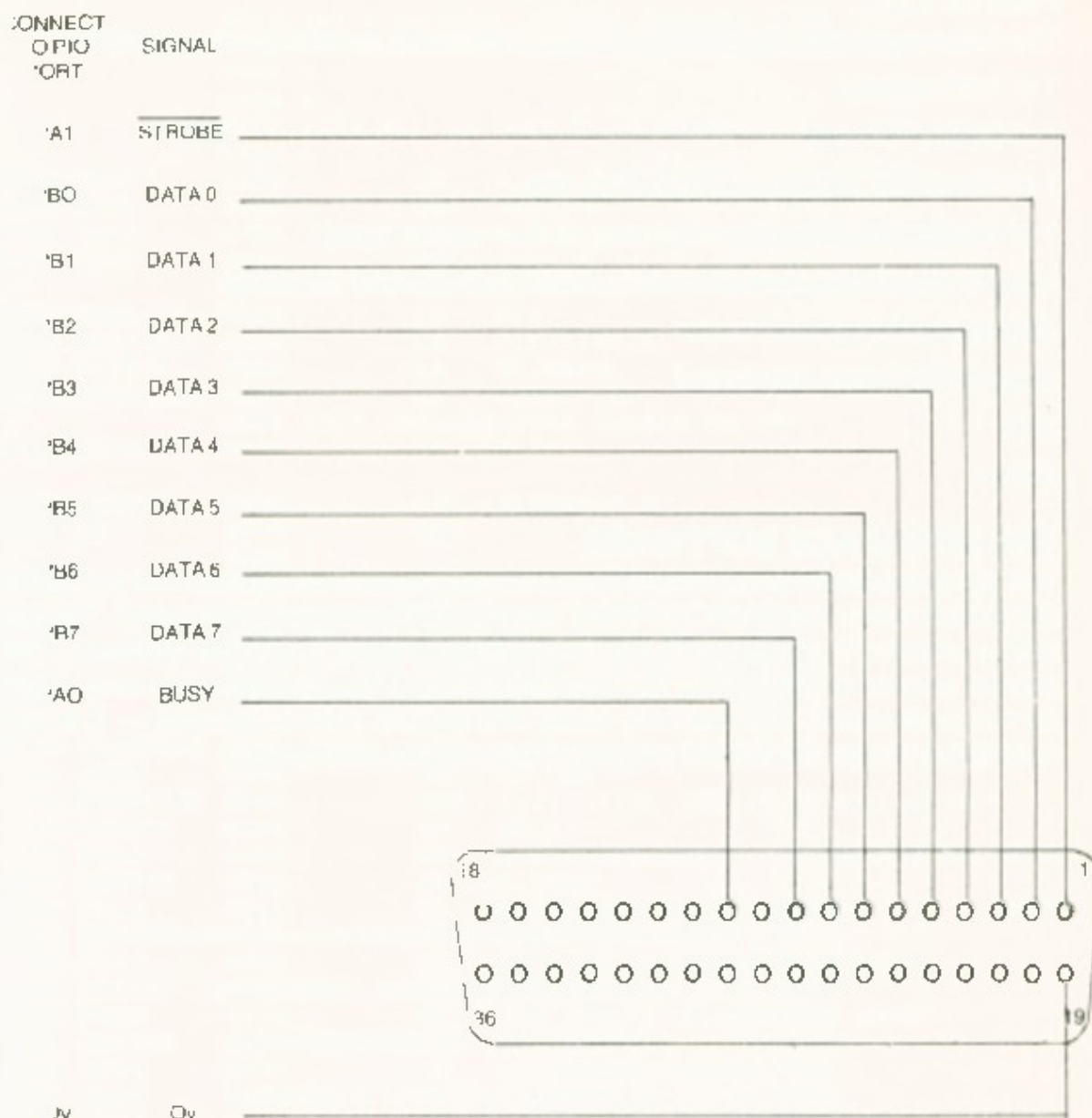


FIG. 14a — CONNECTIONS BETWEEN Z80A PIO AND PRINTER

## ARTS LIST

The following parts are required for the Centronics parallel printer interface:

- of 10K resistor
- of Z80A PIO chip
- of 40 pin DIL socket for Z80A PIO chip
- of Centronics 36-way printer plug (IDC or solder type)
- length of suitable interconnection cable (with at least 12 cores)
- 8-way card edge connector
- piece of 0.1" pitch veroboard
- wire
- solder

## THE INTERFACE SOFTWARE

The job of the interface software is to send the correct characters to be printed on the printer when the BASIC 'LPRINT' and 'LLIST' commands are used. The use of this software is explained in the next section entitled 'Using the interface'. This current section explains how the program works at the machine code level. If you do not understand Z80 machine code, this section may be ignored — it is only necessary to use the BASIC programs listed in the next section.

### EAP Z80 Assembler — Source Listing

```
F4E          0100          ORG  £7F4E
              0110 ; *****
              0120 ;
              0130 ;    CENTRONICS PARALLEL INTERFACE
              0140 ;
              0150 ;    for a 16k ZX SPECTRUM
              0160 ;
              0170 ;    (c) Copyright A C DICKENS
              0180 ;
              0190 ;    January 1984
              0200 ;
              0210 ; *****
              0220 ;
F4E 005F     0230 CA      EQU  95 ; CONTROL PORT A
F4E 007F     0240 CB      EQU 127 ; CONTROL PORT B
E4E 001F     0250 DA      EQU  31 ; DATA PORT A
E4E 003F     0260 DB      EQU  63 ; DATA PORT B
              0270 ; Line length storage byte
E4E 50B1     0280 LENGTH EQU  £50B1 ;Decimal 23681
              0290 ;RESERVED WORD TABLE start address
F4E 0095     0300 TABLE EQU  £0095
              0310 ;PRINTER ROUTINE ADDRESS VECTOR
E4E 50C5     0320 ECHO  EQU  £50C5
              0330 ;
              0340 ;INIT sets up the PIO ports and the
              0350 ;printer routine vector.
              0360 ;
E4E 31FF     0370 INIT  LD   A,255
E50 D35F     0380      OUT (CA),A
E52 3FFD     0390      LD   A,253
```



```

E54 D35F      0400      OUT  (CA),A
               1410 ;A1 = printer STROBE (output from P10)
               1420 ;A0 = printer BUSY line (input to P10)
               1430 ;
               1440 ;
               1450 ;SET ALL of port B as outputs
               1460 ;

E56 3EF1      0470      LD   A,255
E58 D37F      0480      OUT  (CB),A
E5A 3E00      0490      LD   A,0
E5C D37F      0500      OUT  (CB),A
               1510 ;SET a default of NO LF after a CR
E5E 3E00      0520      LD   A,0
E60 FD7776     0530      LD   (IY+118),A
E62 01707E     0540      LD   RC,MAIN
E64 ED43C55C   0550      LD   (ECHO),BC
               1560 ;SET default line length to 70
E66 3F46      0570      LD   A,70

E6C 32815C     0580      LD   (LENGTH),A
E6F C9        0590      RET
               1600 ;
               1610 ;MAIN output routine is entered with the
               1620 ;character to be printed in the A reg.
               1630 ;Reserved words, TAB, AT, commas in a
               1640 ;PRINT statement and standard ASCII
               1650 ;characters are all dealt with.
               1660 ;

E70 FDCB7646   0670 MAIN   BIT   0,(IY+118) ;Was last char a TAB?
E74 205C       0680      JR    NZ,TAB
E76 FDCB764E   0690      BIT   1,(IY+118) ;Was last chr an AT?
E7A C2007F     0700      JP    NZ,AT
E7D FE06       0710      CP    6 ;PRINT comma?
E7F CA097F     0720      JP    Z,PCOM
E82 FE16       0730      CP    22 ;A1 control character?
E84 CA197F     0740      JP    Z,PAT
E87 FE17       0750      CP    23 ;TAB control character?
E89 CA1C7F     0760      JP    Z,PTAB
E8C FEAB       0770      CP    165 ;Is it a reserved word?
E8E D2277F     0780      JP    NC,RESWRD
E91 FE0D       0790      CP    13 ;ENTER code?
E93 2B2C       0800      JR    Z,ENTER
E95 FE20       0810      CP    32 ;Control code?
E97 DB         0820      RET   C ;Ignore control codes
E98 FE80       0830      CP    128 ;Standard ASCII character?
E9A 3802       0840      JR    C,PROUT
               1850 ;
               1860 ;It must be a user defined graphic if
               1870 ;here, so output a SPACE
               1880 ;

E9C 3E20       0890 SPCOUT LD   A,32
E9E FD3477     0900 PROUT  INC   (IY+119) ;Increment CHR counter
FA1 47         0910      LD   B,A
CA2 CAB15C     0920      LD   A,(LENGTH)
EAS FDBE77     0930      CP    (IY+119)
EAB 3014       0940      JR    NC,PRINT
               1950 ;
               1960 ;Maximum allowed line length has been
               1970 ;reached, so go to new line
               1980 ;

FAA 3E0D       0990      LD   A,13 ;ENTER code
EAC CD457F     1000      CALL OUTPUT
EAF FD367701   1010      LD   (IY+119),1
EB3 FDCB7656   1020      BIT   2,(IY+118) ;Should an LF be supplied?

```

EB7 2805	1030	JR	Z, PRINT
	040	;Output an LF	
EB9 3E0A	1050	LD	A, 10 ;LF code
EBB CD457F	1060	CALL	OU{PUT
EBF 78	1070	PRINT	LD A, B ;Restore value in the A reg.
EBF 180F	1080	JR	LINK
	090	;	
	100	;Deal with an FNIER code	
	110	;	
EC1 FD367700	1120	ENTER	LD (IY+119), 0
EC5 FDCB7656	1130	BIT	2, (IY+118)
EC9 287A	1140	JR	Z, OU{PUT
ECB CD457F	1150	CALL	OUTPUT
	160	;Supply a LF	
ECE 3E0A	1170	LD	A, 10
ED0 1873	1180	LINK	JR OU{PUT
	190	;	
	200	;Deal with TABs	
	210	;	
ED2 FDCB76A6	1220	TAB	RES 0, (IY+118)
ED6 FDBE77	1230	CP	(IY+119)
ED9 3807	1240	JR	C, TAB1 ;Fast print head?
EDB FD9677	1250	SUB	(IY+119)
EDE 47	1260	LD	B, A
EDF 04	1270	INC	B
EE0 1819	1280	JR	TAB2
EE2 47	1290	TAB1	LD B, A
EE3 04	1300	INC	B
EE4 FD3677FF	1310	LD	(IY+119), 255
EE8 3E0D	1320	LD	A, 13 ;ENTER
EEA CD457F	1330	CALL	OUTPUT
EED FDCB7656	1340	BIT	2, (IY+118) ;LF after ENTER?
EF1 2805	1350	JR	Z, TAB3
EF3 3E0A	1360	LD	A, 10
EF5 CD457F	1370	TAB4	CALL OUTPUT
EF8 FD3477	1380	TAB3	INC (IY+119)
FFB 3E20	1390	TAB2	LD A, 32 ;SPACE character
EFD 10F6	1400	DJNZ	TAB4
FFF C9	1410	RET	
	420	;	
	430	;Deal with AI code	
	440	;	
F00 FDCB76C6	1450	AI	SET 0, (IY+118)
F04 FDCB768E	1460	RES	1, (IY+118) ;fake 2nd value after AI
F08 C9	1470	RET	
	480	;	
	490	;Deal with PRINT comma code	
	500	;	
F09 FD7E77	1510	PCOM	LD A, (IY+119)
F0C E60F	1520	AND	15 ;Columns are 15 spaces apart
F0F C8	1530	KEI	Z
F0F 3E20	1540	LD	A, 32 ;SPACE
F11 CD457F	1550	CALL	OUTPUT
F14 FD3477	1560	INC	(IY+119)
F17 18F0	1570	JR	PCOM
	580	;	
	590	;Skip first parameter after an AI	
	600	;	
F19 FDCB76CE	1610	PAI	SET 1, (IY+118)



```

F1D C9      1620      RET
              630 ;
              640 ;Record that last code was a TAB so that
              650 ;a TAB parameter is expected next time
              660 ;this routine is called.
              670 ;
F1E FDCB76C6 1680 PTAB   SET  0,(1Y+118)
F22 FDCB768E 1690      RFS  1,(1Y+118)
F26 C9      1700      RET
              710 ;
              720 ;Print out a reserved word
              730 ;
F27 219500   1740 RESWRD LD   HL,TABLE ;Reserved words in ROM
F2A D6A4     1750      SUB  164
              760 ;Scan the reserved word table until A
              770 ;equals 0. Each reserved word is stored
              780 ;as its ASCII codes with the last letter
              790 ;having BIT 7 set to a 1.
              800 ;
F2C CB7E     1810 RESW1   BIT  7,(HL)
F2E 23       1820      INC  HL
F2F 28FB     1830      JR   7,RESW1
F31 3D       1840      DEC  A
F32 FD00     1850      CP   0
F34 20F6     1860      JR   NZ,RESW1
              870 ;
              880 ;HL now points to the start of the
              890 ;correct word. Output a SPACE first.
              900 ;
F36 CD9C7E   1910      CALL SPCOUT
F39 7E       1920 RESWOT LD   A,(HL)
F3A 23       1930      INC  HL
              940 ;
              950 ;Output a character of the reserved
              960 ;word taking care of line length.
              970 ;
F3B CD9E7E   1980      CALL PROUT
F3E CB7F     1990      BIT  7,A ;End of word?
F40 28F7     2000      JR   7,RESWOT
              2010 ;
              2020 ;Output a SPACE to terminate word
              2030 ;
F42 C39C7E   2040      JP   SPCOUT
              2050 ;
              2060 ;OUTPUT routine sends the character in
              2070 ;the A reg. to the printer. If the printer
              2080 ;is BUSY then the routine waits for it
              2090 ;to become READY.
              2100 ;
F45 F5       2110 OUTPUT PUSH AF
F46 D33F     2120      OUT  (DB),A
              2130 ;WAIT for BUSY to go low
F48 DB1F     2140 LOOP   IN   A,(DA)
F4A CB47     2150      BIT  0,A
F4C 20FA     2160      JR   NZ,LOOP
              2170 ;
              2180 ;STROBE data into the printer
              2190 ;
F4E 3EFD     2200      LD   A,FFD
F50 D31F     2210      OUT  (DA),A
F52 3EFF     2220      LD   A,EFF
F54 D31F     2230      OUT  (DA),A
F56 F1       2240      POP  AF
F57 C9       2250      RET

```

The following paragraphs describe the operation of the above Assembler program.

The BASIC 'LLIST' and 'LPRINT' commands have been designed so that the character output routine address is stored in RAM at £5CC5 (23749 decimal). Normally, this would be set to point to the character output routine contained within the BASIC ROM. However, by POKEing a new output routine address into £5CC5, characters to be output can be routed via a new, user defined routine. This is the basis of the new output routine listed here.

The first part of the program labelled as routine INIT is the initialisation routine. The PIO chip is initialised so that all port B lines are defined as outputs (for the 8 data lines). Only A0 and A1 of port A are used. A0 is used as an input from the printer BUSY line, and A1 is used as a STROBE output to the printer. The defaults for linefeeds and carriage returns are also set up.

At this stage, it is worth noting the use of the Z80 IY register. In the Spectrum it is always set to £5C3A (23610 decimal). Therefore, location IY+118 is 23728 and IY+119 is location 23729. If you look at the list of the Spectrum System Variables chapter in the BASIC programming manual, you will see that locations 23728 and 23729 are not used. This program therefore makes good use of them. Location IY+118 (23728) records if a linefeed should be generated after carriage return and also whether or not the previous print instruction was an AT or a TAB. Location IY+119 (23729) stores the number of characters which have been printed out on the current line. The only other variable used by this program is labelled LENGTH. This is in location 23681, again an unused byte in the System Variables area. It contains the number of characters per line of the printer, and may take any value between 1 and 255.

Finally, at the end of the INIT routine, the output routine vector (labelled ECHO) is set to point to the start of the output program, MAIN. Routine INIT is called once to initialise the parallel printer. The parallel printer will then remain in use until the BASIC command 'NEW' is performed, or until a different address is POKEd into location ECHO.

Routine MAIN expects to be entered with the character to be printed contained in the Z80's A register. It will perform in the following ways, dependent upon the character in the A register:

#### A=0-31

These are the Control Codes, as listed in Appendix A of the *Spectrum BASIC programming* manual. Of all of the control codes available, only four are directly relevant to the printer output format. These are code 6 (PRINT comma), code 13 (ENTER), code 22 (AT control) and code 23 (TAB control). Upon receiving a 'PRINT comma' code, subroutine PCOM is entered. This routine outputs SPACES so that all characters separated by commas are printed in columns 16 characters wide. TAB and AT codes cause flags to be set in location IY+118. The code received after a TAB is then used to move the printer head to the correct column position. AT codes are followed by two parameters (the line and column positions). The first parameter after an AT is therefore ignored. The second parameter moves the print head to the correct column position, as with TAB. The ENTER code sends a carriage return character to the printer, followed by a linefeed (if selected). Carriage return characters will also be sent to the printer if the number of characters already printed out exceed the programmed line length.



### **x=32-127**

All of these codes represent standard ASCII characters, so they are all sent directly to the printer for printing.

### **x=128-164**

These codes represent the User Defined and Block graphics characters in the Spectrum. Since none of the available Centronics printers support this part of the Spectrum character set, a blank SPACE is output to the printer.

### **x=165-255**

These codes all represent BASIC keywords. There is a table containing BASIC keywords in the ROM. Keywords are stored in ASCII format, the last letter of each keyword having bit 7 set to 1. Subroutine RESWRD scans this table up to the relevant keyword. It then outputs the keyword to the printer character by character until the last character with bit 7=1 is reached.

You should now have a general idea about the operation of this software. Individual routines and their operational description can be seen in the assembler listing. The Assembler program printout and the BASIC program listings were all produced on an EPSON MX80F/T dot matrix printer using the Centronics parallel interface described in these pages.

## **USING THE INTERFACE**

Once the hardware for the interface has been constructed and connected to the printer, the new printer program will have to be typed in. Two BASIC program listings are provided for this. One is for 16K Spectrums, the other is for 48K Spectrums. They are written so that the interface machine code is POKEd into the uppermost bytes of available RAM. Once the relevant program has been RUN, the LLIST and LPRINT commands can be used. Other BASIC programs should be LOAded on top of the BASIC interface routine. Do not type NEW, because this will delete the machine code which has been POKEd into the top of memory.

```
00 REM *****
10 REM
20 REM Centronics Parallel Interface Routine
30 REM
40 REM (c)by Adrian Dickens January 1984
50 REM
60 REM *****
70 REM
80 REM This is for a 16K Spectrum
90 REM
100 CLEAR 32333
110 LET a=32334
120 READ x
130 IF x=256 THEN RANDOMIZE USR 32334: STOP
140 POKE a,x
150 LET a=a+1
160 GO TO 220
170 DATA 62,255,211,95,62,253,211,95,62,255,211,127
180 DATA 62,0,211,127,62,4,253,119,118,1,112,126,237
190 DATA 67,197,92,62,70,50,129,92,201
200 DATA 253,203,118,70,32,92,253
210 DATA 203,118,70,194,0,127,254,6,202,9,127
220 DATA 254,22,202,25,127,254,23
```

```

30 DATA 202,30,127,254,165,210,39,127,254,13,40
40 DATA 44,254,32,216,254,128,56,2,62,32
50 DATA 253,52,119,71,58,129,92,253,190,119,48,20
60 DATA 62,13,205,69,127,253,54,119,1,253,203,118
70 DATA 86,40,3,62,10,205,69,127,120,24,15
80 DATA 253,54,119,0,253,203,118,86,40,127,205,69
90 DATA 127,62,10,24,115
00 DATA 253,203,118,134,253,190,119,56,7,253,150,119
10 DATA 71,4,24,25,71,4,253,54,119,255,62,13,205,69
20 DATA 127,253,203,118,86,40,3,62,10,205,69,127,253
30 DATA 52,119,62,32,16,246,201
40 DATA 253,203,118,198,253,203,118,142,201
50 DATA 253,126,119,230,15,200,62,32,205,69,127,253
60 DATA 52,119,24,240
70 DATA 253,203,118,206,201,253,203,118,198,253,203
80 DATA 118,142,201,33,149,0,214,164
90 DATA 203,126,35,40,251,61,254,0,32,246,205,156,126
00 DATA 126,35,205,158,126,203,127,40,247,195,156,126
10 DATA 245,211,63,219,31,203,71,32,250
20 DATA 62,253,211,31,62,255,211,31,241,201,256

```

```

00 REM *****
10 REM
20 REM Centronics Parallel Interface Routine
30 REM
40 REM (c)bv Adrian Dickens January 1984
50 REM
60 REM *****
70 REM
80 REM This is for a 48K Spectrum
90 REM
000 CLEAR A5101
100 LET a=65102
200 READ x
300 IF x=256 THEN RANDOMIZE USR 65102: STOP
400 POKE a,x
500 LET a=a+1
600 GO TO 220
70 DATA 62,255,211,95,62,253,211,95,62,255,211,127
80 DATA 62,0,211,127,62,4,253,119,118,1,112,254,237
90 DATA 62,197,92,62,70,50,129,92,201
000 DATA 253,203,118,70,32,92,253
10 DATA 203,118,78,194,0,255,254,6,202,9,255
20 DATA 254,22,202,75,255,254,73
30 DATA 202,30,255,254,165,210,39,255,254,13,40
40 DATA 44,254,32,216,254,128,56,2,62,32
50 DATA 253,52,119,71,58,129,92,253,190,119,48,20
60 DATA 62,13,205,69,255,253,54,119,1,253,203,118
70 DATA 86,40,3,62,10,205,69,255,120,24,15
80 DATA 253,54,119,0,253,203,118,86,40,122,205,69
90 DATA 255,62,10,24,115
00 DATA 253,203,118,134,253,190,119,56,7,253,150,119
10 DATA 71,4,24,25,71,4,253,54,119,255,62,13,205,69
20 DATA 255,253,203,118,86,40,3,62,10,205,69,255,253
30 DATA 52,119,62,32,16,246,201
40 DATA 253,203,118,198,253,203,118,142,201
50 DATA 253,126,119,230,15,200,62,32,205,69,255,253
60 DATA 52,119,24,240
70 DATA 253,203,118,206,201,253,203,118,198,253,203
80 DATA 118,142,201,33,149,0,214,164
90 DATA 203,126,35,40,251,61,254,0,32,246,205,156,254
00 DATA 126,35,205,158,254,203,127,40,247,195,156,254
10 DATA 245,211,63,219,31,203,71,32,250
20 DATA 62,253,211,31,62,255,211,31,241,201,256

```



## **SOME BASIC NOTES ON USING THE PROGRAM**

The number of characters printed per line on the printer can be changed by POKEing the desired number into location 23681. For example, suppose that a line length of 64 characters per line is required. Simply **POKE 23681,64**.

Some printers turn the paper on by one line when they receive a carriage return (CR) character, but others only move the paper onto the next line when they receive a linefeed character (LF). In order to support both types of printers, bit 2 of location  $1Y+118$  (decimal 23728) determines whether or not linefeeds are provided.

to generate linefeeds, **POKE 23728,4**

to suppress linefeeds, **POKE 23728,0**

Since a printer is likely to be used in lots of applications, the interface program could be stored in an EPROM chip. Such a chip could be connected to the edge connector of the Spectrum. The programs provided in this section will operate from an EPROM, because all variables are stored in the System Variables area in RAM.

## 10. CONNECTING A VIDEO MONITOR TO YOUR SPECTRUM

Most people use a standard colour or monochrome television set with their Spectrum. Good pictures can be obtained on televisions, but coloured characters on coloured backgrounds can sometimes be difficult to see. This is a compromise which must be accepted if a television is used as the computer display device. However, it is possible to obtain a marked improvement in picture quality by connecting a specialised computer display device in place of the television. Such a device is called a Video Monitor.

A Video Monitor looks very similar to a television. It will have a screen, together with some controls to adjust contrast, brightness and the horizontal and vertical position of the display. Notable absences are the speaker and channel tuning controls usually associated with a television. These controls are not necessary on a monitor. The video signal connected to its input is displayed on the screen — there is no possibility of tuning into another signal. Effectively, the video modulator circuit on the Spectrum and the video demodulator circuit on the television have been bypassed by a direct link. A good analogy exists between video and audio signals.

Normally, when a television is being used with the Spectrum, the signal from the Spectrum is just like that transmitted by the television networks. It must be received at the television end, sorted out from all of the other television channels, then displayed. Radios operate in a similar manner. With audio signals, it is much better to connect the signal source (record or tape deck) directly to the amplifier. Transmitting the signal, then receiving it on a radio tends to reduce the sound quality dramatically. Exactly the same happens with video signals.

If you decide to add a video monitor to your Spectrum, there are two basic types — colour or monochrome. Provided that the particular monitor which you choose has a COMPOSITE VIDEO input, it should be suitable.

Connection to the Spectrum is made via the rear edge connector. The composite video output is connected to 15B on the edge connector. A suitable earth connection is pin 14B on the edge connector. Figure 15a shows how to connect the video plug to the rear edge connector. A suitable length of COAXIAL single core cable should be used.

If the Spectrum is an Issue 3, simply plug the video plug into the socket on the monitor, and start to use the Spectrum as if a television were being used. If an Issue 2 Spectrum is being used, a link may have to be made inside the Spectrum.

The position of this link on the circuit board is shown in figure 15b. Any short length of interconnection wire can be used to make the link. Once the link has been made, the monitor can be used as with the Issue 3 Spectrum.



# APPENDIX A — CABLES

Appendix A contains information on the various cables used in the system. It includes a list of the cables and their specifications, and a description of how to connect them to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

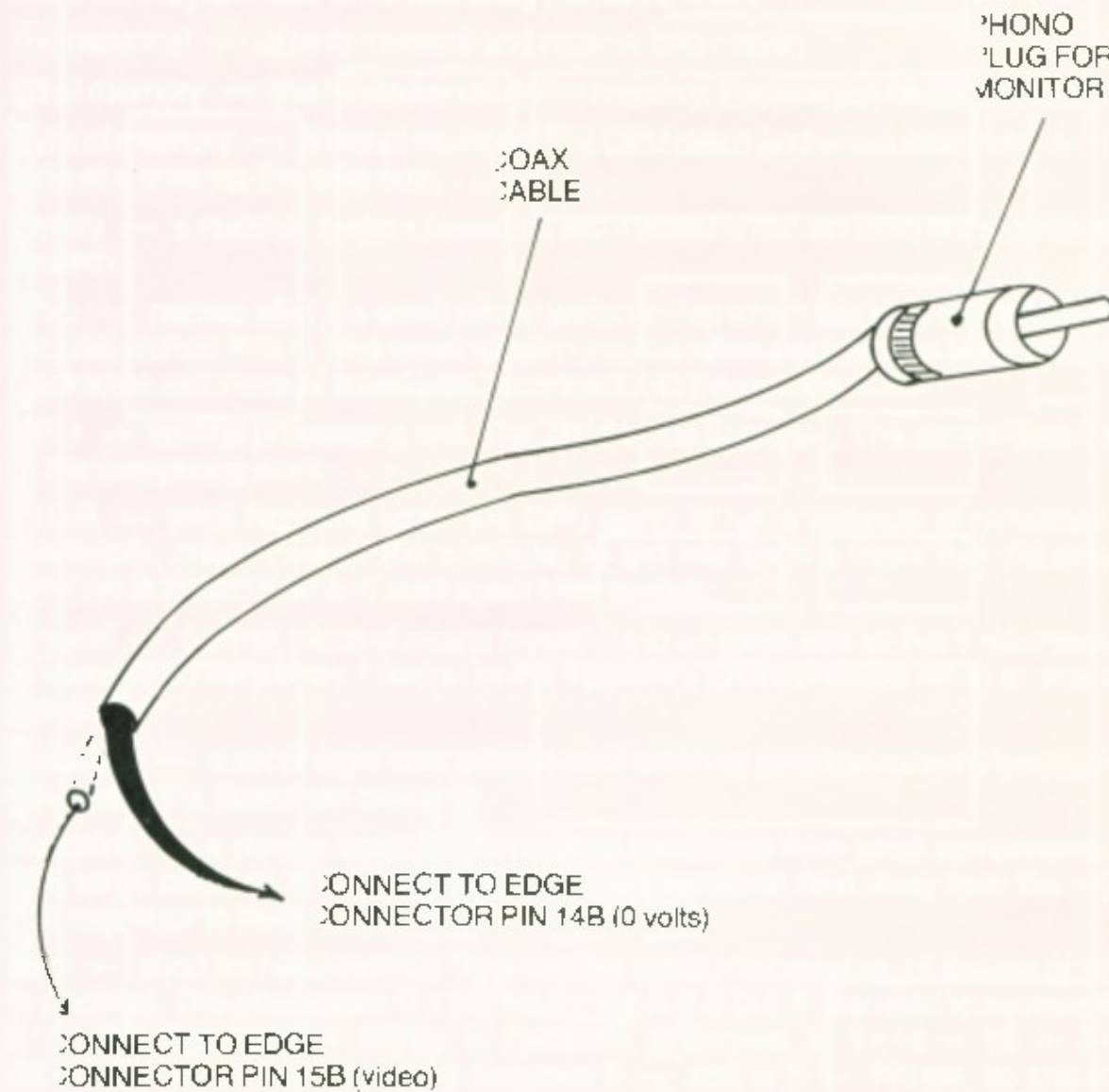
Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.

Appendix A is organized into two sections. The first section, "Cables," lists the cables and their specifications. The second section, "Connections," describes how to connect the cables to the system.



IG. 15a — CONNECTING A VIDEO PLUG TO THE REAR EDGE CONNECTOR

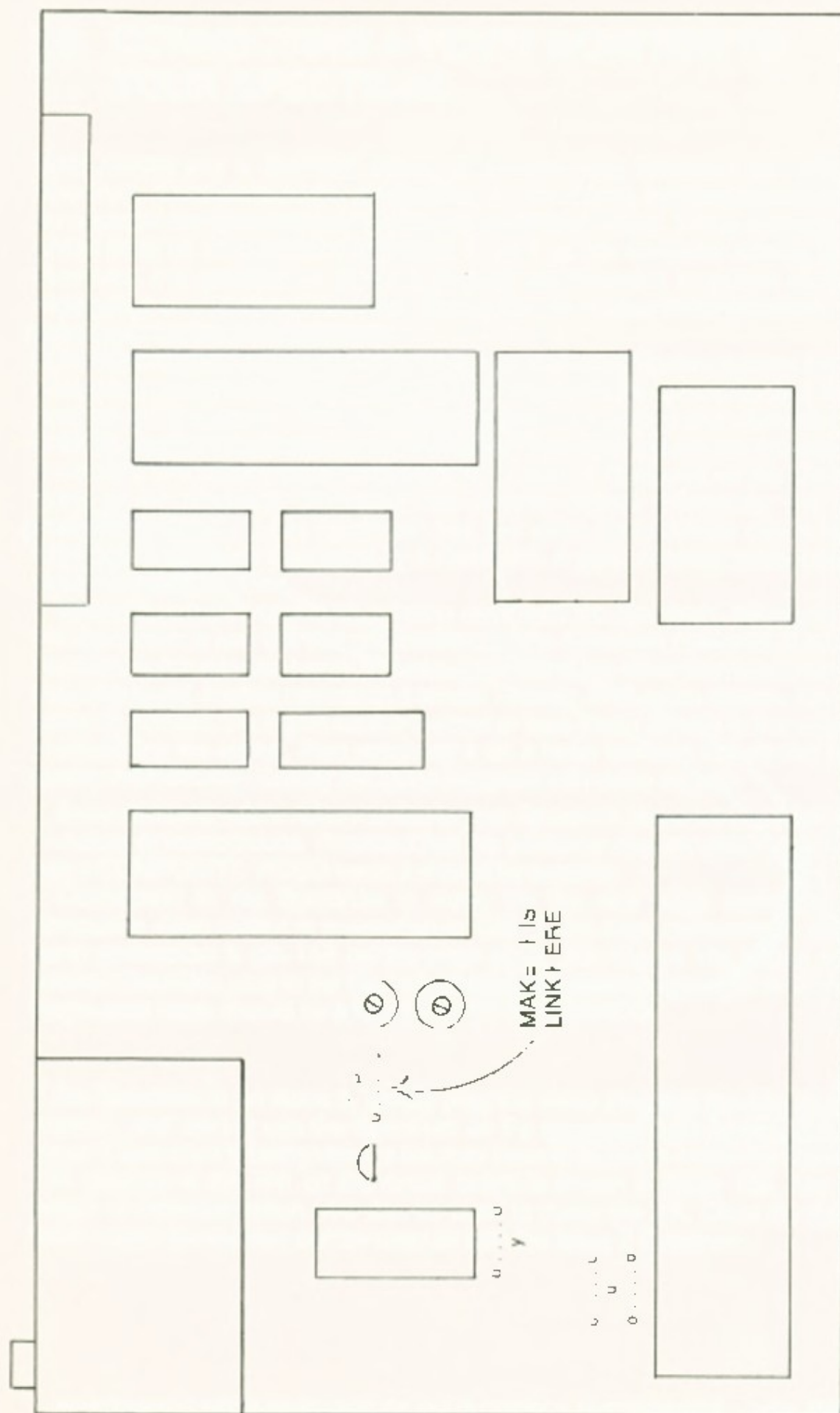


FIG. 15b — VIDEO LINK POSITION ON ISSUE 2 SPECTHUMS ONLY



## APPENDIX A — GLOSSARY

**Address bus** — a set of 16 connections which allow the CPU to select the location at which it wishes to perform a particular operation.

**Active low** — signals which are 'active low' are deemed to be operational when they are at logic 0.

**Analogue to digital converter (ADC)** — a circuit to convert an analogue voltage into digital number which can be read by a computer.

**Asynchronous** — when two devices are operating independently of one another, they are said to be operating asynchronously.

**Bidirectional** — able to transmit and receive information.

**Bit of memory** — the fundamental unit of memory. Each bit can be either 0 or 1.

**Byte of memory** — 8 bits of memory. The CPU reads and writes data 8 bits — 1 byte at a time via the data bus.

**Binary** — a number system based on two instead of the ten used in the decimal system.

**Chip** — named after the small silicon wafer or chip which has all of the computer circuits etched into it. This small silicon circuit is often packaged in black plastic packages with rows of metal pins to connect it to the outside world.

**CPU** — central processing unit. The Z80A in the Spectrum. This device does all of the computing work.

**Crash** — what happens when the CPU starts to run a program of nonsense. Generally, crash causes the computer to get into such a mess that the only solution is to turn it off and start again from scratch.

**Crystal** — a small piece of quartz crystal cut to an exact size so that it resonates at some fixed frequency. Used to fix the frequency at which clocks run to an exact value.

**Data bus** — a set of eight connections over which all data transactions between devices in the Spectrum occur.

**Heatsink** — a device designed to conduct heat away from small components. The large piece of aluminum in the Spectrum conducts heat away from the +5 volt regulator to keep it cool.

**Hex** — short for hexadecimal. This number system is based on 16 instead of 10 in decimal. The characters '0 1 2 3 4 5 6 7 8 9 A B C D E F' are used to represent 0 — 15. The number 1FHex is therefore equivalent to  $1 \times 16 + 15 = 31$  decimal. Hexadecimal is useful because 0 — 255 is 0 — FFHex, so only 2 digits are required to represent any of the possible numbers which can be transferred on the data bus.

**High** — sometimes used to indicate logic level 1

**Interrupt** — a signal produced by external devices to interrupt whatever the CPU is doing and make it do something else.

**Light emitting diode** — a device which will only pass current in one direction. Light is emitted when the device is conducting.

**Low** — sometimes used to signify logic level 0.

Machine code — a program in binary which the CPU can understand directly.

Memory — the devices in which all of the information about BASIC and programs is stored.

MHz — a frequency of 1 million oscillations per second.

Peripheral device — some device connected to the CPU. The keyboard and cassette recorder are both peripheral devices.

Refresh — an operation which must be performed regularly on certain types of memory if they are to retain their stored data.

RAM — random access memory can be read from or written to at any address by the CPU. Two types are commonly used, static which does not need refreshing and dynamic (as used in the Spectrum) which does.

ROM — read only memory, as the name implies can only be read from but not written to. The BASIC operating system in the Spectrum is stored in this type of memory.

State — an input or output can normally only be in one of two states, 0 or 1 (\* but see tristate).

Transducer — a device which converts some physical quantity such as speed, air pressure or temperature into an electrical signal suitable for processing by a computer.

Tristate — sometimes several outputs from different chips can be connected together. So that their data cannot conflict by having a logic 0 and logic 1 state connected together (shorting the power supply through the chips!), all but one of the outputs would be placed in a tristate condition. The tristate outputs can then be either a 1 or 0 and it doesn't matter.

ULA — uncommitted logic array. Mass produced device which can be committed to perform a particular function in the final stages of manufacture.

Vener diode — used to stabilise the voltage across it at some level defined by the diode characteristics.



## APPENDIX B — REFERENCES

1. Sinclair ZX Spectrum BASIC programming manual, Sinclair Research, 1982.
2. Sinclair ZX Spectrum Introduction booklet, Sinclair Research, 1982.
3. Oxford Illustrated Dictionary, Oxford Univ. Press 1962.
4. Mostek Z80 and PIO databooks.
5. Mostek 1980 memory databook and designers guide.
6. Texas Instruments, TTL Databook, 1980.
7. Ferranti Semiconductors, Quick Reference Guide 1981.
8. Zilog Microcomputer Components Data Book, 1980.
9. National Semiconductor, Linear Databook, 1980.
10. National Semiconductor, CMOS Databook, 1981.
11. Ferranti Electronics Ltd., Data Converter Handbook, 1980.
12. J.C. Nichols, E.A. Nichols and P.R. Rony, Z80 Microprocessor Programming & Interfacing Book 2, 1981.

## APPENDIX C — SPECTRUM PARTS LIST

### Resistors

R1 470R  
 R2 470R  
 R3 470R  
 R4 470R  
 R5 470R  
 R6 570R  
 R7 470R  
 R8 470R  
 R9 8K2  
 R10 8K2  
 R11 8K2  
 R12 8K2  
 R13 8K2  
 R14 8K2  
 R15 8K2  
 R16 8K2  
 R17 330R  
 R18 330R  
 R19 330R  
 R20 330R  
 R21 330R  
 R22 330R  
 R23 330R  
 R24 3K3 (or 1K)  
 R25 180R  
 R26 680R  
 R27 680R  
 R28 10K  
 R29 1K5  
 R30 1K  
 R31 220K  
 R32 100R  
 R33 680R  
 R34 15R  
 R35 10K  
 R36 680R  
 R37 1K  
 R38 3K3  
 R39 3K3  
 R40 1K  
 R41 1K5  
 R42 1K  
 R43 3K  
 R44 5K1  
 R45 1K  
 R46 1K

R47 220R  
 R48 4K7 (or 2K2)  
 R49 18K (or 8K2)  
 R50 4K7  
 R51 2K2  
 R52 2K2  
 R53 390R  
 R54 100K  
 R55 56R  
 R56 220R  
 R57 330R  
 R58 1K  
 R59 1K8  
 R60 100R  
 R61 15R  
 R62 15R  
 R63 220R  
 R64 15R  
 R65 10K  
 R66 10K  
 R67 10K  
 R68 10K  
 R69 10K  
 R70 220R  
 R71 220R  
 R72 10K  
 R73 1K  
 R74 10K  
 R75 10K  
 R76 1K  
 R77 1K  
 R78 470R

### Capacitors

C1 47nF  
 C2 47nF  
 C3 47nF  
 C4 47nF  
 C5 47nF  
 C6 47nF  
 C7 47nF  
 C8 47nF  
 C9 — C24 Decoupling Capacitors  
 C25 22uF 16v  
 C26 47 nF  
 C27 1uF 63v  
 C28 22uF 16v



229 47nF  
 230 47nF  
 231 100nF  
 232 100nF  
 233 47nF  
 234 22uF 16v  
 235 10nF  
 236 47nF  
 237 33pF  
 238 33pF  
 239 10nF  
 240 47nF  
 241 47nF  
 242 47nF  
 243 100nF  
 244 100uF 16v  
 245 100uF 16v  
 246 1uF 50v  
 247 22uF 16v  
 248 47nF  
 249 47nF  
 250 22uF 16v  
 251 NOT USED  
 252 100pF (or 150pF)  
 253 100pF (or 150pF)  
 254 470pF  
 255 — C62 Decoupling Capacitors  
 263 47pF  
 264 100pF  
 265 22uF 16v  
 266 47nF  
 267 100pF  
 268 100nF  
 269 100nF  
 270 100nF  
 271 100nF  
 272 16pF  
 273 16pF  
 274 4.7uF 63v  
 275 100nF

#### Transistors

TR1 ZTX 313  
 TR2 ZTX 313  
 TR3 ZTX 313  
 TR4 ZTX 651  
 TR5 ZTX 213  
 TR6 ZTX313  
 TR7 ZTX450  
 TR8 BC184P  
 TR9 BC184P

#### Miscellaneous

7805 1 amp 5 volt regulator  
 X1 14 MHz crystal  
 X2 4.4336 MHz colour crystal  
 VR1 2K2 horizontal preset  
 VR2 2K2 horizontal preset

#### Diodes

D1 — D14 IN4148  
 D15 high power silicon diode  
 D16 Zener diode

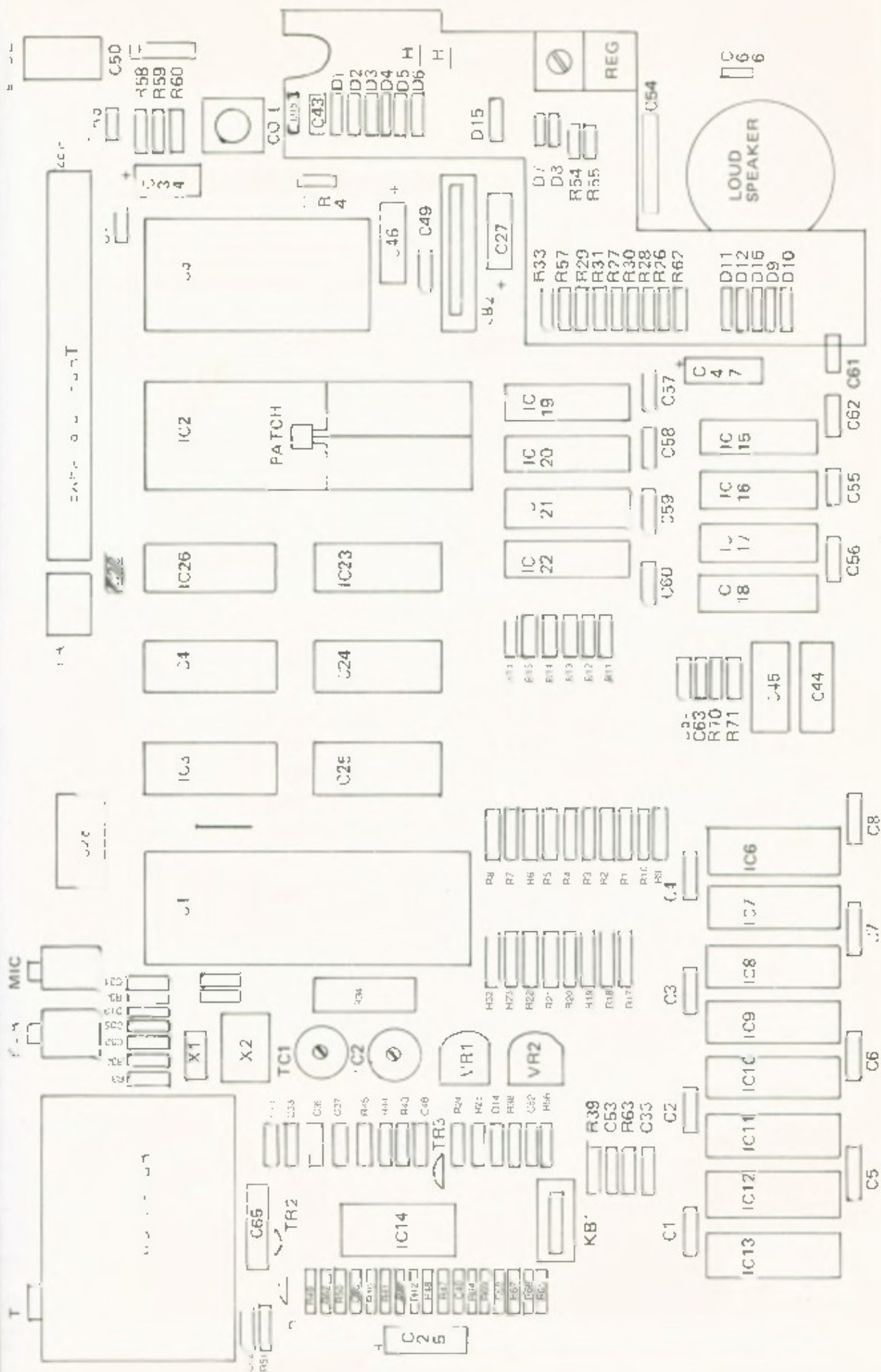
#### Integrated circuits

IC1 Sinclair ULA  
 IC2 Z80A CPU  
 IC3 74LS157  
 IC4 74LS157  
 IC5 16K ROM  
 IC6 — IC13 16K×1 4116 Dynamic RAMS  
 IC14 LM1889N  
 IC15 — IC22 32K×1 4532 Dynamic RAMS  
 IC23 74LS32N  
 IC24 74LS00N  
 IC25 74LS157N  
 IC26 74LS157N

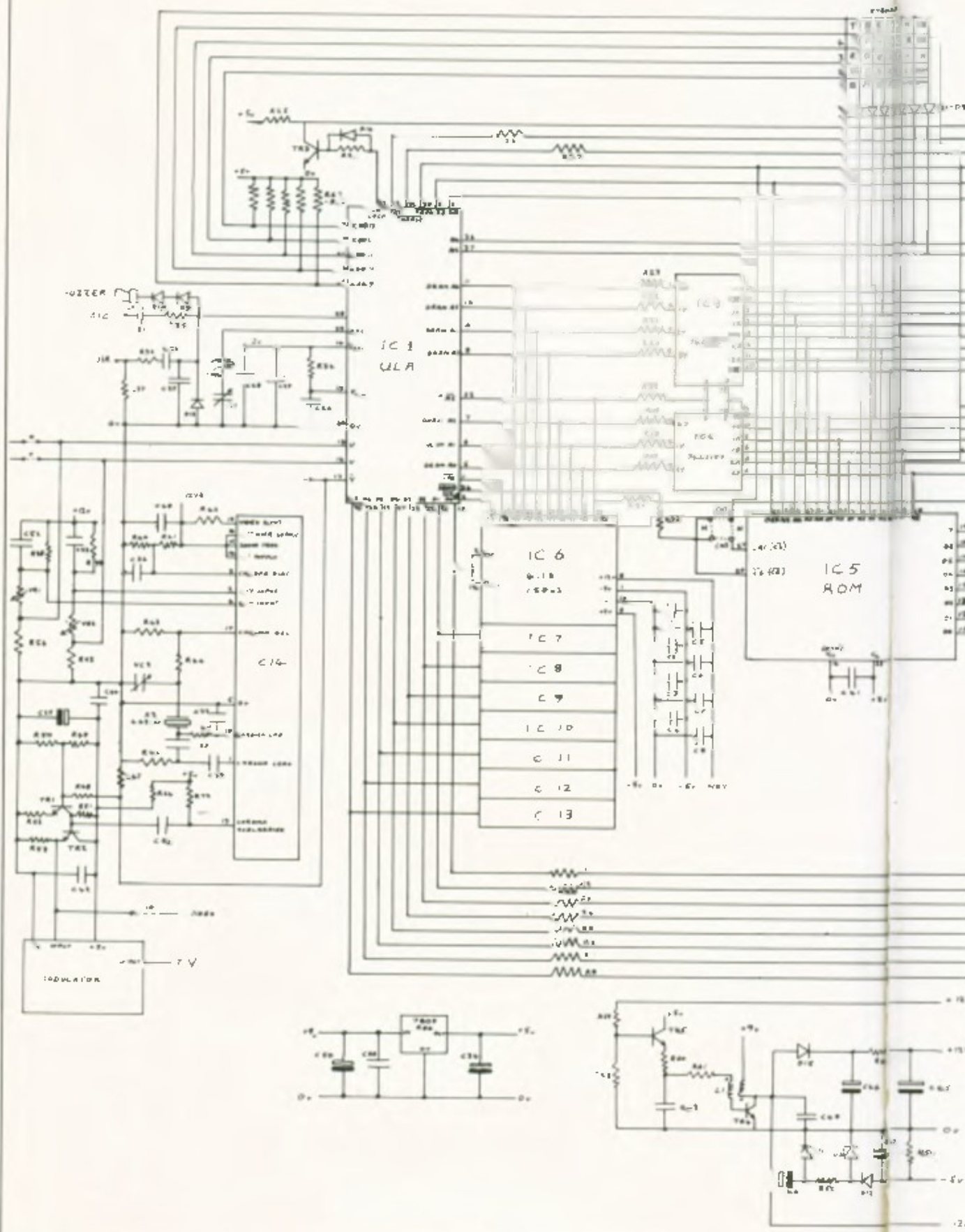
**NOTE:** This is a complete list of parts, as used in Issue 1, Issue 2 and Issue 3 Spectrum computers. Whilst many components are common to all three versions, some components will only be found in one or two of the Issues.



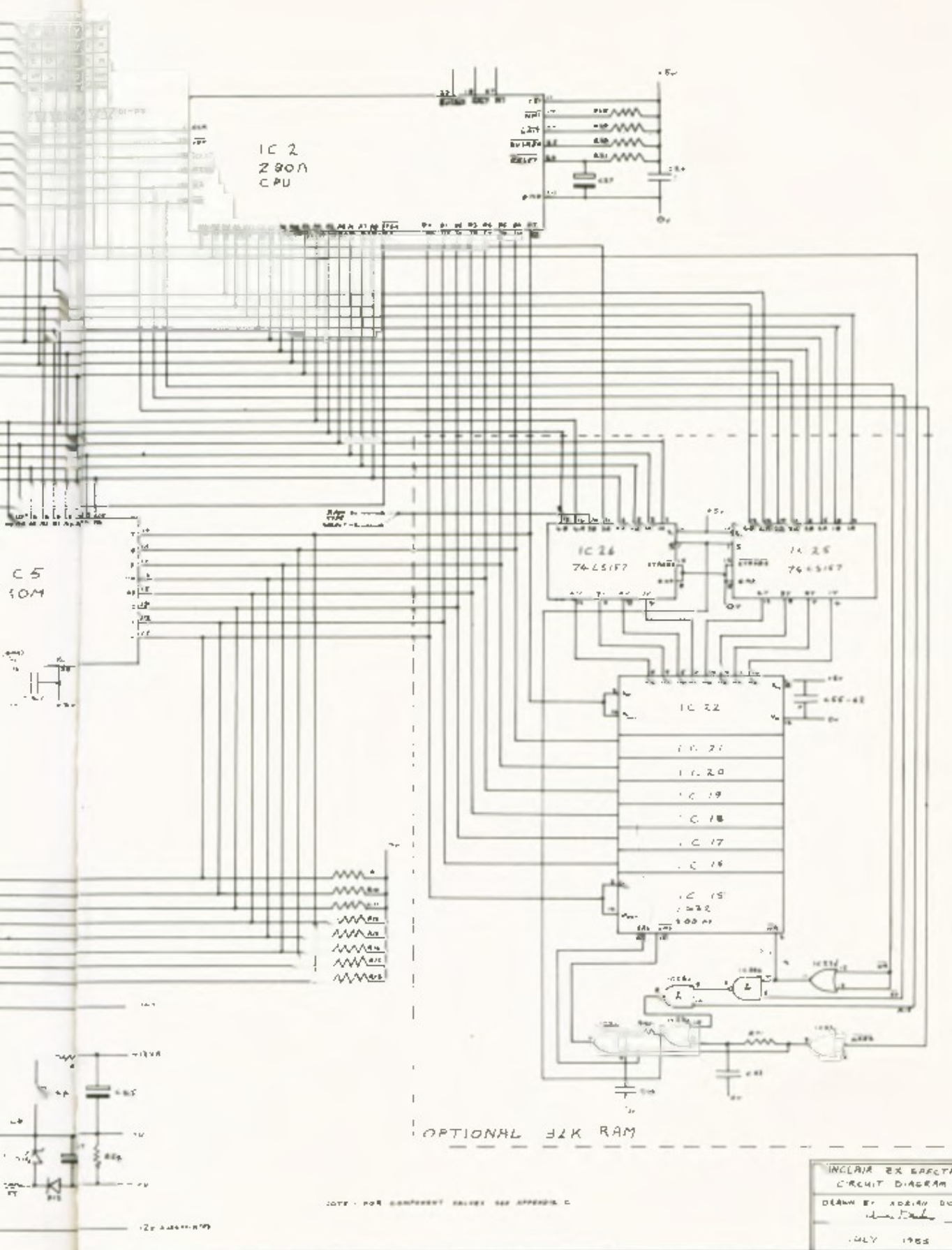




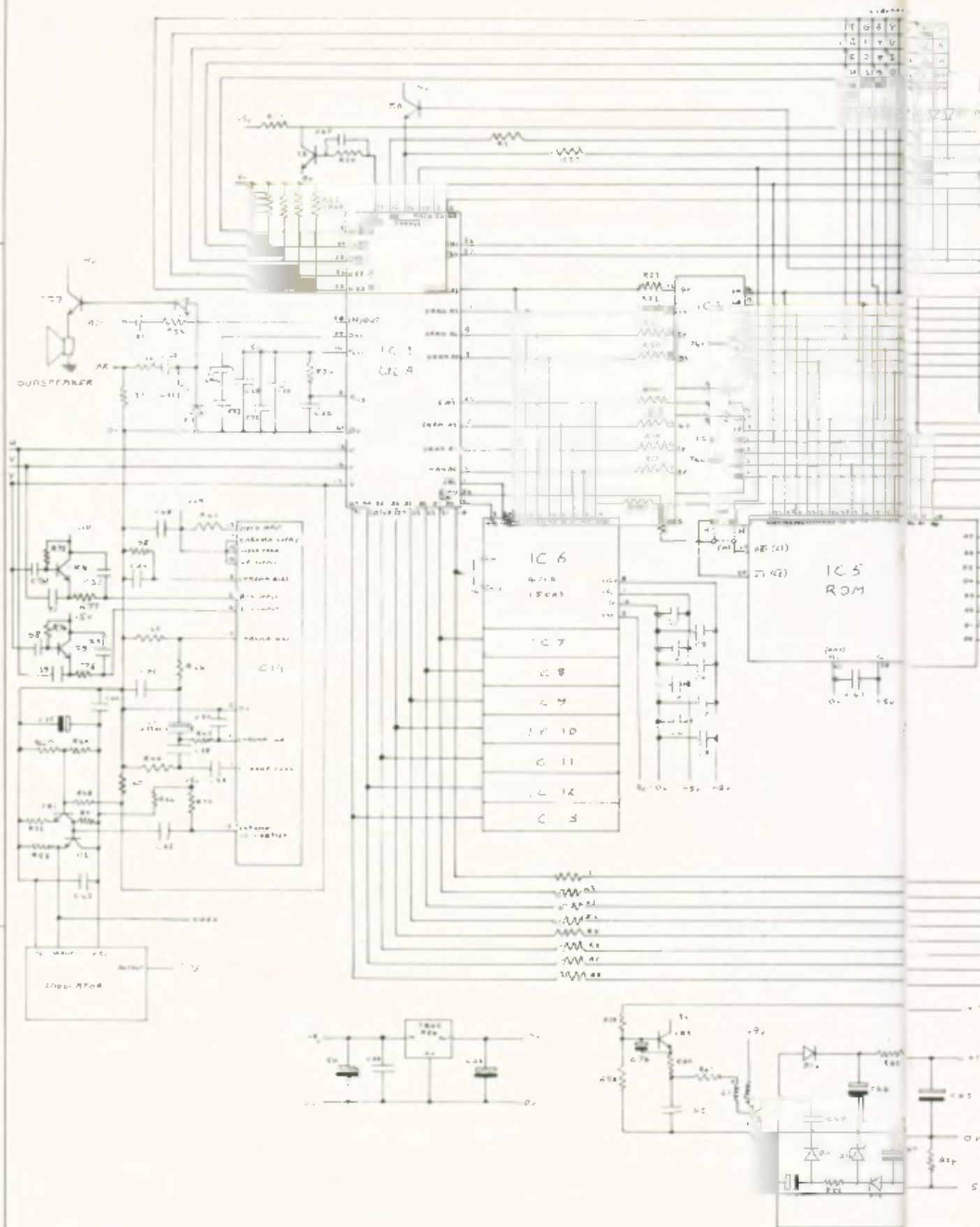
COMPONENT LAYOUT FOR ISSUE 2 BOARD



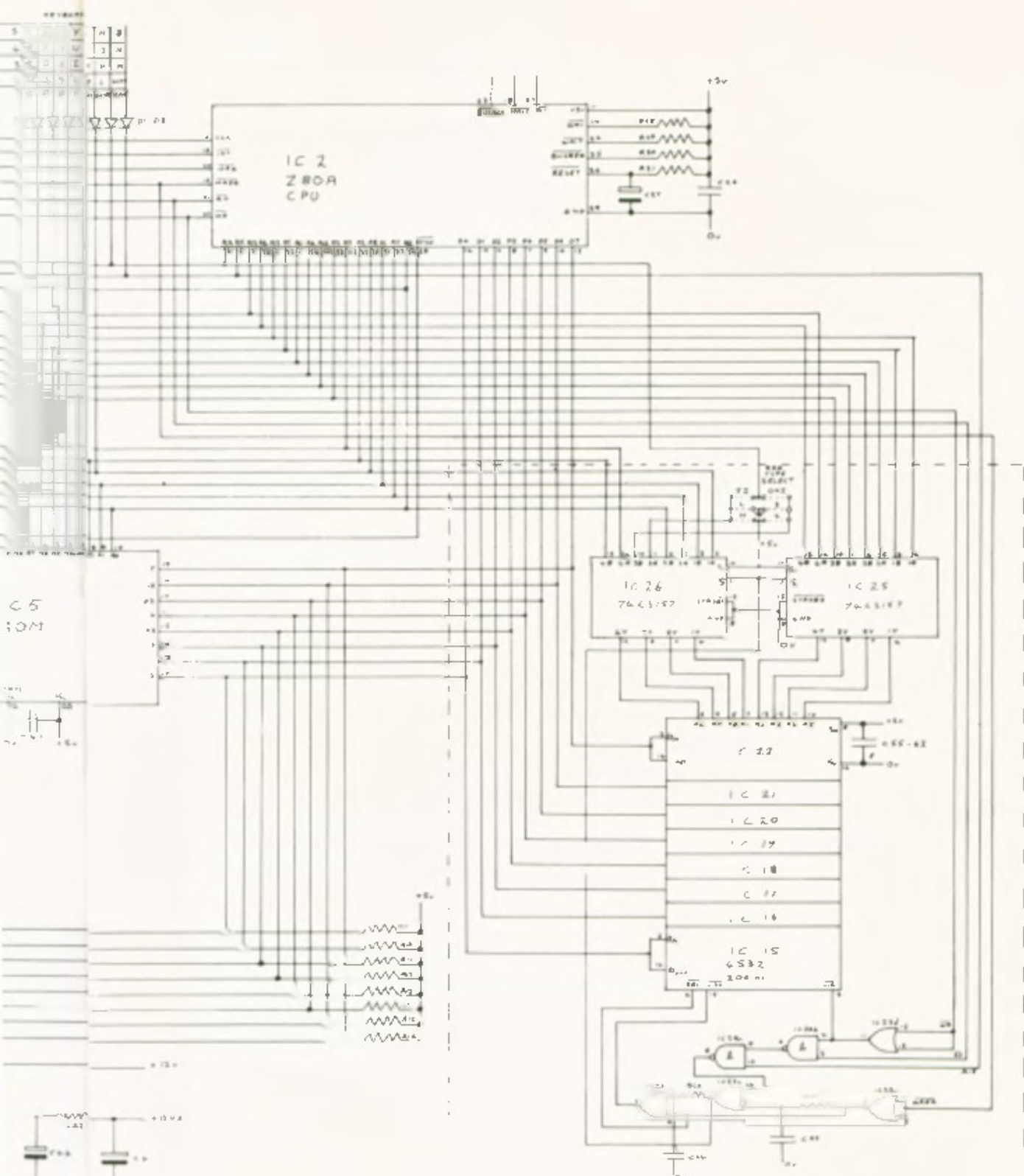




APPENDIX E — ISSUE 2 ZX SPECTRUM CIRCUIT DIAGRAM







OPTIONAL 32K RAM

NOTE: FOR COMPONENT VALUES SEE APPENDIX A

INCLUDE ZX SPECTRUM  
CIRCUIT DIAGRAM ISSUE 3  
DRAWN BY ROGER PICKENS  
JANUARY 1984

# APPENDIX E — ISSUE 3 ZX SPECTRUM CIRCUIT DIAGRAM

# INDEX

## A

ACTIVE LOW .....	23
ADDING YOUR OWN CIRCUITS .....	74
ADDRESS BUS .....	23, 62
ADJUSTMENTS .....	56-60
AMPLIFIER .....	44
ANALOGUE TO DIGITAL CONVERTER .....	90-95

## B

BASIC ROM .....	31
BEEP .....	44
BIDIRECTIONAL .....	109
BINARY .....	9, 109
BIT .....	109
BUSES .....	9
BUZZER .....	43-44
BYTE .....	109

## C

CASSETTE I/O .....	43-44
CENTRONICS .....	96
CHIP .....	109
CLOCK .....	23, 43
COLOUR MONITORS .....	42
COLUMN ADDRESS STROBE (CAS) .....	27
COMPONENTS .....	7, 109
CONTROL BUS .....	11
CPU .....	9-11, 109
CRASH .....	109
CRYSTAL .....	109

## D

DATA BUS .....	24, 109
DECOUPLING CAPACITORS .....	19
DIAGNOSING FAULTS .....	65-66
DRAWING PROGRAM .....	95
DYNAMIC RAM .....	27



## E

EAR SOCKET .....	43-44
DGE CONNECTOR .....	62-64
XPANSION MEMORY .....	47
XPERIMENTS .....	77

## F

FALSE .....	23
FAULTS .....	65-66

## G

GREY SCALE .....	56-60
------------------	-------

## H

HALT .....	24, 71
HEATSINK .....	15, 109
HEXADECIMAL .....	109
HIGH .....	109

## I

IN .....	17
INPUT .....	74
INTERRUPT .....	24, 62, 74, 109

## J

JOYSTICK .....	84-88, 93-95
----------------	--------------

## K

KEYBOARD .....	33-34, 80-82
----------------	--------------

## L

LIGHT EMITTING DIODE (LED) .....	74
LOGIC .....	9
LOW .....	109

## M

MACHINE CODE .....	110
MEMORY .....	110
MICROPROCESSORS .....	23
MONITORS .....	106
MULTIPLEXED .....	27

<b>N</b>	
NON-MASKABLE INTERRUPT (NMI)	26, 62
<b>O</b>	
OPERATING SYSTEM	9
OUT	77
OUTPUT	74
<b>P</b>	
PARALLEL INPUT/OUTPUT CIRCUIT	74-77
PATCH	41
PERIPHERAL	110
POWER SUPPLY	15-19
PRECAUTIONS	7
<b>R</b>	
RANDOM ACCESS MEMORY (RAM)	11, 27, 110
READ ONLY MEMORY (ROM)	11, 31, 110
REFERENCES	111
REFRESH	24, 110
REGULATOR	15
RESET	26, 64
ROW ADDRESS STROBE (RAS)	27
<b>S</b>	
BOUND	43
STATE	110
STATIC RAM	27
SYSTEM	9
<b>T</b>	
TRANSUCER	90, 110
TRISTATE	110
TRUE	23
TUNING	56
<b>U</b>	
UNCOMMITTED LOGIC ARRAY (ULA)	37, 110



**J**

/VIDEO CIRCUIT .....	54
/VIDEO MEMORY .....	27
/VIDEO MONITOR .....	106

**V**

VAIT .....	24
------------	----

**Z**

80 CPU .....	23
80A CPU .....	23
80B CPU .....	23
80A PIO .....	26
ENER DIODE .....	15, 110

PC.W. April 12/14 - '89  
Spectrum +3 to Silver  
Reed

Henry Burek of Barnsley, South  
Yorkshire, writes:

I have a Sinclair ZX Spectrum  
+3 and a Silver Reed EX32  
typewriter/printer. Both are  
equipped with interfaces which are  
Centronics-compatible parallel.  
When the two are connected to-  
gether nothing happens.

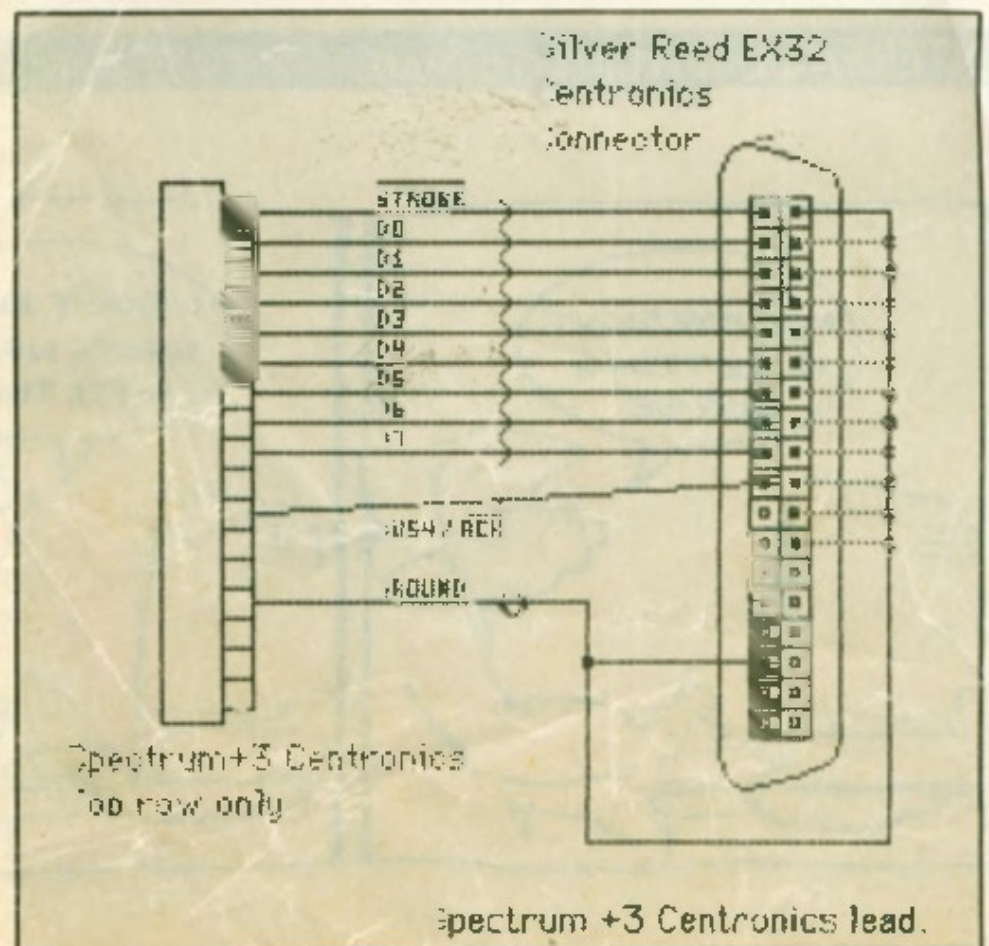
I know the Spectrum +3 and the  
connecting lead are satisfactory as  
they drive another Centronics prin-  
ter - a Seikoshja GP100A - success-  
fully. I know the Silver Reed EX32 is  
satisfactory, as it can be driven by a  
8K Spectrum via a Kempston inter-  
face.

What can I do to make the Silver

Reed and the Spectrum +3 work  
together? They are both supposed to  
have the same interface standard.

As far as I can see, it should  
work. The only possibilities I  
can think of are that either  
the ground lines are connected  
incorrectly or that the BUSY/  
Acknowledge lines are connected  
wrongly. The latter seems the most  
likely and you should make sure  
that the BUSY(11) input of the  
Spectrum +3 is connected to the  
ACK(10) pin of the Centronics  
connector. Apart from that I do  
not know what to suggest. It  
should work.

If you wish, you will be able to  
use the RF (UHF) output to con-  
nect to a TV while watching the  
monochrome picture on the Am-  
strad monitor.





## NOTES

---

# Spectrum Hardware Manual

## Customer Registration Card

Please fill out this page (or a photocopy of it) and return it so that we may keep you informed of new books, software and special offers. Post to the appropriate address on the back.

Date .....19 .....

Name .....

Street & No. ....

City .....Postcode/Zipcode .....

Model of computer owned .....

Where did you learn of this book:

- ☐ FRIEND ☐ RETAIL SHOP  
☐ MAGAZINE (give name) .....  
☐ OTHER (specify) .....

Age? ☐ 10-15 ☐ 16-19 ☐ 20-24 ☐ 25 and over

How would you rate this book?

QUALITY: ☐ Excellent ☐ Good ☐ Poor  
VALUE: ☐ Overpriced ☐ Good ☐ Underpriced

What other books and software would you like to see produced for your computer?





1. Please complete the card and post it to

the appropriate address.

## **United Kingdom**

Melbourne House (Publishers) Ltd  
Castle Yard House  
Castle Yard  
Richmond, TW10 6TF

## **Australia and New Zealand**

Melbourne House (Australia) Pty Ltd  
2nd Floor, 70 Park Street  
South Melbourne, Victoria 3205

# SPECTRUM



Melbourne  
House

This is probably the second most useful book you can buy for your Spectrum — the first being the programming manual that comes with the machine.

The Spectrum Hardware Manual should command a place in your library of reference works about the Spectrum.

'Very useful stuff, indeed.' — Personal Computer News.

Adrian Dickens explains exactly what is inside the Spectrum, and how it works.

Full circuit diagrams and a detailed explanation of each component make it easy to understand the hardware side of this remarkable microcomputer.

Many features not revealed in the Sinclair Manual are discussed here: how to adjust the colours for your own TV set, how to amplify the sound of the internal loudspeaker, and much more.

Practical hardware projects include how to connect a full size keyboard, connecting the Spectrum to the outside world, and how to build your own joysticks for use with the Spectrum. The Spectrum Hardware Manual is a book that will be an essential companion to anyone wishing to discover how the Spectrum operates or wishing to expand its potential.

Now includes information for Spectrum Issue 3.



Melbourne  
House  
Publishers

ISBN 0-86161-115-2



9 780861 611157