£2.25

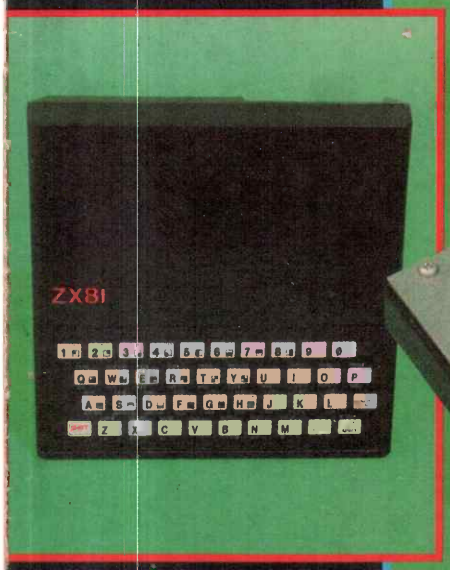
AN ARGUS SPECIALIST PUBLICATION

Vol.5 No.1

# ELECTRONICS DIGEST

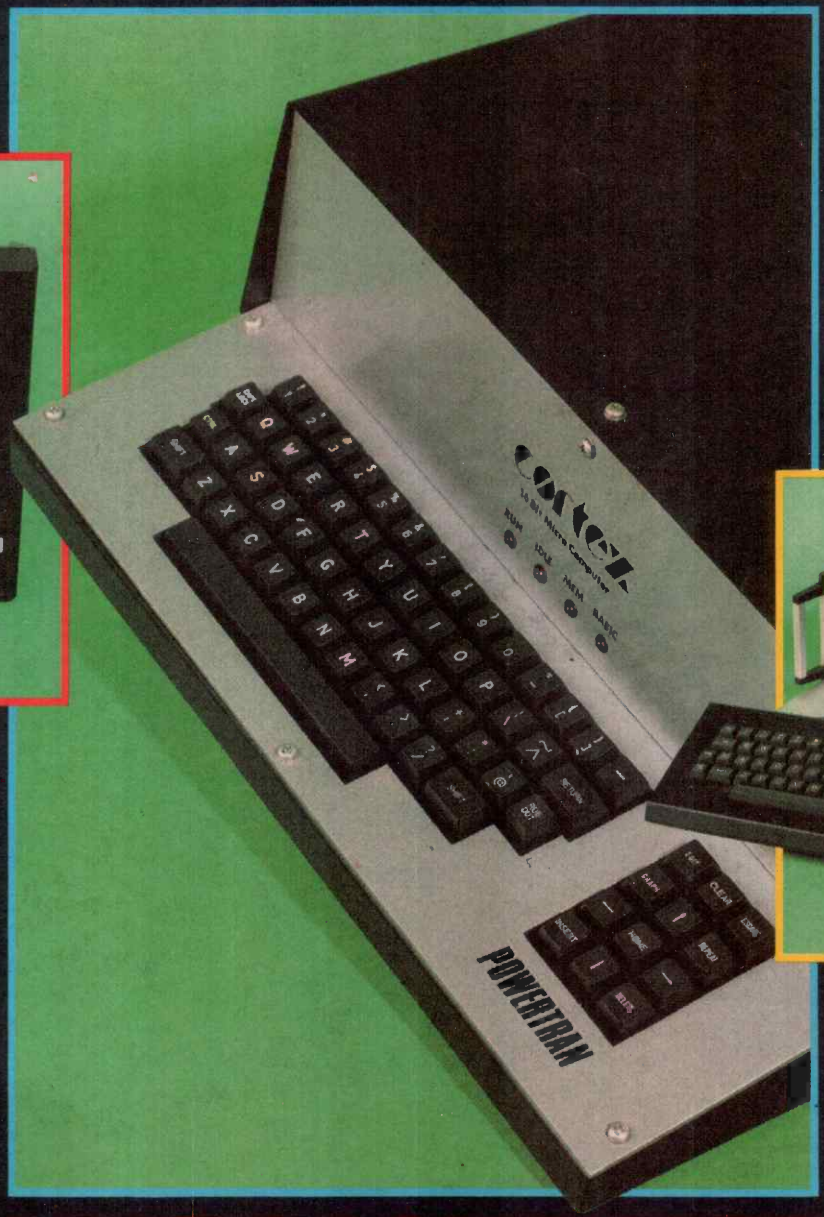


Build your own 16-bit full colour computer




Make more of your ZX81

Control and robotic systems




Boards for the Microtan 65 and other 6502 computers

## MICROCOMPUTER PROJECTS

From the publishers of electronics today

# "Run more than ten tasks on a ZX81-FORTH ROM?"

Sure! More than 10 tasks simultaneously and, in some cases, up to 300 times faster! That's what replacing the basic ROM with the new FORTH does for the ZX81 – and more!

The brains behind the breakthrough belong to David Husband, and he's building Skywave Software on the strength of it. Already orders are flooding in and it's easy to see why.

The ZX81-FORTH ROM gives you a totally new system. In addition to multi-tasking and split screen window capability, you can also edit a program while three or four others are executing, schedule tasks to run from 50 times a second to once a year, and with a further modification switch between FORTH and BASIC whenever you like.

The ZX81-FORTH ROM gives you a normal keyboard with a 64 character buffer and repeat, it supports the 16k, 32k, 64k RAM packs, it is fig-FORTH compatible and it supports the ZX printer.

The price, too, is almost unbelievable. As a "fit it yourself Eprom", complete with manual, it's just £25+VAT.
Add £2 p&p UK (£5 Europe, £10 outside Europe) and send your order to the address below.

**Skywave SOFTWARE**
David Husband
73 Curzon Road, Bournemouth,
BH1 4PW, ENGLAND.
Tel: (0202) 302385.
International +44 202 302385.

---

# cortex SOFTWARE HARDWARE

**MDEX** disc operating system — from £95

**MDEX.** Language power
FORTH PASCAL SPL QBASIC META
Software to make the CORTEX go!

**CORTEX** 1 Mb Disc Drives
80 track double-sided double-density
£235

**TMS9909** disc controller I.C. £24.50

**CORTEX** game tapes
Space Bugs, Nibblers, Pontoon each £6

*Please add VAT to all prices*

**MICRO PROCESSOR ENG LTD**
**21 HANLEY ROAD SHIRLEY**
**SOUTHAMPTON**
**SO1 5AP**
**TEL: 0703 780084**

---

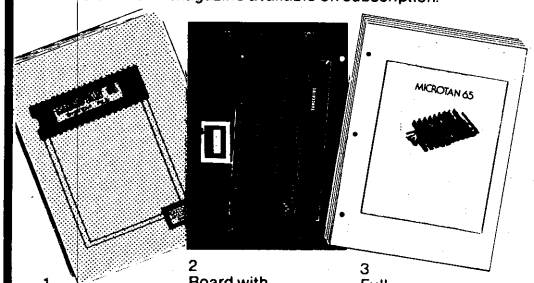
# MICROTANIC COMPUTER SYSTEMS

## MICROTAN 65

***NO OTHER COMPUTER IS AS PERSONAL!***

For less than £60 you can start building your own Computer that truly suits your needs and, of course, eventually far more superior to any Computer available off-the-shelf.

MICROTAN 65 comes in kit form, complete with manual, full instructions, board with components, (kit form or fully built) our full back-up service, and your own Microtan World Magazine available on subscription.




*BUILD AS FAST OR SLOW AS YOU LIKE!*

1 Your Binder
2 Board with components (built or kit form)
3 Full instructions manual

**FLEXIBLE & EXPANDABLE SYSTEM — 1K to 256K!**

*Just look at the options:*

1 DISK CONTROLLER
2 REAL TIME CLOCK
3 EPROM PROG. CARD
4 SOUND BOARD
5 SERIAL 1/0 BOARD
6 PARALLEL 1/0 BOARD
7 ASCII KEYBOARD
8 INDUSTRIAL CONTROLLER BOARD
9 MASS EPROM STORAGE BOARD
10 HIGH RES. GRAPHICS 256x256
11 PRINTER FACE BOARD .
12 40K RAM BOARD


*Microtan World Magazine*

Full range of hardware and software products available

## ACE TRACE

**NEW PRODUCT**

**FOR DRAGON 6809**

***Machine Language Monitor-Disassembler***
***Line Editor & Trace Facilities***

This unique piece of software combines all the tools required to write and debug machine code programmes. It is written in position independent code and hence can reside anywhere in the Dragon's memory map from $0600 to $7FFF. It enables you to TRACE through both RAM and ROM, simulating the 6809 in slow motion displaying the CPU's every move as it happens. Also included is a powerful Monitor Disassembler Line Editor and standard 6809 Assembler supporting all Motorola mnemonics.

CASSETTE BASED **£14.95** Plus 60p P&P

**Microtanic Computer Systems Ltd**
Showroom: 16 Upland Road, Dulwich, London SE22
Tel: 01-693 1137 & 01-299 1419
**DEALER ENQUIRIES WELCOME**

# ELECTRONICS DIGEST

# Volume 5 No. 1

**Editor:**
Dave Bradshaw
**Group Editor:**
Wendy J. Palmer
**Managing Editor:**
Ron Harris
**Advertisement Manager:**
Paul Stanyer
**Chief Executive:**
T. J. Connell

ORIGINATED BY: Tabmag, Northampton.
PRINTED BY: Garden City Press, Letchworth.


Subscription rates upon application to Electronics Digest, Subscriptions Dept., Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP1 1BB.

PUBLISHED BY: Argus Specialist Publications, 1 Golden Square, London, W1R 3AB.

DISTRIBUTED BY: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS (British Isles).



## INTRODUCTION

**Welcome to the world of computing electronics! Once you've decided that there's more to computing than playing games, you enter the real world of the future — and it is a future where computers will be used more and more for an ever increasing variety of tasks.**

**It will be increasingly necessary for the engineer of tomorrow to understand both computers themselves and the way that they can be connected to other systems. We hope that this slim volume will, in a practical way, contribute towards that understanding.**

## CONTENTS



This volume contains reprints from 'Electronics Today International', also published by Argus Specialist Publications. Other work permitting, we are prepared to attempt to answer readers' queries over difficulties in managing to get these projects working. However, we expect readers to make reasonable efforts themselves, such as checking suppliers' advertisements in other publications for sources of components, and making and attempting to interpret diagnostic measurements. We also expect readers to be prudent in their choice of constructional project, bearing their own real capabilities in mind. All readers' queries must be written and accompanied by an s.a.e. — we are not able to answer telephone enquiries, as these seriously disrupt our work. We are not able to advise on modifications. However, we would like to hear of any difficulties with any suppliers mentioned in 'Buylines'.

# CORTEX PART 1

## Announcing the ETI Cortex, in all its 16-bit splendour. This advanced design uses up-to-the-minute technology and forms the basis of a powerful home or business system.

Setting records is getting to be a habit at ETI. We were the first magazine to publish a DIY computer (the Triton). And we were the first magazine to publish a full-feature 16-bit computer, and at a price that makes a lot of commercial machines look a bit sick. The Cortex forms the basis of a versatile computer system that expands with your imagination and is based on state-of-the-art VLSI technology; the 'why use five chips if one will do?' philosophy.

### Processing Power

The processor is a high-speed 16-bit device which possesses a unique system of RAM-based registers. The Cortex kit is supplied with a full 64K bytes of dynamic RAM (ie 32K words of 16 bits), and 24K of BASIC and assembler in an overlaid memory organisation (we'll explain what that means later). The high definition colour VDU has a separate 16K of RAM outside the CPU's 64K memory map and some extraordinary features that result in superb graphics capabilities. Disc drives may be interfaced easily as the controller chips fit on the PCB, and the resident BASIC can be overwritten by disc-based languages; the first that will be available is UCSD Pascal. The latter features will make the system particularly attractive to business users, and Cortexes (Corti?) will be available ready-built as well as in kit form.

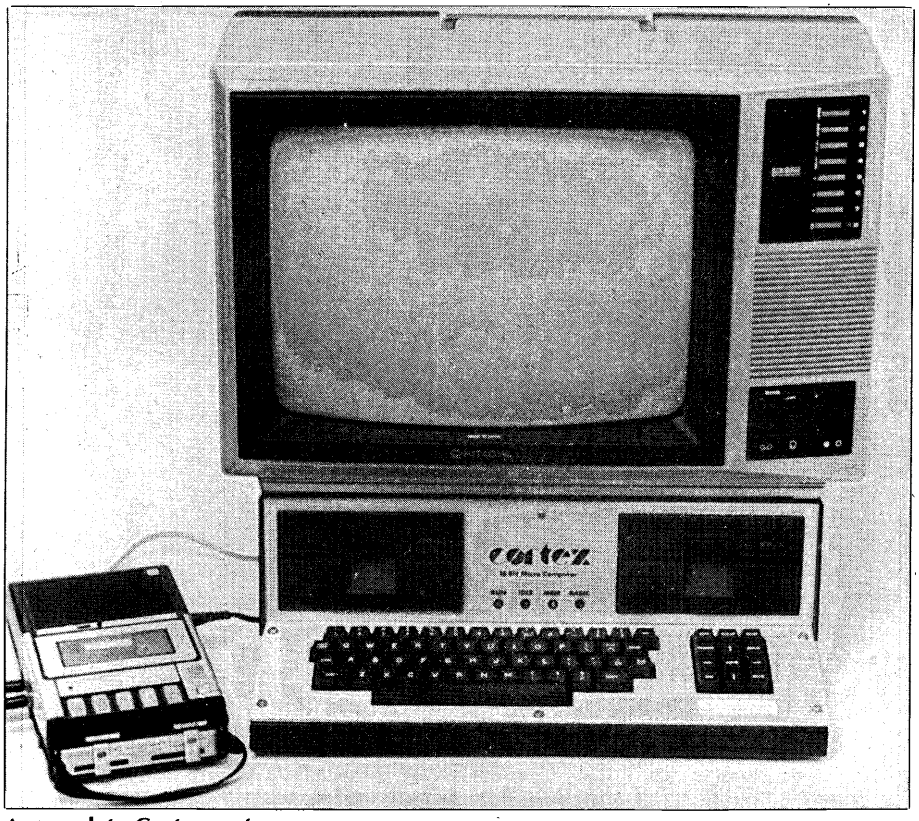
The heart of the Cortex is the TMS9995 CPU. As with all the components in this project it was selected from the wide range of currently available CPUs on a price/performance basis. The 9995 is based on the unique memory-to-memory architecture of the TMS9900. Thus it has the same architectural features of this powerful 16-bit processor and an enhancement of its rich, mini-computer style, instruction set. It is fabricated in state-of-the-art N-channel silicon gate MOS technology, enabling single 5V operation and high speed (12MHz) to be achieved in a compact silicon area.

As well as the 16-bit CPU, fabricated onto the same chip are a number of extra features that make this device the obvious choice for a large number of general purpose applications. 256 bytes (128 words) of on-chip RAM enables four complete, fast access register files (workspaces) to be implemented in full speed memory. A clock generator is also included to minimise the number of external components. Also available are a timer/event counter, a prioritised Interrupt Interface and a 16-bit flag register which can be output on the I/O control bus. The I/O bus is completely separate from the main memory map, and enables 32K of individual I/O bits to be manipulated individually or simultaneously in groups.

### Artful Architecture

The term 'memory-to-memory' implies the fundamental difference between traditional eight-bit CPU architectures and that of the 9995. That is, all transfers in the machine are from one main memory location to another. Only three 16-bit registers exist on the CPU itself; the Program Counter, the Status Register and the Workspace Pointer.

The Workspace Pointer contains the address of the start of a block of RAM anywhere in the main memory map. This address is designated register zero; the next 15 contiguous memory locations are designated Registers 1–15. These registers may then be used by the



**A complete Cortex system.**

programmer as scratch registers. A large number of instructions exist that make maximum use of the reduced addressing needed to access these. For example MPY R4, R5, performs a 16-bit multiply of the contents of Register 4 with Register 5 and stores the 32-bit result in Register 5 (most significant word) and Register 6.

What advantage does this give a programmer over other architectures? Well, in addition to providing no restriction on the choice of addressing modes (register 0 can still be accessed as memory location 7XXXX) the power comes when an interrupt or other subroutine call occurs. It is obviously desirable, especially on receipt of an interrupt, to be able to react as rapidly as possible. On a register-oriented machine it is necessary to save the contents of the working registers in main memory, a slow operation requiring a large number of reads and writes to push the registers onto the stack. On the 9995 the context switch occurs by changing the value in the Workspace Pointer. This points to a new area of fresh registers ready to process the interrupt. The full context of the previous operation is retained in the previous workspace registers, which, being in main memory, are still preserved intact. A return is similarly implemented easily and rapidly by restoring the old Workspace Pointer value. In many real time control situations with a large number of external events occurring this architecture is the only one that allows for efficient processing.

## External Affairs

The system clock is externally generated and fed to the CPU via the XTAL2 input. A clockout signal is also provided that is one-fourth of the crystal frequency (3 MHz). The 64K bytes of system memory are directly addressed by A0-A15. Here the convention is that A0 is the most significant bit. A0-A14 are also used to directly address the separate I/O structure of the Communication Register Unit (CRU). The bit to be accessed is held on A0-A14 and the data is present on A15. The data is then clocked out by CRUCLK. Reading a CRU bit into the CPU is achieved in the same manner but is read into the CRUIN line.

The data bus is multiplexed from the internal 16-bit architecture to eight bits externally, D0-D7. A memory access is signalled by MEMEN and data direction is controlled by DBIN. WE indicates a memory write. For the control of slow memories a READY line signals to the CPU if the memory is ready to complete the current access. If not, the CPU waits for another CLKOUT cycle before continuing.

In the Cortex all these features are used to provide the fastest BASIC available to a home constructor. Even then the speed is still limited by I/O transfers. This can be seen as evidenced by the amount of time the 'idle' LED is on. This LED is directly driven by a status decoder that indicates when the CPU is no longer executing any instructions but is waiting for an external interrupt.

## Dynamic Designing

We don't doubt that all you digital dabblers have watched the plummeting price of memory in our advertisers' pages, and the most dramatic trend has been in dynamic memory (DRAM). This is partly due to the simplicity of the basic cell design in a DRAM; and this simplicity means in turn that the most dramatic cell density increases have been and will be made in DRAMs. With 16K DRAMs prices have pretty much levelled out at the bottom of the curve, but the 64K DRAMs are in hot pursuit.

More importantly from the technical point of view, certain 64K DRAMs offer upward compatibility to the next generation of 256K DRAMs. (Why are the steps quadrupling: 16K to 64K to 256K? Because DRAMs have a two-part

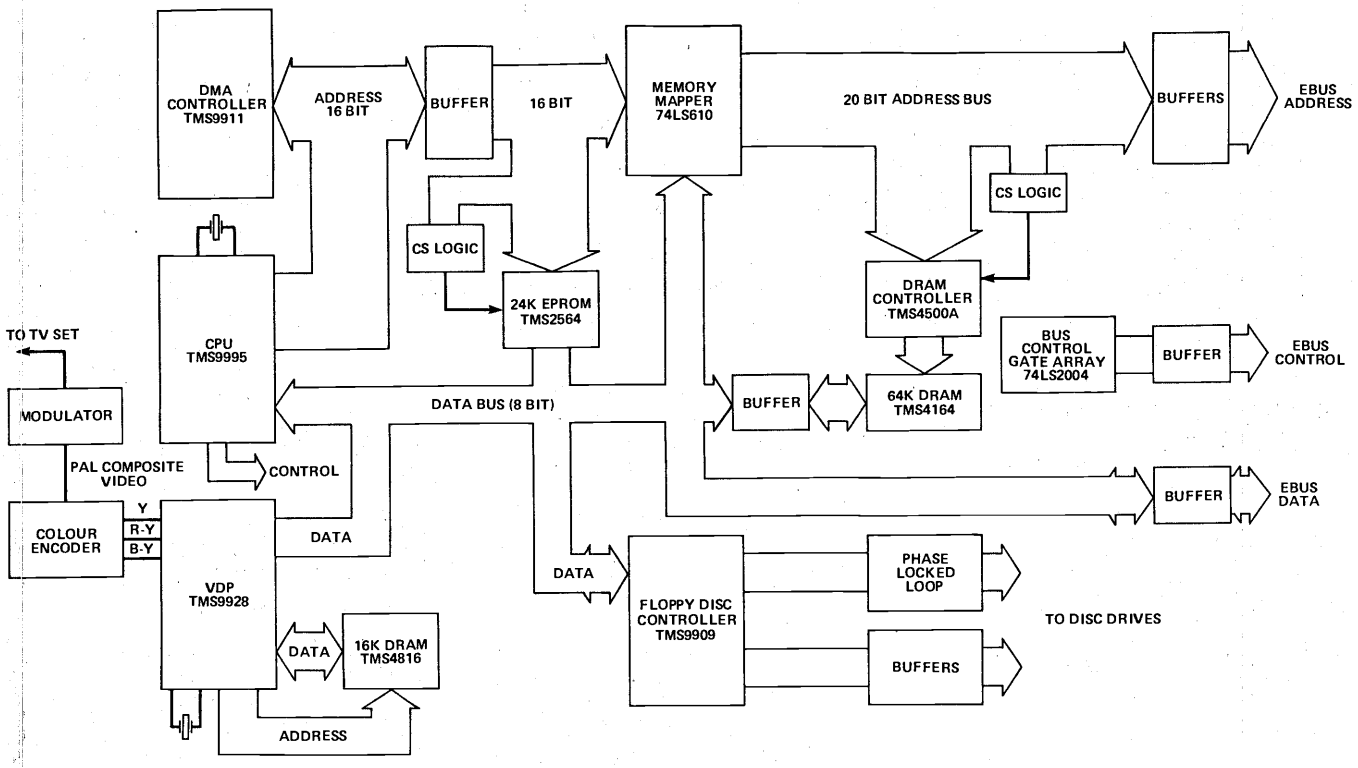
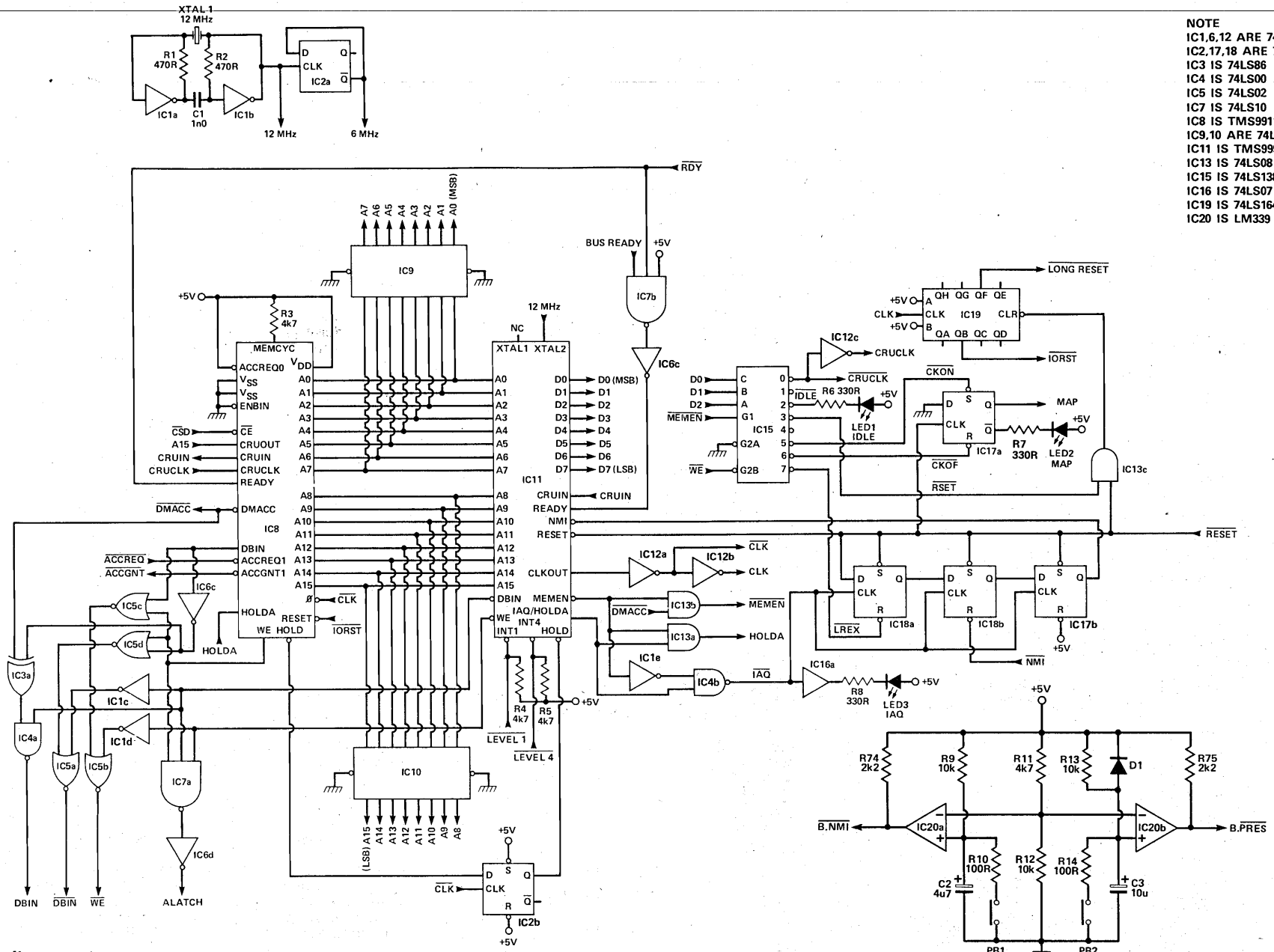



**Fig. 1 Block diagram for the complete Cortex.**




NOTE
IC1,6,12 ARE 74LS04
IC2,17,18 ARE 74LS74
IC3 IS 74LS86
IC4 IS 74LS00
IC5 IS 74LS02
IC7 IS 74LS10
IC8 IS TMS9911
IC9,10 ARE 74LS244
IC11 IS TMS9995
IC13 IS 74LS08
IC15 IS 74LS138
IC16 IS 74LS07
IC19 IS 74LS164
IC20 IS LM339

**Fig. 2 Circuit diagram of the CPU and DMAC circuitry. Note that in this and all the other circuit diagrams no IC pin numbers are shown; we strongly recommend using the special PCB for this project.**

## HOW IT WORKS — CPU AND DMAC

The heart of the system is the CPU, IC11 (a TMS9995). It has a 16-bit internal architecture and an eight-bit wide external data path. The master clock for the system is formed by IC1a,b and associated components; the 12Mhz clock rate of the CPU enables it to complete a memory read or write in only 166nS! This is too fast for present DRAM technology so the automatic wait state feature of the CPU is used. This automatically assumes that memory is not ready and extends the memory access to 500nS. The cycle can be further extended by a low level on the READY input to the CPU; this occurs, for example, when the DRAM is not ready because a refresh cycle is taking place.

The CPU signals the type of memory cycle by driving either DBIN or WE (write) low after driving MEMEN low. If the memory cycle is an instruction fetch then the IAQ/HOLDA signal goes high until both bytes have been fetched. This condition is decoded by IC6e, IC14a and buffered by IC16a to light LED3 to provide a front panel indication.

The CPU has a bit-mapped I/O interface which is separate from the memory data bus; the process is carried out by a section of the CPU called the Communications Register Unit (CRU). The data transfers are serial, bit by bit, each bit having a unique address. This allows 32K bits to be accessed (not 64K since address line A15 carries the data). The value of the data bit is on CRUOUT (A15) for output cycles and a WE/CRUCLK pulse is generated to strobe the data into the I/O devices. On input cycles the data is sampled from the CRUIN line and a pulse is generated on the DBIN line to enable the bus buffers and so on. During all serial I/O operations the MEMEN signal stays high. Any number of bits from one to 16 can be transferred, each bit taking 500nS to transfer if the READY input is high.

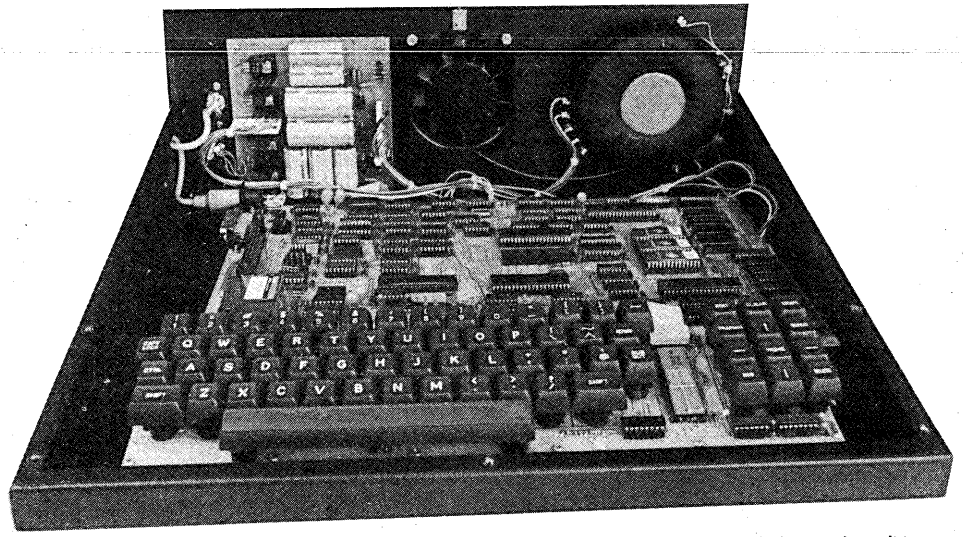
There are some special I/O signals for control use called IDLE, LREX, CKON, CKOF and RSET. IC15 decodes these separately from the normal I/O operations by using the three-bit code output on D0–D2 of the data bus. IDLE pulses continuously whenever the IDLE instruction is executed; it indicates that the CPU is in an internal loop waiting for interrupts (ie doing nothing). LED1 on the front panel lights to indicate this state. LREX is used for the single instruction execution logic which causes an NMI interrupt to occur two instructions after executing the LREX instruction. The two-instruction delay is generated by the series flip-flops IC18a, IC18b and IC2b.

CKON and CKOF are used to switch the memory mapper device from passive to active and vice versa: the signals set or reset the Q output of IC17a to enable or disable the memory mapper (IC26) via IC24a, 25a. When Q is high, Q is low, and LED2 lights to signal that external memory is being accessed.

The RSET signal causes all I/O devices to be reset and sets the CPU interrupt mask to disable all interrupts. Normally both RESET and RSET are high, so the output of IC13c is high and IC19, an eight-bit parallel-out serial shift register, clocks out a continuous series of 1s. A low on RSET or RESET sets all the outputs of IC19 low (specifically IORST and LONG RESET); when the CLR input returns high, 1s are clocked through the shift register first taking IORST then LONG RESET high.

The Direct Memory Access controller (IC8, a TMS9911) is used to provide transparent high speed data transfer to and from the floppy disc controller (FDC) into memory. The address bus of the CPU is tri-state, as are the address outputs of the DMAC. Only one device is in control of the address bus at any one time. When the FDC requires the memory it signals on ACCREQ (access request); the DMAC then signals to the CPU using the HOLD signal that it requires the bus. When the CPU reaches the end of the current memory cycle it tri-states all its outputs (except MEMEN) and signals HOLDA ('acknowledged'). The DMAC now takes over the bus, signals ACCGNT to the FDC, and transfers the data byte between the memory and the FDC.

After completing the memory cycle(s) the DMAC then relinquishes control back to the CPU by releasing HOLD. The CPU then continues as if nothing had happened. The TMS9911 was designed for use with the TMS9900 CPU; when it is used with the TMS9995, extra gating is required to make the signals compatible. Parts of ICs 3, 4, 5, 6 and 7 take care of this. In this application only one of two channels in the DMAC is used; the other is free for the user to experiment with.



multiplexed address bus, so adding one extra address pin is the equivalent of two extra address lines and thus four times the addressing range.) Since all 16K DRAMs operated on a 128-cycle refresh, some 64Ks followed suit. However, that extra address pin called for twice the number of sense amplifiers on the chip, which occupied valuable silicon area (see this month's article on Designing Micro Systems for more about DRAM structure). For 256K DRAMs the waste of chip area is intolerable and to get a product that is capable of manufacture, 256-cycle refresh is essential. The TMS4164 64K DRAM, the first commercially available production device, adopted 256-cycle refresh from the start, as well as following the JEDEC-approved pin-out. This means that the devices are not only upward pin-compatible from the TMS4116, but will also be upward compatible in the future to 256K devices. Simply plug new chips in the old sockets and you've got four times the memory!

For these reasons we chose the TMS4164 to provide a full 64K memory map using only eight chips. Not only is it compact, fast and very reliable, but it is the first 64K device to be successfully encapsulated in a low-cost plastic package.

In our application, refresh is achieved in a manner typical of the Cortex philosophy; a single-chip DRAM refresh controller is used, the TMS4500A. This device provides all the necessary control and arbitration functions for handling 64K DRAMs in a microprocessor system. It accepts the 16 address lines, A0–15, and multiplexes them to row address and column address (RAS and CAS) at the appropriate times, generates refresh signals for 128 or 256-cycle memories and arbitrates synchronously between access requests and refresh cycles. Synchronisation is important for achieving reliable arbitration, and the CPU clock is utilised as the main timing reference.

### ROM To Manoeuvre

Similar design criteria were applied to the choice of EPROMs to store the system firmware. In order to economically store the large number of bytes required to provide

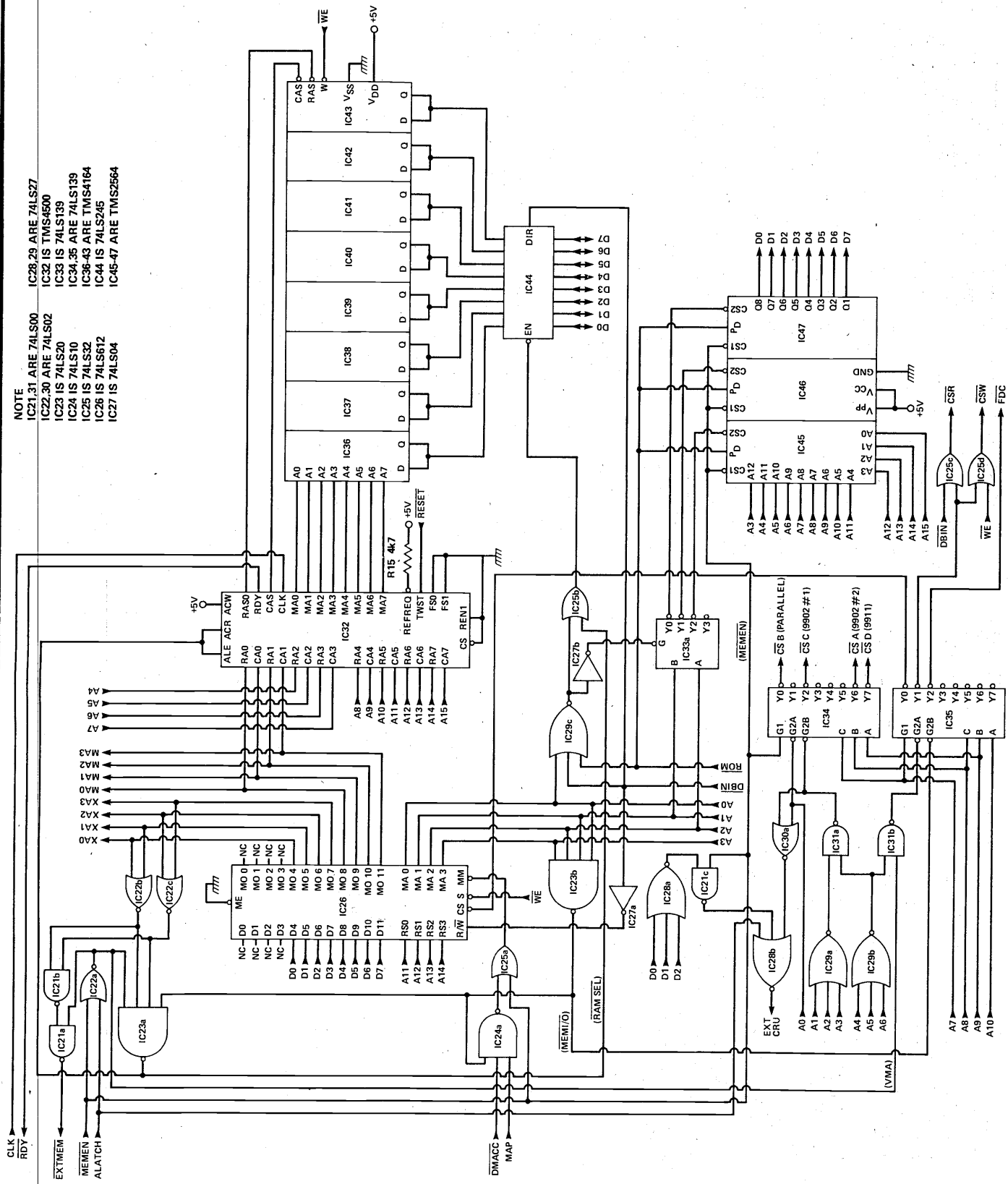



Fig. 3 Circuit diagram of the memory section.

## HOW IT WORKS — MEMORY

**The 24K of EPROM (IC45, 46, 47) contains the assembly language support and the BASIC interpreter. The EPROMs are switched in and out of the memory map by the I/O bit 'ROM' (see I/O section). This signal powers up in the active (low) state, with the EPROMs on. The DRAM (ICs 36–43) is also accessed during a read of the EPROMs but the data buffer IC44 is not enabled; this means that any write while the EPROMs are on is put into the DRAM, so that it behaves as a 'phantom' or ghost. The BASIC interpreter copies itself into the DRAM and then switches the EPROMs off. This has two advantages; first, during execution the interpreter overlays sections of code and then re-copies the relevant section of EPROM back to conserve memory. Second, to enable disc-based operating systems (eg Pascal) to be used, the system needs to be able to operate RAM-resident.**

**The addresses for DRAM accesses are passed through the memory mapper device, IC26. This segments the CPU's 64K address map into 16 pages of 4K; each page has a 12 bit register (MO0 to MO11) of which only eight bits are used (MO4 to MO11). The outputs replace the top four bits from the CPU and add four more. Thus each 4K block can be anywhere in the 1M total address reach (20 bits). The CPU at any one time still only has a 64K address map but by dynamically loading the mapper during program execution the full 1M can be used. The mapper registers are loaded or read as 16 memory locations in the memory-mapped I/O area in high memory.**

**The chip select logic defines the first 64K as on-board memory and the remaining memory map as off-board, using the E-BUS interface (see later). The bottom 32K of memory has the 'phantom' DRAM under the EPROM although only 24K of this is used. The DRAM occupies 60K of the memory map; the top 4K is sub-divided into 256 bytes of high-speed internal to the CPU, and then a memory-mapped I/O region for the Video Display Processor, the memory mapper and the floppy disc controller. Eight memory-mapped slots are decoded, leaving some spare chip selects for user experimentation and expansion.**

**The DRAM controller (IC32) takes the 16-bit least significant address signals and multiplexes them on outputs MA0–7 to supply the memory devices with the correctly-timed waveforms. Once an access request is made, the addresses are first latched and then the controller arbitrates between a refresh cycle and an access cycle. If the controller is busy refreshing the memory then it signals a 'not ready' state to the CPU on the READY line, which suspends operation until the signal returns high again. The controller has to use 'cycle-steal' refresh, as the CPU is nearly always accessing memory and not enough free time can be guaranteed. The refresh cycle obviously slows down the CPU, but by less than 10%, which is a small penalty to pay for the large amount of memory at a low cost.**

a comprehensive and versatile BASIC language, high capacity devices were required. The cost and capacity trends in EPROM technology have followed a similar path to DRAMs; thus the TMS2564 was chosen to complement the TMS4164 in the system. These devices, like all the major devices in the project, operate off a single 5V rail.

Each 2564 can store 8K of program organised in the now industry-standard 8K×8 format. Three chips are used to store the firmware: 24K in all. However, an important design feature of the Cortex should be mentioned here; the EPROMs phantom the DRAMs in the memory map. At power-up the EPROMs are enabled, and after checking the DRAMs the program then copies the full operating system from EPROM into RAM. Once this has been completed the EPROMs are disabled. Thus the operating system is running in RAM, allowing changes to be made or sections deleted to create extra space. This is most noticeable when you're only operating the assembler section, since the whole of the BASIC interpreter may then be eliminated from memory, freeing an extra 20K for the user! Thus the Cortex is the only computer offering a full 64K of RAM to the user as a standard feature!

### Graphic Descriptions

No home computer is considered worthy of the name these days unless it features impressive graphics capabilities, and the Cortex certainly won't disappoint you in this respect (just look at our cover and the photos in this article for some examples of the display power). What's more, all this is controlled with just one IC, the TMS9929! This chip is a memory-mapped video controller, which produces all the relevant video signals internally; it's a very versatile device capable of producing extremely complex displays, including 3D simulations. In order to achieve this its mode of operation is very different to the majority of display devices currently in use.

For simple text displays the video output of a computer may be produced by using a small random access memory to address a larger read-only memory that stores the character shape. Teletext displays fall into this category. The next step up from this is to have a display where each displayed dot is a bit (or bit pattern) stored in RAM. The first method means that shapes can be positioned very rapidly but the repertoire of shapes is limited to those in the graphics/text ROM. The second scheme is slower but allows more complex shapes to be created and lines to be plotted. However, problems occur when trying to overlay shapes as everything must be done in software — making things even slower.

The 9929 uses variants on both these schemes to produce extremely complex shapes with the minimum of software overhead. Tables are designated in an area of RAM to define pattern shapes, characters or graphics. A separate table area is designated to define attributes to the shape, such as colour and size. Finally an area of RAM akin to the Teletext RAM contains pointers for each screen location that point to the desired shape and its associated attributes. The advantage of this system is that large, dramatic changes can be made to the display very rapidly by making simple changes in the pattern look-up table. Further, all displays of one particular shape can be modified simultaneously by updating the pattern generator table.

A distinction is made in hardware between graphics and text modes, though text is available in graphics mode. They may be interchanged using the BASIC commands TEXT and GRAPH. In GRAPH mode the resolution is 256×192 (48K pixels), and the colour of each pixel can be set to any of the 16 available display colours, the only limitation being that only two colours per eight pixels may be used horizontally. To put it another way, the background may be any one of the 16 colours, while the colours of the pixels (foreground) can be set to one of 16 colours in blocks of 32×192. Control is exercised from BASIC using the COLOUR and PLOT commands.

In TEXT mode the screen format is 24 rows of 40 characters per row, each character being defined by a 6×8 bit matrix. In this case an attribute table is not required; instead the character colour is defined by a colour register on the 9929 itself that stores one foreground and one background colour. The character shapes are held in RAM in the VDP's memory

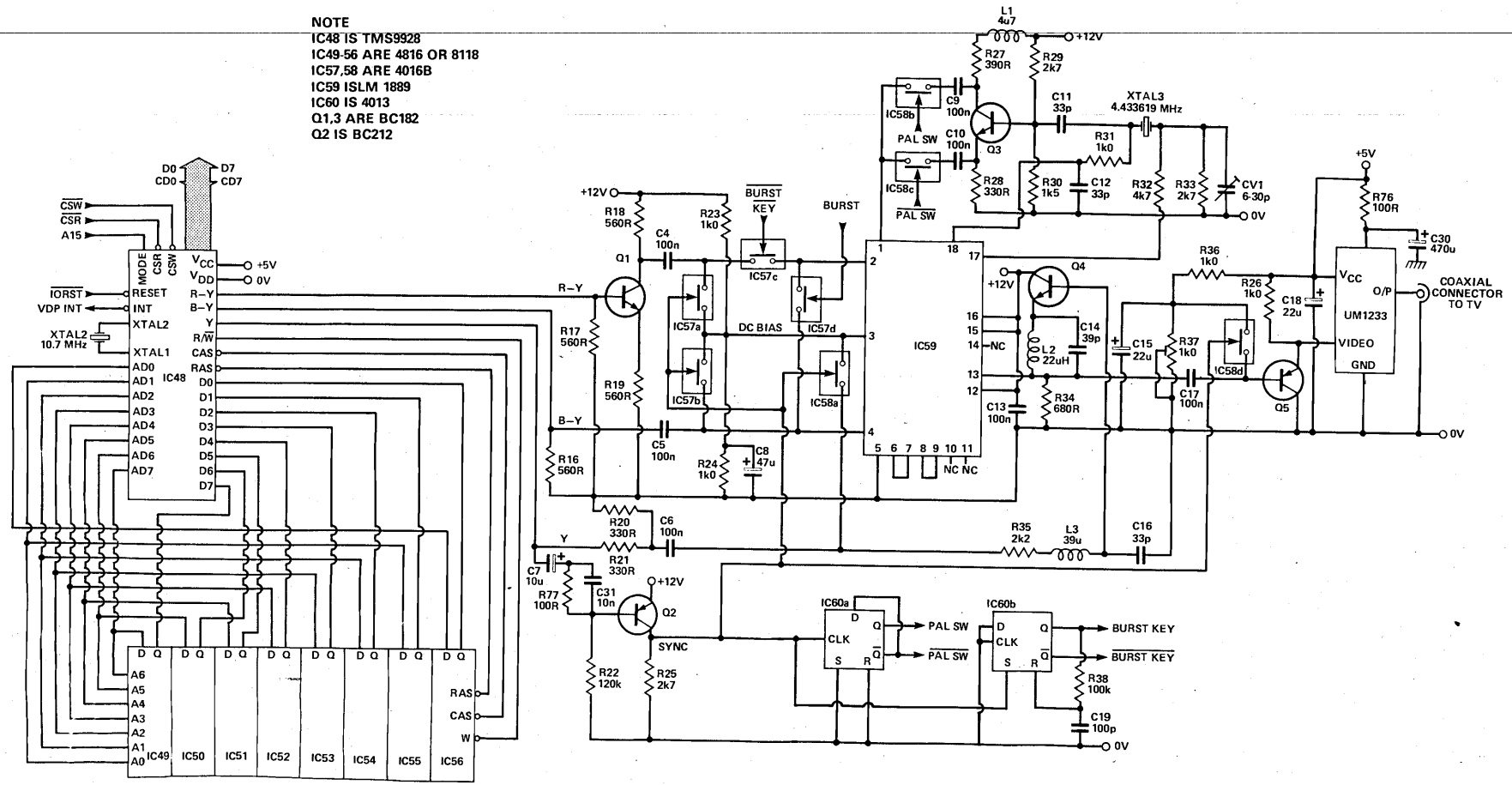



Fig. 4 Circuit diagram of the video display circuitry.

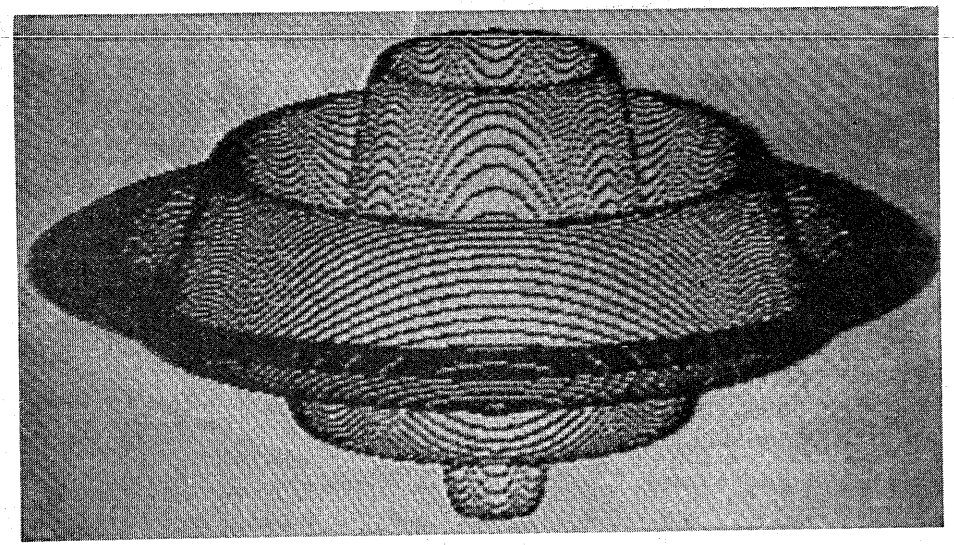
## HOW IT WORKS — VIDEO DISPLAY PROCESSOR AND PAL ENCODER

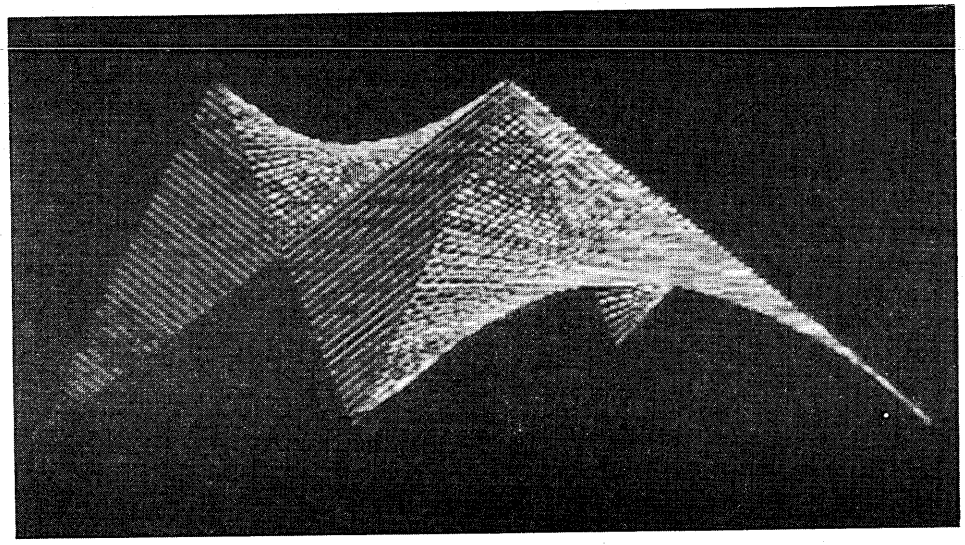
The TMS9929 (IC48) is a 625-line non-interlaced video display processor; it directly drives 16K of memory which is completely separate from the main CPU memory. The VDP fetches data from its DRAM (ICs 49–56) at such a rate that the DRAM is automatically refreshed many times over. There's very little else to say about this section of the circuitry — IC48 does everything internally! The VDP outputs a composite luminance and sync waveform on the Y output and colour difference signals on the R–Y and B–Y outputs. These signals contain all the information needed to produce either R–G–B or PAL-encoded colour. The VDP is controlled by a two-byte memory-mapped slot in high memory.

The sync is separated from the Y (luminance) signal by Q2 and associated circuitry, and used to drive DC restoration clamps IC57a,b,58a; these charge capacitors C4, 5, 6 to a reference voltage during the sync pulse. The sync is also used to toggle the PAL (Phase Alternate Line) switch, IC60a, which gates an inverted or non-inverted chroma oscillator signal into one of the two analogue multipliers in IC59. The chroma oscillator is built around Q3 and XTAL3, a crystal whose resonant frequency is 4.433619 MHz, that of the PAL colour subcarrier. The non-inverted signal is taken from the collector of Q3 via C9 and IC58b, while the inverted signal is taken from Q3 emitter via C10 and IC58c.

The second flip-flop, IC60b, is used as a monostable which, at the beginning of each line, connects the burst pulse which occurs on the B–Y signal to the R–Y input of IC59. This switching is done by IC57d. The inverting amplifier Q1 on the R–Y line from the VDP, IC48, is to match the direction of the burst pulse with the direction of the video signal to yield the correct colours.

The luminance signal is low-pass-filtered by R28, L3, C16 and then summed with the chrominance output of IC59 via the chroma trap L2, C14; this filter removes colour fringing effects. The signal is then DC-shifted by another DC restoration clamp (IC58d) to feed the RF modulator.





and so may be easily user-modified. The CHAR command allows any of the 256 possible character definitions to be altered.

Table 1 shows the 16 colours which are available; this 'palette' has been arranged to give not only a good colour display, but also a good monochrome display, as the colours produce an even grey scale on a black-and-white TV. Eight grey levels are generated.

One peculiarity may have caught your eye in Table 1: what is the point of a transparent colour? A transparent object will allow you to see what's behind it, but in most graphic displays 'behind' is meaningless. However, the VDP in the Cortex considers its display to consist of 36 planes prioritised one above the other. When you look at the screen you're seeing an image which can be considered analogous to holding 36 colour slides, one above the other in a stack, and peering through them all.

**TABLE 1**

| Code | Colour | Code | Colour |
|---|---|---|---|
| 0 | transparent | 8 | medium red |
| 1 | black | 9 | light red |
| 2 | medium green | 10 | dark yellow |
| 3 | light green | 11 | light yellow |
| 4 | dark blue | 12 | dark green |
| 5 | light blue | 13 | magenta |
| 6 | dark red | 14 | grey |
| 7 | cyan | 15 | white |

The rearmost plane is black to allow images to be built up over it. The next plane is for external video and need not concern us here. On top of this is the backdrop plane which lies directly behind the text/graphic plane. This defines the border colour as well. Since this plane defines the colour of the whole screen, it is now obvious that the only way to see the external video input or the black rearmost plane is to set the backdrop to transparent. The text/graphic plane is written to by the TEXT and GRAPH commands discussed earlier.

This leaves another 32 planes sitting in front of the four mentioned above; these are called the sprite planes. A sprite is a graphic shape that can be user-defined from BASIC with the SPRITE command. Sprites can be displayed in a variety of sizes depending on the size and magnification flags; these give four possible modes. SIZE 0 means that a block of 8×8 bits is used to define the sprite, while SIZE 1 uses 16×16 bits (but reduces the total number of different shapes from 256 to 64). The display size can be varied with the MAG command; MAG 0 maps one bit in the shape onto one pixel while MAG 1 maps one bit onto a 2×2 block of pixels on the screen.

Each sprite has four attributes associated with it; its plane (or priority), its colour and its X and Y screen coordinates. Again, each sprite can be one of 16 colours, and those bits set to 1 in the sprite definition adopt the defined colour while the other bits are set to transparent. The screen coordinates define the position of the top left-hand corner of the sprite, and sprite positions can therefore be rapidly changed by simply altering two bytes in memory. The colour can be changed equally quickly by altering one byte. Because the planes are prioritised, 0 to 31, if any shape is positioned coincident with another shape, only the one with the highest priority plane will be displayed. This gives rise to simple 3D simulation. A status flag is set to indicate when any two sprites 'touch' each other. Any point in the text or graphic plane will only be seen if all the points directly above it on the 32 sprite planes are transparent.

All these features mean you can generate very versatile and complex displays; but they also use up a reasonable amount of memory. We don't believe that screen RAM should be stolen from the user RAM, so the VDP only occupies two bytes in the CPU memory map. These two registers are all that's required to write all the relevant information through the VDP chip and into its own 16K of DRAM.

# CORTEX PART 2

## Build yourself a better brain: in this article we explain the remaining Cortex circuitry and the construction of the main board.

Serial I/O on the Cortex is handled by a versatile UART, the 9902. The CPU communicates with the UART via its serial I/O bus, based on the Communication Register Unit or CRU, which requires only three wires; thus the device fits easily into an 18-pin package. The 9902 is fully programmable and the range of variations is so great that it's outside the scope of this article. In the Cortex the chip is configured to handle RS232 eight-bit codes with even parity and 1½ stop bits; the communication rate can be set from BASIC using the BAUD command and the device is activated using the UNIT statement. The parameter for UNIT is a 16-bit word, each bit corresponding to a channel that can be either on (1) or off (0).

Channel 0 is the keyboard/screen channel; channel 1 is the 9902 that is already wired into the PCB. Channels 2–15 are implemented in software and only require the addition of extra 9902s on the CRU bus. The Cortex powers up set to UNIT 1. Executing UNIT 2 disables the keyboard and passes control to the 9902. UNIT 3 enables both simultaneously.

**TABLE 1**

| SIZE | DDEN | TRANSFER RATE (kHz) | DIVISION RATIO (IC87) | MONOSTABLE PERIOD (uS) | COMMENTS |
|---|---|---|---|---|---|
| 0 | 0 | 125 | 12 | 3.0 | 5¼" single density |
| 1 | 0 | 250 | 6 | 1.5 | 5¼" double density |
| 0 | 1 | 250 | 6 | 1.5 | 8" single density |
| 1 | 1 | 500 | 3 | 0.75 | 8" double density |

### Construction

The main board and the keyboard both have plated-through PCBs, ie there are tracks on both sides and connections between the sides are made by the copper that has been plated onto the sides of each hole. There are therefore no track-link pins; it is, however, good practice to apply solder to EVERY hole to reinforce the connections which in some cases carry power. This happens automatically when boards are 'flow soldered' by

### HOW IT WORKS—I/O

**The I/O map space is split into two regions; the bottom region is for on-board I/O devices and the top region causes an off-board access. (The CPU has an internal I/O area of 16 bits, some of which is reserved for specific hardware functions; the rest is free for the user.) The on-board I/O area of the Cortex is decoded by IC34 into eight 32-bit slots, of which only four are used. Two slots (CS A and CS C) are used for the Asynchronous Communications Controllers (ACCs), the third (CS B) for the parallel I/O for the keyboard data, flags and control lines (such as 'ROM', mentioned in the Memory section), and the fourth for the DMA controller IC8 (CS D).**

NOTE
IC16 IS 74LS07
IC61 IS 74LS74
IC62,63 ARE 74LS251
IC64 IS 74LS259
IC65 IS 74LS32

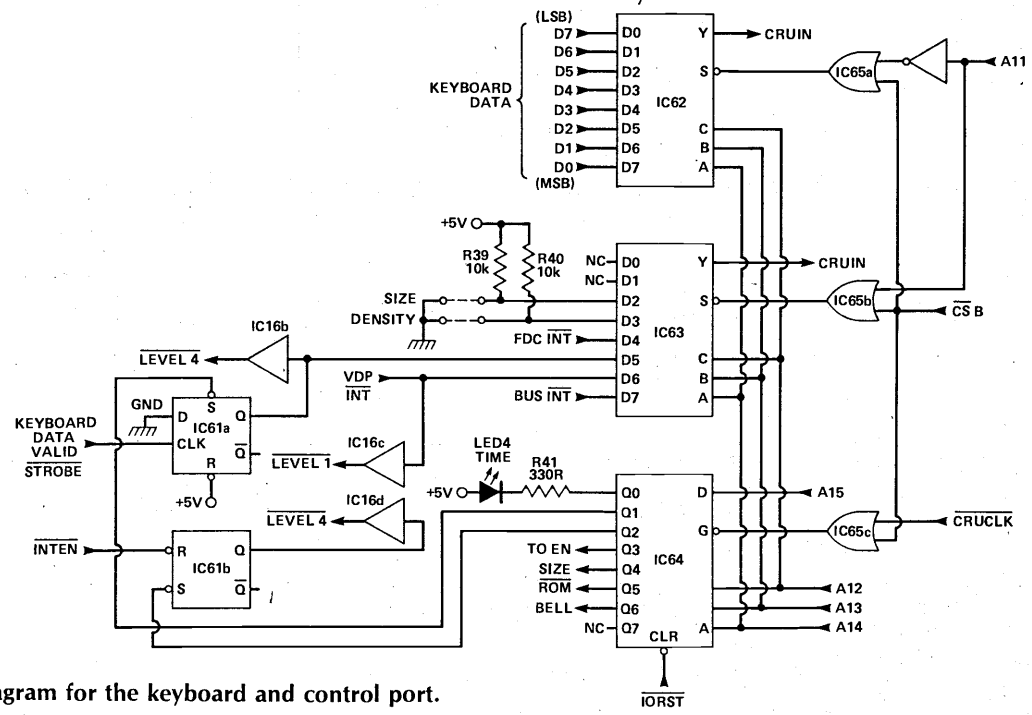



Fig. 1 Circuit diagram for the keyboard and control port.

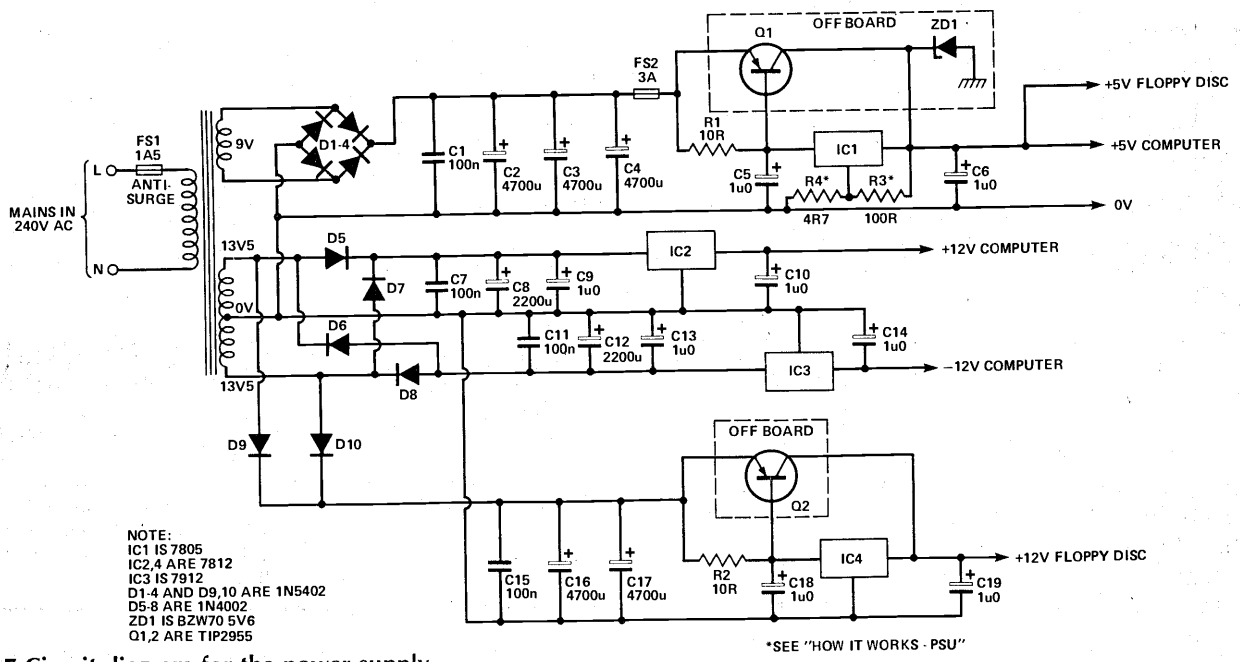



Fig. 7 Circuit diagram for the power supply.

## HOW IT WORKS — PSU

The computer main board and keyboard together require a 5 V at 3 A supply, together with low current ± 12 V rails. One amp plastic voltage regulators on small finned heatsinks are used for the 12 V supplies; for the 5 V supply a 1 A regulator is also used but the current-carrying capacity is boosted by bypassing it with a 15 A power transistor, the base current of which passes through the regulator. R1 prevents the off-load input current of the regulator from turning on the transistor when there is no load during testing. The resistor also increases the speed of operation of the transistor. The 1uf capacitors are for the stability of the regulator and the 100nF capacitors are used to remove fast transients orginating from the mains. The zener will clamp any spikes that reach the output.

R3 and R4 increase the output voltage of the 5V supply and may be needed if the volts at IC48 are low (less than 4.8V); otherwise omit R3 and replace R4 with a link.

To simplify the addition of floppy discs these are powered from the same board. The drivers require about 0A7 at 5 V which is also supplied by Q1; they also require + 12 V at 1A6 with higher surges at switch-on, and this is provided by a separate section using Q2 controlled by IC4.

## PARTS LIST — MAIN BOARD

Resistors (all ¼W 5% except where stated)

| | |
|---|---|
| R1,2 | 470R |
| R3-5,11,15,-32 | 4k7 |
| R6-8,20,21,-28,41 | 330R |
| R9,12,13,39-,40,46,52,55-,61 | 10k |
| R10,14,45,4-7,58,63,77 | 100R |
| R16-19 | 560R |
| R22 | 120k |
| R23,24,26,3-1,36 | 1k0 |
| R25,29,33,68 | 2k7 |
| R27 | 390R |
| R30 | 1k5 |
| R34 | 680R |
| R35,60,74,75 | 2k2 |
| R37 | 1k0 vertical preset pot |
| R38,53,54 | 100k |
| R42 | 6k8 |
| R43 | 3k9 |
| R44 | 39k |
| R48-50 | 8k2 |
| R51 | M0 |
| R56,59,68 | 4k7 resistor array |
| R57 | 22k |
| R62 | 27k |
| R64-67,71 | 150R resistor array |
| R69 | 5k6 |
| R72,73 | 3k3 |
| R76 | 10R |

Capacitors

| | |
|---|---|
| C1 | 1n0 ceramic |
| C2,18 | 16 V PCM mounting electrolytic |
| C3,7,25,26 | 10u 16 V PCB electrolytic |
| C4-6,9,10,1-3,17 | 100n polycarbonate |
| C8 | 47u 10V PCB electrolytic |
| C11,12,16 | 33p ceramic |
| C14 | 39p ceramic |
| C15,27 | 22u 16V PCB electrolytic |
| C19 | 100p ceramic |
| C20 | 22n polycarbonate |
| C21,31 | 10n polycarbonate |
| C22 | 330n polycarbonate |
| C23 | 2n2 polycarbonate |
| C24 | 5n6 polystyrene |
| C28 | 100u 16V PCB electrolytic |
| C29 | 330p ceramic |
| C30 | 470u 10V PCB electrolytic |

Semiconductors

| | |
|---|---|
| IC1,6,12,27,81 | 74LS04 |
| IC2,17,18,61,69,88,92 | 74LS74 |
| IC3 | 74LS86 |
| IC4,21,31,93 | 74LS00 |
| IC5,22,30 | 74LS02 |
| IC7,24 | 74LS10 |
| IC8 | TMS9911 |
| IC9,10,84,94-96,98,99 | 74LS244 |
| IC11 | TMS9995 |
| IC13,77,90 | 74LS08 |
| IC15,34,35 | 74LS138 |
| IC16,66,80,82,83 | 74LS07 |
| IC19 | 74LS164 |
| IC20,79 | LM339 |
| IC23 | 74LS20 |
| IC25,65,78,91 | 74LS32 |
| IC26 | 74LS612 |
| IC28,29 | 74LS27 |
| IC32 | TMS4500 |
| IC33,85 | 74LS139 |
| IC36-43 | TMS4164 |
| IC44,97 | 74LS245 |
| IC45-47 | TMS2564 |
| IC48 | TMS9929 |
| IC49-56 | 4816 or 8118 |
| IC57,58 | 4016B |
| IC59 | LM1889 |
| IC60 | 4013 |
| IC62,63 | 74LS251 |
| IC64 | 74LS259 |
| IC67,68 | TMS9902 |
| IC70 | 74LS123 |
| IC71 | 75189A |
| IC72 | TL084 |
| IC73 | 74LS73 |
| IC74 | 75188 |
| IC76 | TMS9909 |
| IC86 | 74LS297 |
| IC87 | 74LS163 |
| IC89 | 74LS2001 |
| Q1,3,4 | BC182 |
| Q2,5 | BC212 |
| Q6 | BC212 |
| D1-4 | 1N4148 |
| LED1-4 | LEDs to choice |

Miscellaneous

| | |
|---|---|
| L1 | 4u7H |
| L2 | 22uH |
| L3 | 33uH |

PCB (see Buylines); case (see Buylines); IC sockets; I/O connectors to suit; UHF modulator (UM1233 or UM1286).

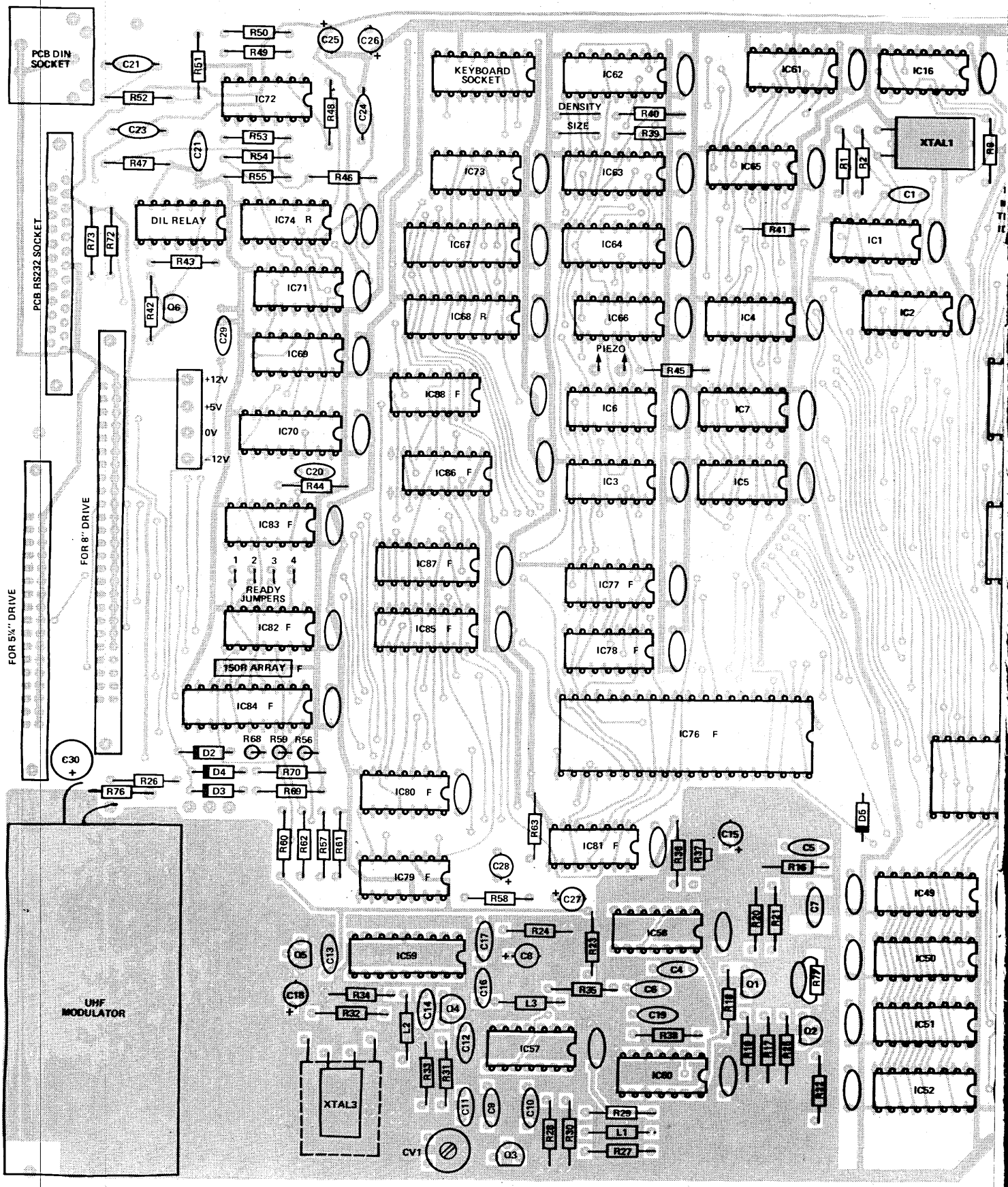
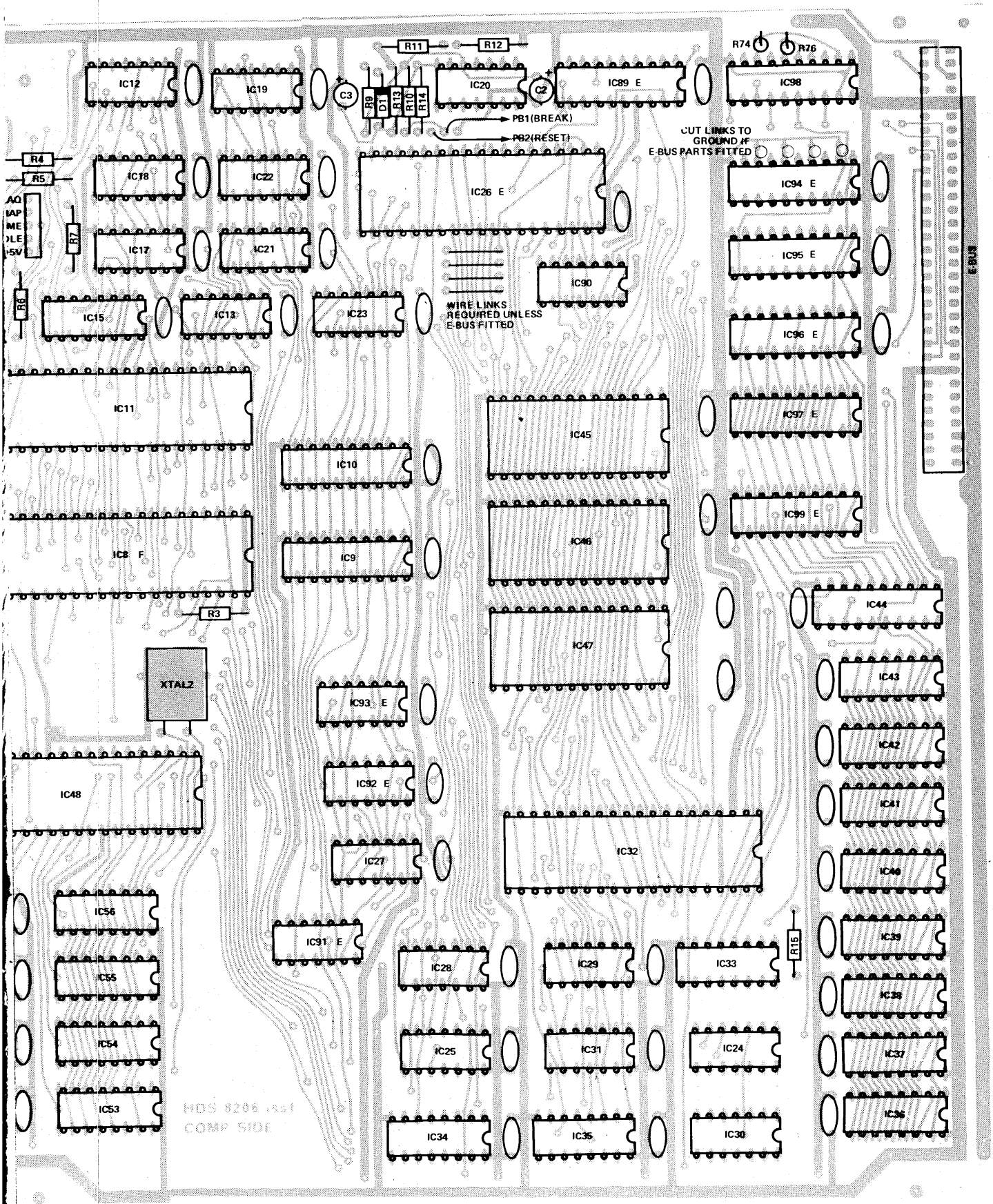



Fig. 4 Component overlay for the Cortex main board. The numerious unmarked unpolarised capacitors are all supply line decoupling capacitors and are 47n ceramic. The tracking shown is of the top foil of the earliest version and may differ on later generation boards (as may the patterns shown for the PSU and keyboard). ICs marked with various letters are for expansion options and are




not supplied with the most basic version of the kit. The expansion options are as follows: R = RS232; F = floppy disc interface; E = E-BUS interface.

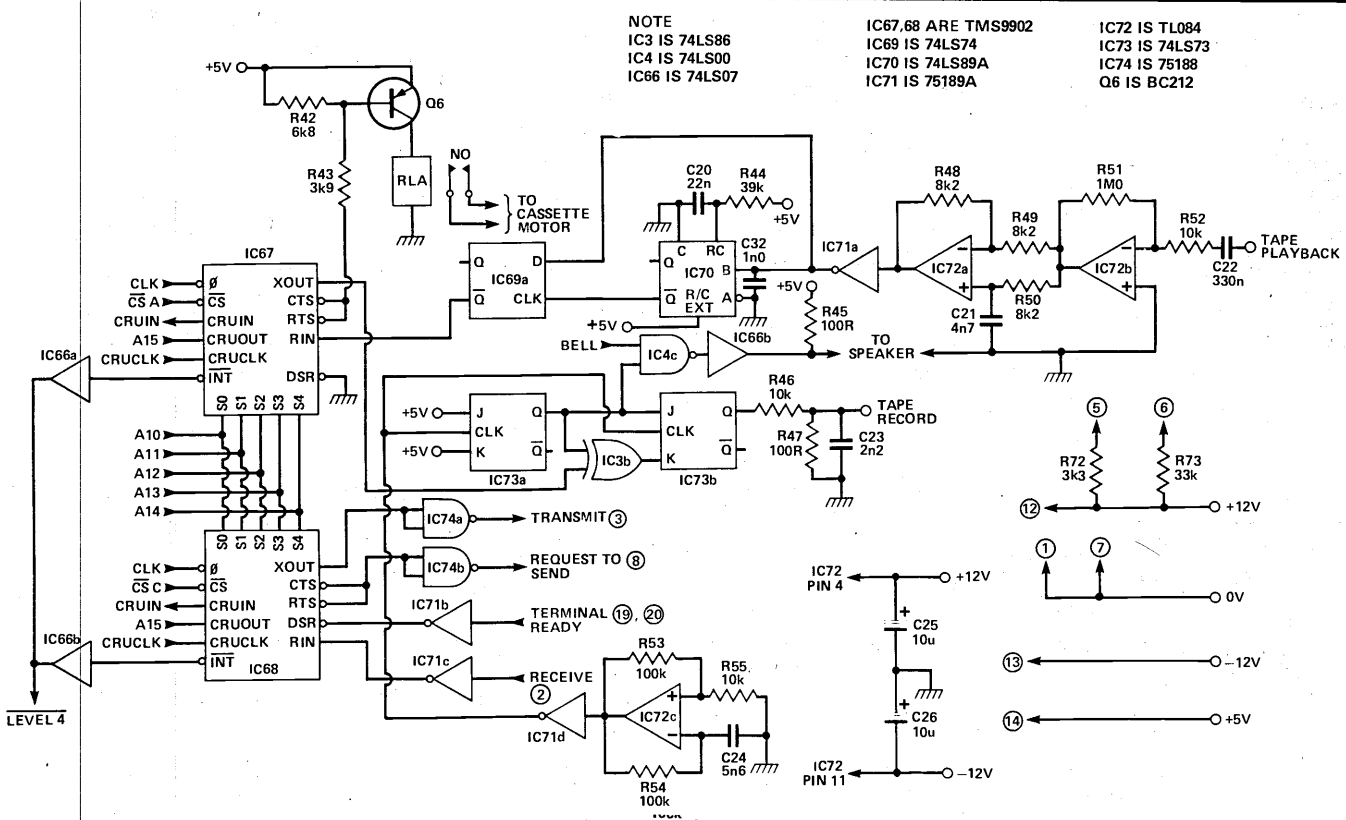



Fig. 2 Circuit diagram of the RS232 and cassette interfaces. Circled numbers are RS232C socket numbers.

## HOW IT WORKS — RS232 AND CASSETTE PORT

The RS232 port consists of IC68, a TMS9902 Asynchronous Communications Controller (ACC) and the TTL-to-RS232 signal level shifters (IC74a,b and IC71b,c). IC68 is a completely software-controlled device; its baud rate can be set at anything from 46 baud to over 100,000 baud. The number of bits to be transmitted or received can also be changed, as can the type, the parity and number of stop bits. The CPU drives the ACC through the serial I/O (CRU) bus. The ACC is decoded as a 32-bit block, each bit being selected by the five address lines A10–A14.

The cassette interface uses another ACC, IC67. First a 4.8kHz op-amp oscillator (IC72c) drives a level shifter (IC71d) before being divided by two in the first flip-flop (IC73a). This ensures that the waveform has a unity mark-space ratio. The serial output from IC67 then controls the action of the second flip-flop, IC73b, via the EXOR gate IC3b. When the output is high, IC73b acts as a shift register, passing through the 2.4kHz tone; however, when the ACC output goes low then synchronously at the next clock pulse, IC73b starts to divide by two, hence generating 1.2kHz. The key point here is the synchronous switch from one tone to the other. The signal is high-pass-filtered and attenuated by R46, R47 and C23 before passing to the tape recorder.

On playback the signal is first amplified by a factor of 100 and buffered in IC72b before going through an all-pass filter, IC72c. This is necessary because of the nature of tape recording. When square waves are recorded on tape they are accurately captured; however, on playback frequency equalisation is carried out in the tape recorder but the phase relationship is destroyed, resulting in a 'spiky' sine wave. This is corrected by the linear phase-shift-versus-frequency characteristic of the all-pass filter. Thus the original square wave shape is recovered at the output of IC72a. This is then level-shifted by IC71a and used to trigger a monostable (IC70a). At the end of the monostable period (312.5 uS) the state of the signal is sampled by the D-type flip-flip IC69a. As the half-periods of the two tones lie either side of the monostable period, each tone generates the opposite logic level at the sample point.

## HOW IT WORKS — FLOPPY DISC CONTROLLER

The TMS9909 (IC76) is a highly complex micro-controller, designed to work in conjunction with the TMS9911 DMA controller to transfer data from floppy discs. The FDC can control up to four drives which can be a mixture of two sizes or types.

All signals that go to the drives are open-collector buffered by IC80,82,83 and terminated by a resistor pack on the last drive in the chain. The signals from the drives are terminated on the board by a resistor pack and then buffered by IC84.

The raw data pulses from the drive, after being buffered by IC84a, are stretched by a monostable (IC70b) by an amount dependent on the data transfer rate selected by the 'SIZE' I/O bit and the 'DDEN' (double density enable) signal (see Table 1). The output of the monostable is used to control IC77, a digital phase-locked loop. The output of IC77 is, in the unlocked state, half the input clock frequency. When the loop is locked to a signal then the PLL inserts or deletes clock pulses in the pulse stream, thus shifting the average frequency. The programmable divider IC87 and divider IC69b are controlled by the 'SIZE' and 'DDEN' signals to select the correct clock frequency. The raw data is synchronised by IC88 to the PLL clock and then fed to the FDC. The FDC separates the interleaved clock and data bits from the pulse stream and sends data bytes via single byte DMA transfers to main memory.

Mini-floppy (5¼") drives require a motor control signal to start and stop the disc rotating. Upon starting, the disc will not be ready for data transfers for one second while the disc gets up to speed. To reduce the time required to access the disc repeatedly IC79b keeps the motor running for five seconds after it is de-selected and IC79a provides the initial one second 'not ready' signal to the FDC. For standard (8") drives that don't generate a 'ready' signal there is a set of four jumpers.

The BASIC interpreter has a 'BOOT' command which causes the FDC to read the first track from disc 1 and execute it as a machine code program. This could, for example, then search for and load the UCSD interpreter. In order that the system can boot from any type of disc there are two jumpers called 'SIZE' and 'DENSITY' which are read by IC63. This enables the BASIC interpreter to set up the FDC correctly.

NOTE
IC3 IS 74LS86
IC69 IS 74LS74
IC70 IS 74LS89A
IC76 IS TMS9909
IC77 IS 74LS08
IC78 IS 74LS32
IC79 IS LM339
IC80,82,83 ARE 74LS07
IC81 IS 74LS04
IC84 IS 74LS244
IC85 IS 74LS139
IC86 IS 74LS297
IC87 IS 74LS163
IC88 IS 74LS74

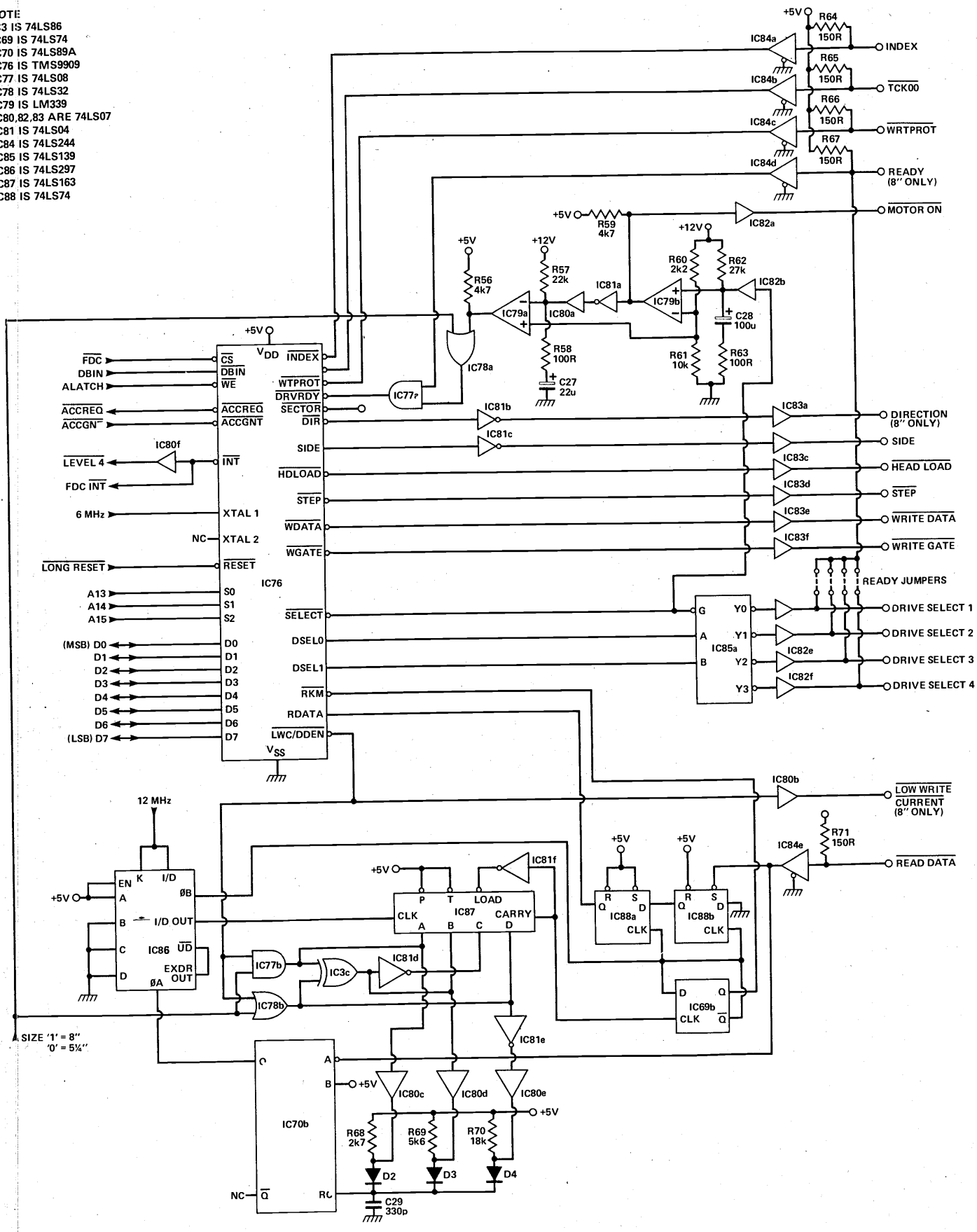
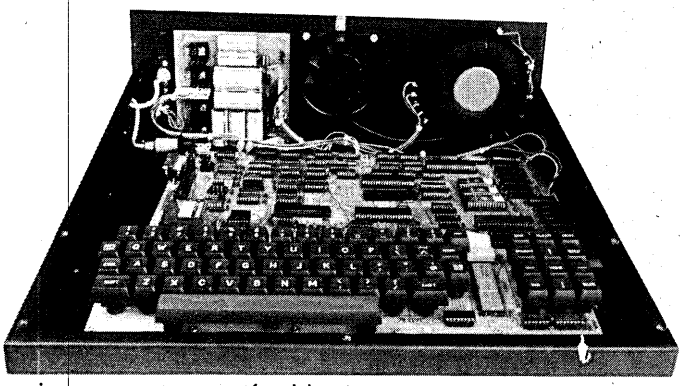
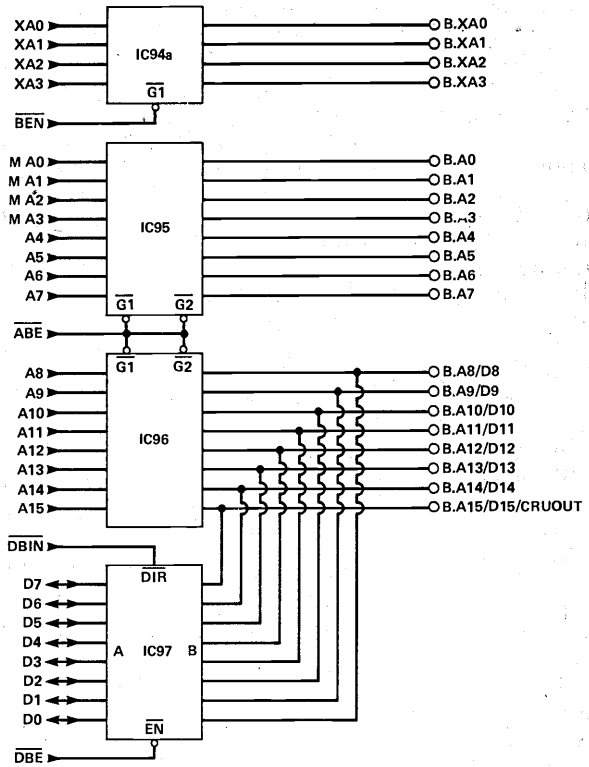
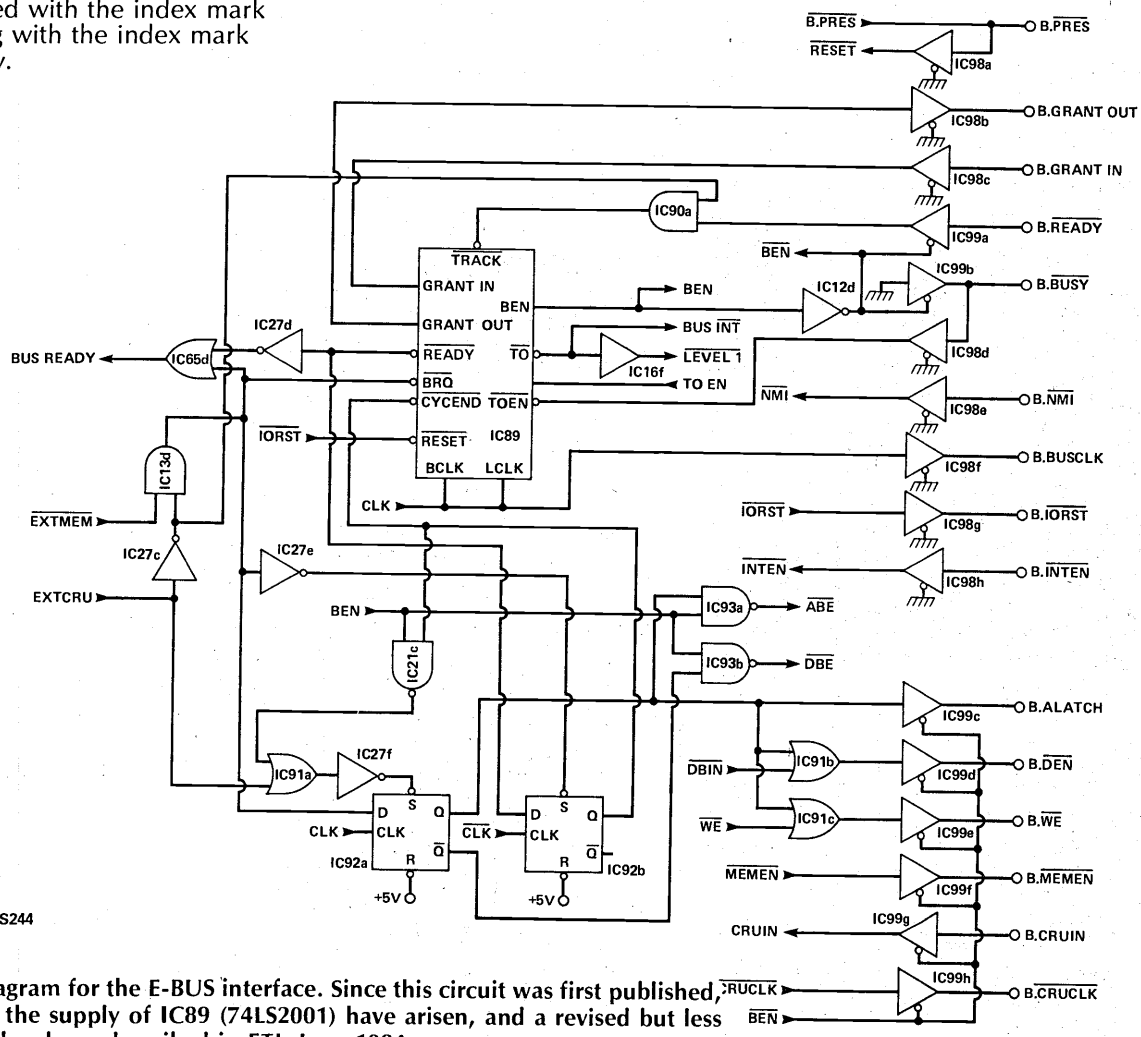



Fig. 3 Circuit diagram for the floppy disc controller section.



passing over a wave of solder in a solder bath during factory assembly. With plated-through boards it is particularly important not to make errors of construction as removal of soldered-in parts is more difficult than on conventional boards and the chances of this being required are much reduced by fitting ALL parts before soldering — if the last part left for fitting is not the one required for the last space you can be pretty sure that the required part is in the wrong holes! IC sockets should be regarded as essential; these are provided with the kits and should be fitted with the index mark corresponding with the index mark on the overlay.




NOTE
IC12,27 ARE 74LS04
IC13 IS 74LS08
IC16 IS 74LS07
IC66 IS 74LS07
IC89 IS 74LS2001
IC90 IS 74LS08
IC91 IS 74LS32
IC92 IS 74LS74
IC93 IS 74LS00
IC94-96,98,99 ARE 74LS244
IC97 IS 74LS245

**Fig. 5 Circuit diagram for the E-BUS interface. Since this circuit was first published, difficulties over the supply of IC89 (74LS2001) have arisen, and a revised but less versatile circuit has been described in ETI, June 1984.**

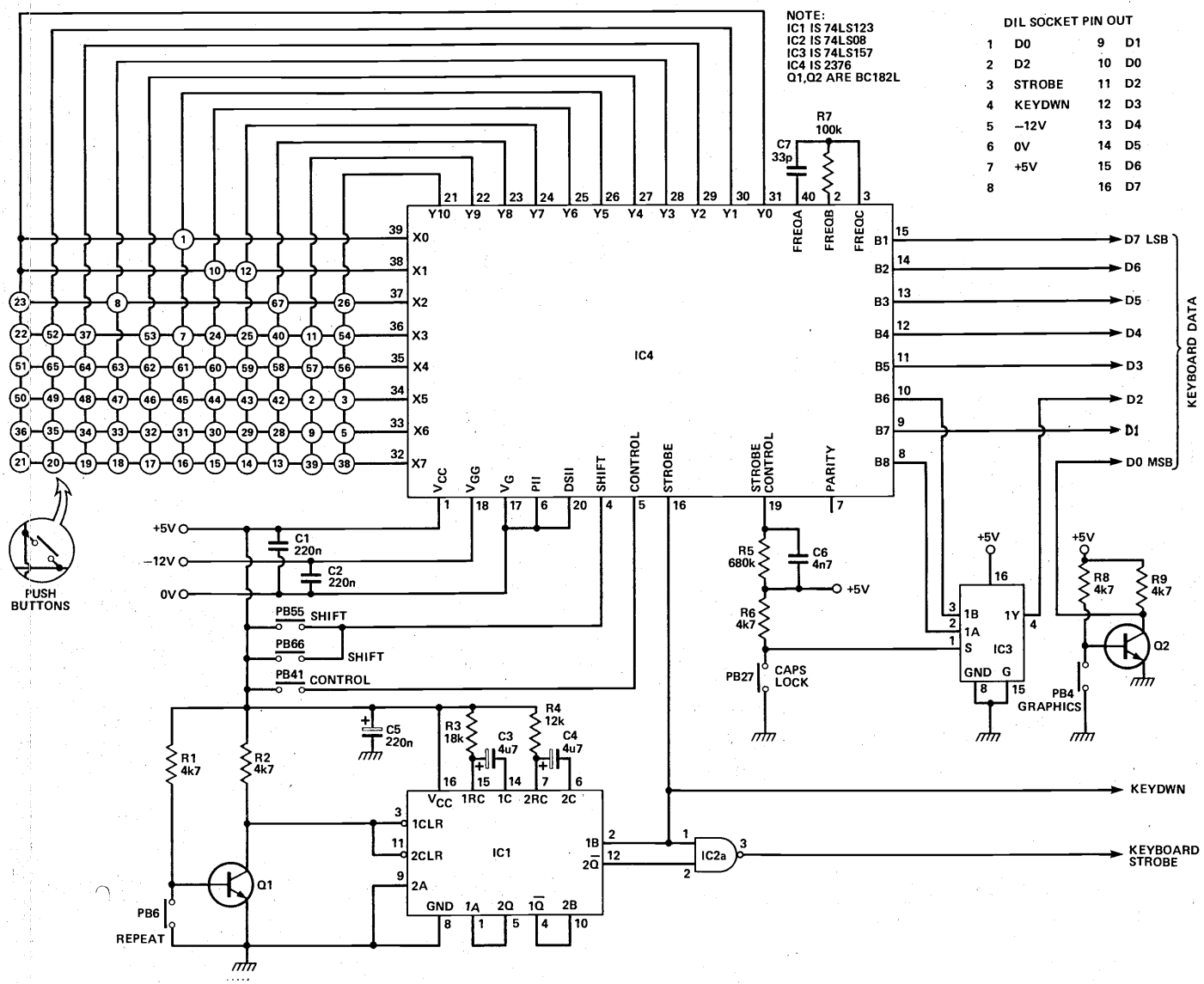



Fig. 6 Circuit of the keyboard; note that the component numbering is separate from all the previous circuitry.

## HOW IT WORKS — E-BUS

The E-BUS is a powerful and compact bus which allows many intelligent cards to share a common resource of memory and I/O cards. In order to share out the resources on the bus, each card has a priority according to its position. This is done by passing a signal down the bus which goes into each card as GRANTIN and comes out as GRANTOUT to form the GRANTIN of the next card. A second signal, BUSY, tells each card if the bus is in use or free. If the bus is free and a card requires the bus, it disables the lower priority cards with the GRANTOUT signal and if the GRANTIN signal and BUSY are OK it asserts BUSY and enables its data and address bus buffers.

Once the bus transfers are complete or if a higher priority card requires the bus, then the card will relinquish control. All these events are synchronised by a backplane clock, BUSCLK. Each data transfer that takes place must signal its completion using READY.

The 74LS2001 gate array (IC89) contains the bus arbitration and control logic to gain and release the bus with timeouts upon error conditions. If the card cannot gain control of the bus after 128 clock cycles, it aborts with a timeout interrupt. Also, if after 16 clock cycles the transfer has not been signalled as complete using the READY line, the controller completes and issues a timeout interrupt.

The E-BUS has provision for a multibit interrupt code signalled by the INTEN signal. This interface only provides a single interrupt level using the INTEN signal. The data, address and interrupt signal are multiplexed onto the same pins to conserve connections. The ALATCH signal is used to enable the address latches when the address is on the bus. Then either DEN or WE will be signalled, to show that either a data read or write is occurring and that data is now on the bus. The INTEN signal can be used to latch the interrupt code.

## KEYBOARD

The keyboard is a separate unit providing a fully encoded output. Most of the work is carried out by the 2376 keyboard encoder (IC4). This IC contains a 50 kHz oscillator and two ring counters of eight and 11 stages, the outputs of which form an XY matrix across which the switches are connected. By this means each key is sequentially scanned. The closing of one of the switches for a sufficient length of time for switch bounce to be completed causes the scanning to stop; a 'valid' signal now appears on the strobe output. The encoder also contains a 2376-bit ROM (hence the IC name) arranged as three groups of 88 words of nine bits. The shift and control inputs select one of the three groups and the individual word is addressed by the ring counters.

IC3 is a data selector. D2 is either the output B6 or B8 depending on whether upper or lower case characters are selected by the CAPS LOCK switch. Repeated entry of a character is accomplished by multiple strobe signals from IC1, which is a dual monostable arranged as an oscillator and is enabled by a high level on the clear inputs.

# CORTEX PART 3

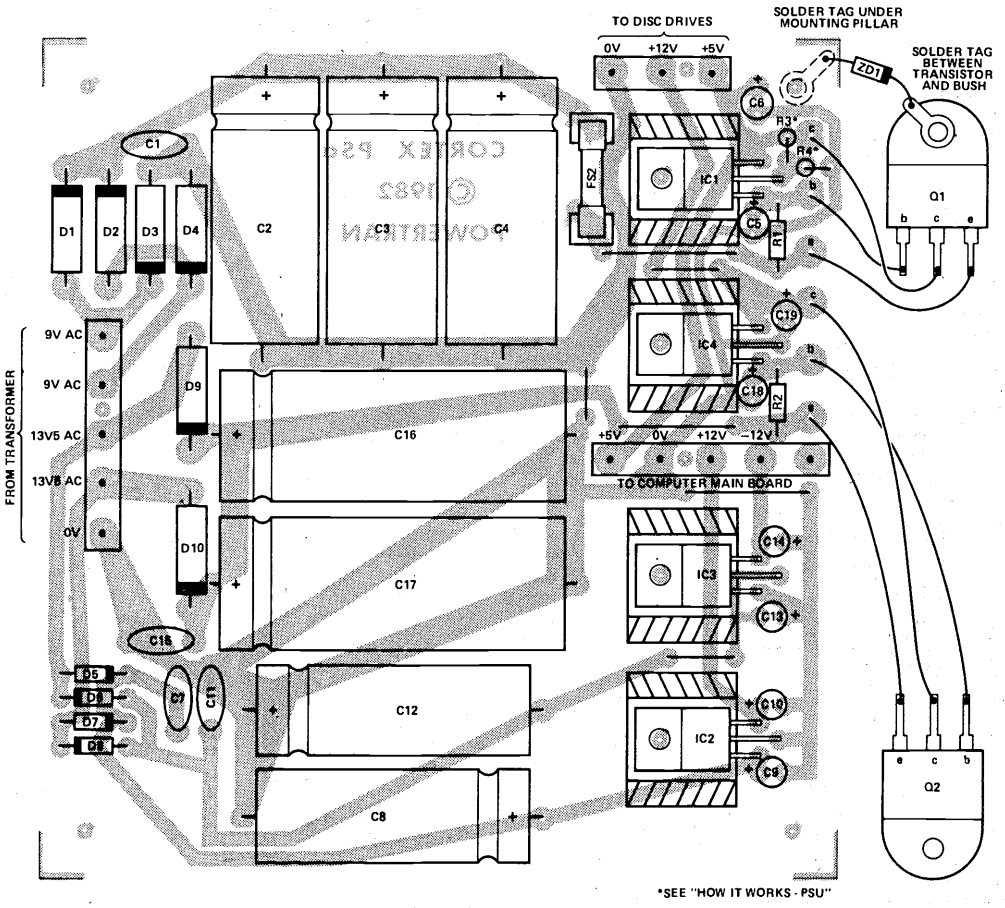



Fig. 1 Component overlay for the power supply.

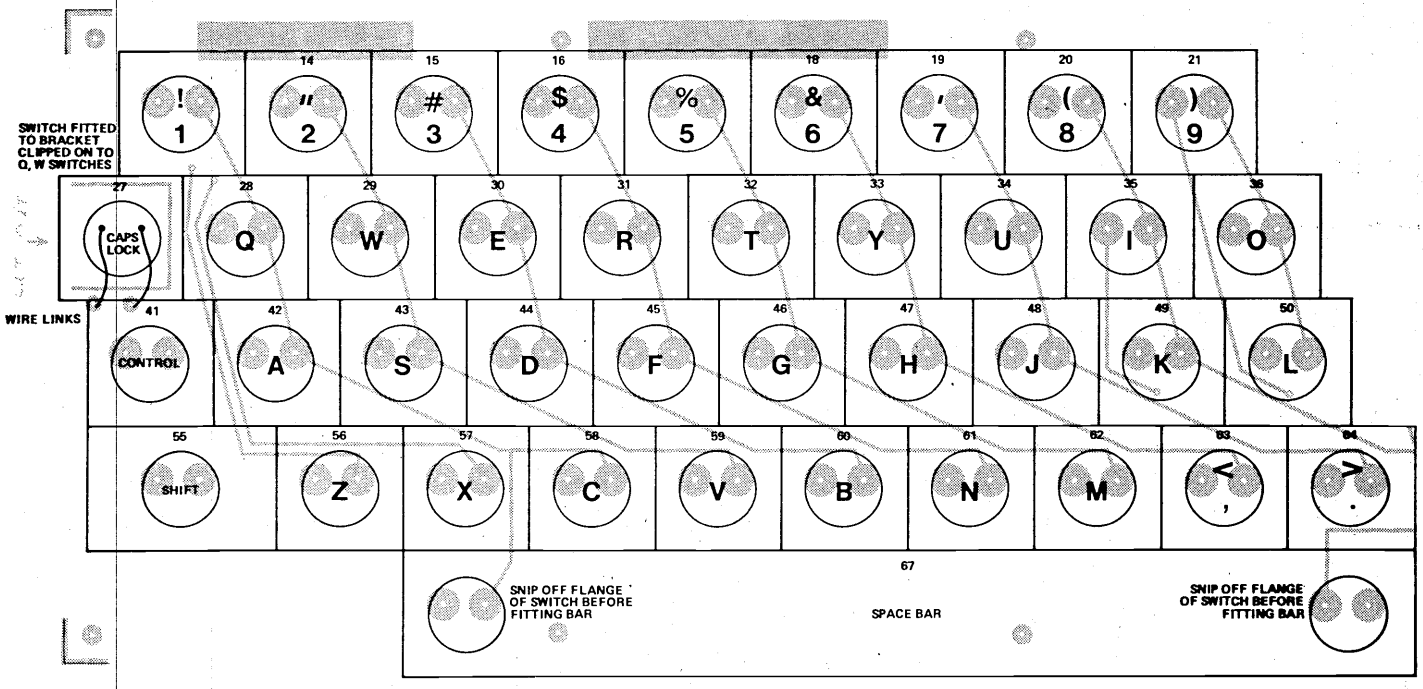



Fig. 2 The keyboard overlay, in two halves so it doesn't get stapled into illegibility.

## PARTS LIST

POWER SUPPLY
Resistors (all ¼W, 5%)

| | |
|---|---|
| R1,2 | 10R |
| R3,4 | see "How it works — PSU" |

Capacitors

| | |
|---|---|
| C1,7,11,15 | 100n polyester |
| C2-4 | 4700u 16 V axial electrolytic |
| C5,6,9,10, 13,14,18,19 | 1u0 35 V tantalum |
| C8,12 | 2200u 25 V axial electrolytic |
| C16,17 | 4700u 25 V axial electrolytic |

Semiconductors

| | |
|---|---|
| IC1 | 7805 |
| IC2,4 | 7812 |
| IC3 | 7912 |
| Q1,2 | TIP2955 |
| D1-4,9,10 | 1N5402 |
| D5-8 | 1N4002 |
| ZD1 | BCW70 5V6 |

Miscellaneous
PCB (see Buylines); one off three way connector; two off five way connectors; transformer (13.5-0-13.5 V at 3 A, 9 V at 4 A; fuseholder clips; four off TV5 heatsinks.

KEYBOARD
Resistors (all ¼W, 5%)

| | |
|---|---|
| R1,2,6,8,9 | 4k7 |
| R3 | 18k |
| R4 | 12k |
| R5 | 680k |
| R7 | 100k |

Capacitors

| | |
|---|---|
| C1,2,5 | 220n 35 V tantalum |
| C3,4 | 4u7 16 V tantalum |
| C6 | 4n7 ceramic |
| C7 | 33p ceramic |

Semiconductors

| | |
|---|---|
| IC1 | 74LS123 |
| IC2 | 74LS08 |
| IC3 | 74LS157 |
| IC4 | 2376 |
| Q1,2 | BC182L |

Miscellaneous
PCB (see Buylines); 67 off momentary push-to-make switches; one off latching push-to-make switch; set of 67 double shot moulded key tops; IC sockets; switch mounting bracket; mounting pillars etc.

The CAPS LOCK switch is physically different from the rest and is wired in with wire links. Press this switch into its bracket together with the Q and W switches before fitting to the board. It is most important that the switches fit squarely on the board. The best way to be sure of this is to solder only one pin of each switch and then holding the board, press in turn each of the switches while reheating the soldered joint. If any are misaligned this will correctly position them. The key tops can now be pressed on and the remaining pins soldered.

The power supply is on a single-sided PCB with six wire links and it is best to fit these before any components. Connections to and from this board are made via connectors and to ensure that their pins are soldered in squarely, fit the sockets on to them during soldering. The power supply is all on the back panel and the power transistors use this as a heatsink, being fitted to it with mica insulating washers.

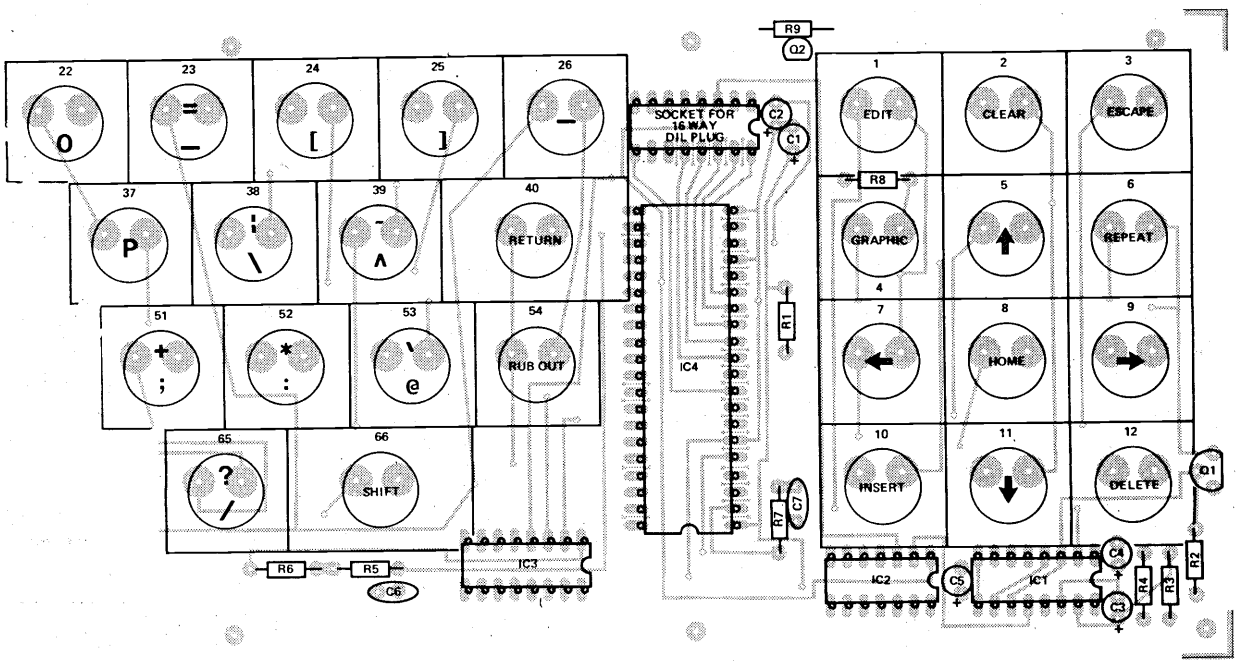
As well as holding the input and output sockets, the rear panel has provision for a cooling fan and one should be fitted when disc drives are used.

The disc drives pass through the front panel and are screwed onto a mounting plate. Plates on the sides of the drives press against the panel, thereby making a rigid sub-assembly which fits into the cover of the computer. The standard kit has a panel with no cut-outs for disc drives and a new panel is provided with the drives when purchased.

There are two positions in which the main board can be fitted. The board has provision for a Eurocard connector for expansion purposes and there is a cut-out in the side of the computer through which the connector passes; for external expansion the board fits at the far right hand side. However, if the add-on units are to be fitted internally then the position to the left is used.

## BUYLINES

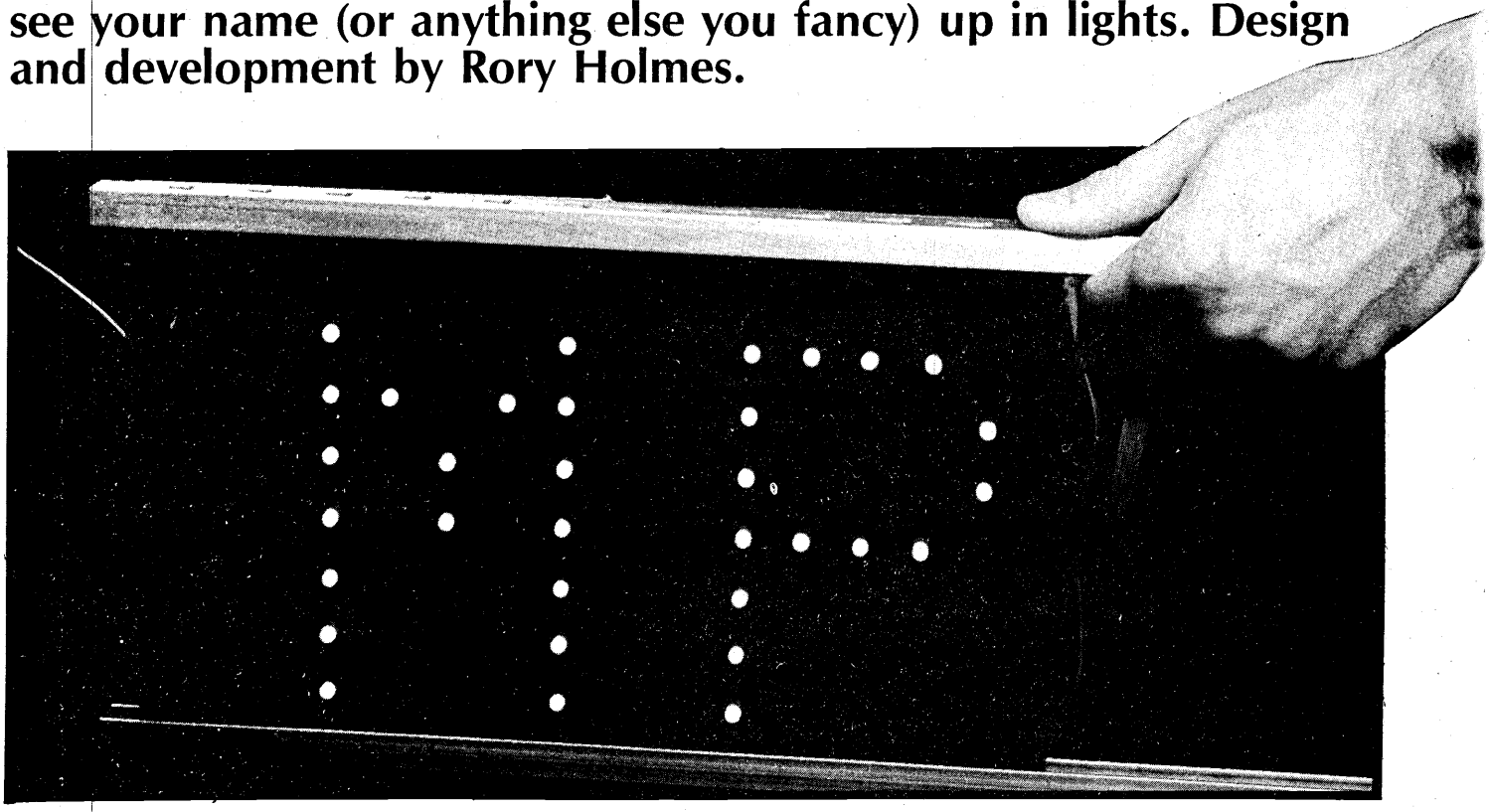
**Powertran are supplying complete kits of parts and component packs for the Cortex. A complete 64K Cortex kit will cost £295 plus VAT, carriage free. A ready-built 64K Cortex will cost £395 plus VAT, carriage free. Prices for add-ons (eg floppy discs, RS232C interface, memory expansion etc) and for component packs (eg PCB, semiconductors etc) can be found in Powertran's brochure. Powertran Cybernetics, Portway Industrial Estate, Andover, Hants SP10 2NM. Telephone 0264 64455.**




ETI

# MESSAGE PANEL

**Wish fulfilment for ad-men, exhibitionists (!) and ego-trippers; see your name (or anything else you fancy) up in lights. Design and development by Rory Holmes.**



MOST readers have probably seen moving message panels of the electronic type by now, either small shop window displays or the larger newsboard versions. They are ideal for advertising, description, news, or just providing information — a moving message in lights is unparalleled at drawing people's attention. Commercial message displays are costly items, being quite complex just in the sheer number of components. Although useful in shops, offices and the home, the chance to have your name in lights has until now been an expensive luxury. ETI decided to design a message panel that would be better suited to the constructor's budget — simple to build and easy to use. By simplifying the circuitry we came up with a design that was easily expandable by single characters and could be controlled in a number of different ways.

The message panel uses light emitting diodes as the illumination, characters being formed on a seven by five matrix of LEDs. The display can be built to any length using identical 'character cards' connected in daisy-chain fashion (see the PCB overlay diagram of Fig. 6); a new card can be simply added on to the end with direct wire links to the previous card.

## Cooee Mr Shifter

The circuitry we have used employs shift registers to store and move the display pattern, each parallel output directly driving the LEDs rather than multiplexing them. Scrolling of the message across the display panel is thus achieved by simply clocking the shift registers as the character pattern data is presented to their inputs. The speed at which this is done determines the speed at which the message passes through the display panel. Since there is one data input for each of the seven rows of shift registers and one shift clock line the entire panel may be driven from one eight bit computer port.

The only other connections required are for the power supply and this can be an ordinary 9 V unregulated supply. Fig. 2 shows the circuit for the power supply we used — each card takes about 200 mA and therefore a 2A supply should do for about 10. Next month we shall feature a small interface PCB that allows the message panel to plug directly into a Sinclair ZX81 or Spectrum computer, complete with the software listings of BASIC programs for entering and running repeat messages of any length with the ASCII character set. We are also going to publish a custom control

## SPECIFICATIONS

**Requires a single 5 to 9 V supply (unregulated).**
**Characters are 6″ high on a 7 by 5 matrix.**
**Each card consists of a 7 by 8 LED matrix.**
**One eight bit port drives the whole panel (seven row data inputs and one clock input).**
**Cards may be plugged together daisy-chain fashion for long character strings.**
**Any message scroll speed is obtainable.**
**Cost of components for each card is around £12.**
**Standard eight-bit shift registers are used for storing and moving the display pattern.**

card that can store and run messages using battery-backed up CMOS memory.

## Construction

We recommend that about six character cards are used for maximum readability, although even two cards will provide perfectly legible messages. They can all be constructed and tested individually (as time and money will allow). Solder in all the links on the PCB first, following the overlay diagram in Fig. 6. The seven links that connect in a line to the positive rail should be made with thick tinned copper wire, while the other 13 links are made with thinner tinned copper wire. If the wire is thin enough it may be laced through all the link connection holes as a continuous wire and then soldered up; this greatly speeds up the process. The resistors, capacitors and transistors should then be soldered in followed by the ICs. Be sure to check the component orientation at this point to save expensive mistakes later on. IC sockets may be used and are recommended to ease any replacement problems in the future. Also note that the components labelled R101, 102 etc are repeated seven times per character card, whereas the additional components (drawn separately in Fig. 1) are required only once on each card.

Now we come to the LEDs. We strongly recommend that good quality red LEDs of the high efficiency type are used, since this makes all the difference both to current consumption and readability of the display. As mentioned in Buylines an ideal LED, having even brightness and high efficiency, can be obtained in quantities of 200 from Zaerix Electronics. At around 11p each they represent good value for money, being sufficient for four character cards. The LEDs should be soldered in at a height of about 16 mm from the surface of the board; their cathodes all go to the 0V rails (the cathode is nearest to the flat).

The following construction method makes it easier to achieve a uniform height and neat rows. All the LEDs should first be inserted and pushed in to less than the required height; the board is then turned over onto a flat table top, and raised up to the right height on suitable blocks. The LEDs may be tapped on their leads to make contact with the table, thus ensuring they all reach the same height. Initially, only their cathodes should be soldered to the 0V rails. When they are all secure the board may be turned over to even up all the rows of LEDs and make any final adjustments before completing the soldering of their other leads. Vero terminal pins may be soldered in if desired to terminate the connections at the card edges; alternatively miniature single-way sockets or just direct wiring could be used.

## Testing

When assembly is complete the board may be tested as follows. Wire up the 0V rail and positive rails to a 9V supply of about 500mA; a PP9 should suffice for one card, although the suggested mains power supply of Fig. 2 (suitable for powering 10 cards) would also be ideal. Now connect a square wave clock signal to the input marked clock on the overlay (preferably variable from about 1 to 10 Hz). If no signal CMOS circuit shown in Fig. 3 could be used. Hopefully all the LEDs should remain in their off states; the power-on reset to the shift registers ensures that the data outputs are all low, and the data inputs to the registors are all held low by the resistors R105, R205 etc. With one end of a crocodile clip lead on the 9V rail, touch the other end to one

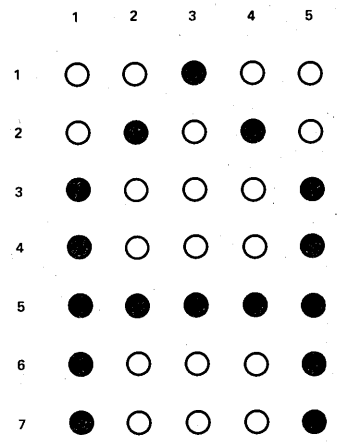
## BUYLINES

**Most of the parts are completely standard but be certain to get the B version of the CMOS 4015. The LEDs should ideally be 5 mm red high efficiency types as explained in the text. Zaerix Electronics Ltd will supply a suitable LED (the L53ID) in quantities of 200 for £23.60 — you can find them at Electron House, Cray Avenue, St Mary Cray, Orpington, Kent BR5 3QJ.**

## HOW IT WORKS

**The circuit operation of the message display panel is completely straightforward. Figure 1 shows the circuit for one row of eight LEDs - this circuit is identically repeated for all the seven rows of each character card. Essentially the message panel is just seven long shift registers (the total length, in multiples of eight, being up to the constructor). The state of each bit of the shift registers is displayed on an LED, and these form the matrix on which characters of the message are written. The shift clocks of all the registers are tied together to form a common clock which simultaneously shifts all the information through the registers. The patterns of display information presented at the seven register data inputs are thus shifted down the line of registers to form the 7 by 5 characters.**

**For instance, the character 'A' has the 7 by 5 pattern of**

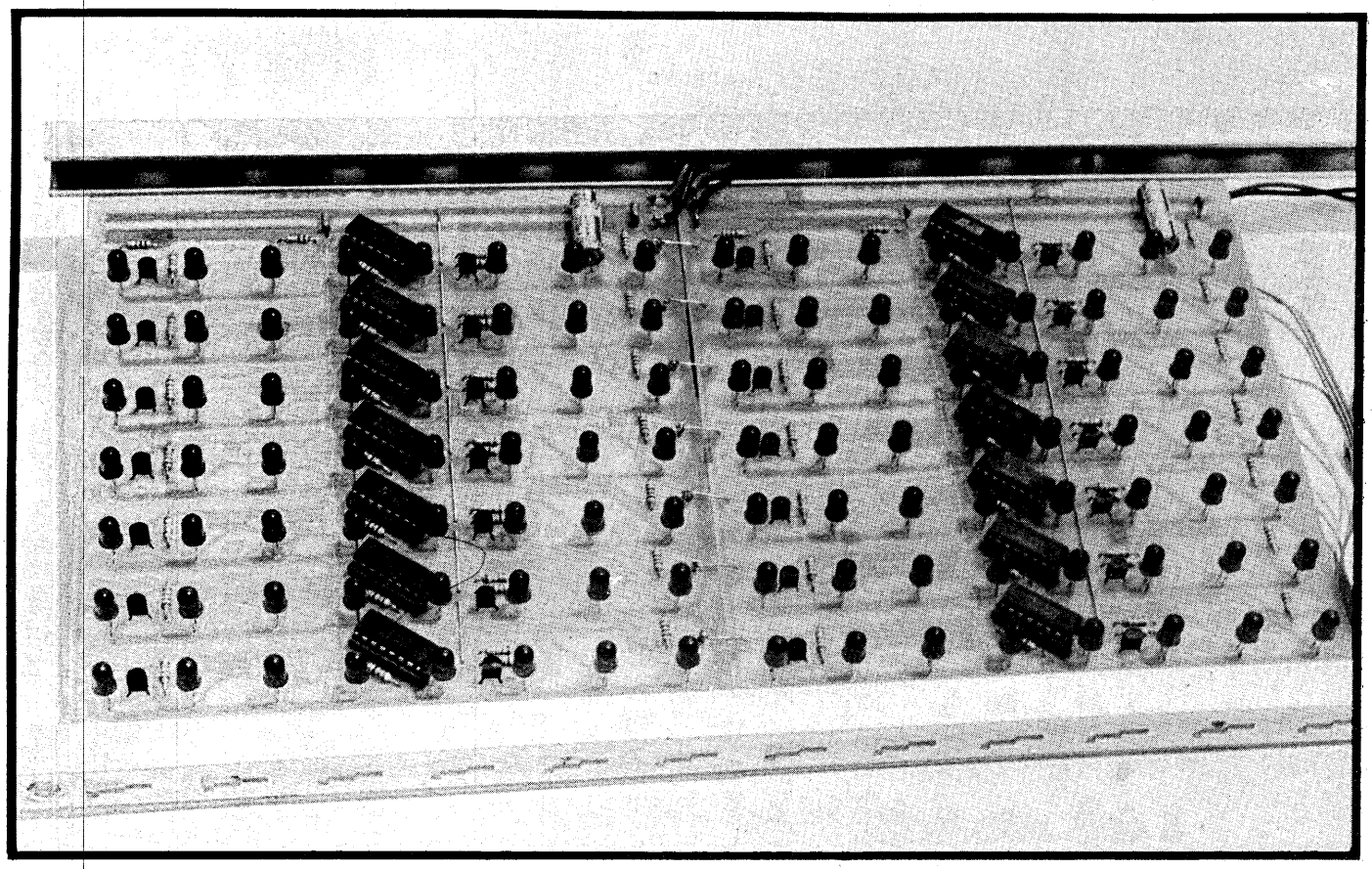



**which consists of five binary numbers 1111100, 0010010, 0010001, 0010010, 1111100. Therefore to write this character onto the LED matrix these binary numbers are input as logic data to the seven row inputs, and the shift clock is pulsed briefly after each byte of data to move the display pattern down the columns. The character may then be scrolled, exactly as it is, down the display panel simply by repeated pulsing of the shift clock line.**

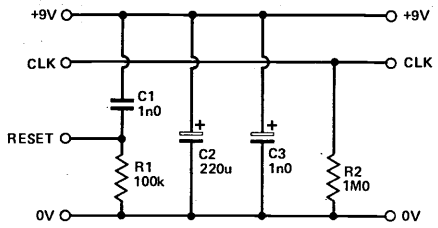
**The actual circuit is based around IC1, a dual four-bit shift register. Both sections are wired together in cascade to produce an eight stage serial-in-parallel-out shift register. IC1 is used to drive six of the LEDs directly. The fourth output bit of each section, which connects in cascade to the next section, has been buffered by transistor drivers, Q1 and Q2, so that the LED loading on the logic levels does not affect the data transfer. The resistors R1 and R4 determine the current supplied to LED1 and LED5. The reset pins (6 and 14) of all the registers are tied together and taken to a power-on reset circuit formed by R1 and C1. This is to prevent the LEDs coming on when power is first applied.**

**Resistor R105 on each register input is not strictly necessary in a fully built panel, but it keeps the data inputs tied to a definite logic-low level, which is useful when testing and unplugging boards. Similarly R2, used once on each card, will hold the common clock line to logic low when not in use. C2 and C3 provide decoupling on an individual card basis.**

**The cards may be powered from any voltage between 5 to 9V. Since some of the LEDs are driven directly from CMOS no more than 9V should be used, as the current and dissipation become too high. At 9 V the LEDs should be quite bright, drawing about 11mA each. For a standard character set there will never be more than 20 LEDs illuminated at once and therefore a total of only 200mA is sufficient for each card. Ten or so boards could thus be powered from the simple power supply circuit shown in Fig. 3.**

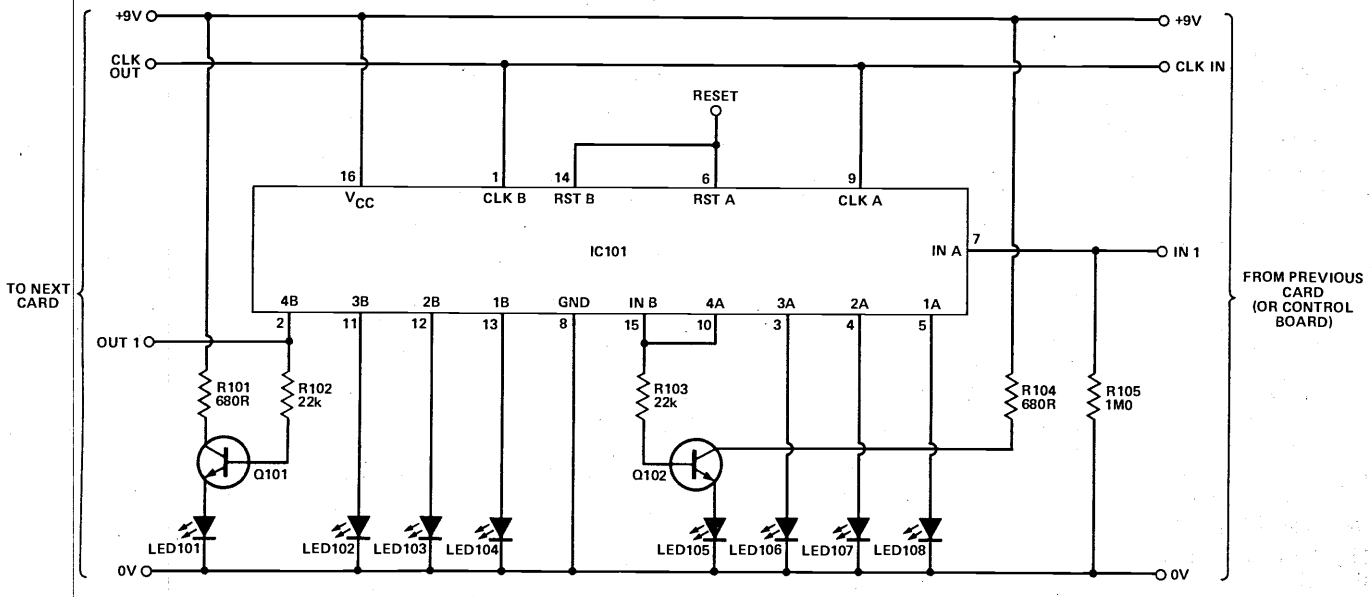


Here we have two character cards linked together and bolted onto the mounting frame (we used standard shelving bracket extrusion). The wiring to the computer port is connected to the right-hand board.




NOTE:
IC101 IS 4015B
Q101,102 ARE BC184L
LED101-108 ARE 5mm
RED LEDS (L53 ID)

**Fig. 1 Circuit diagram for one channel, or row, of one character card. The components in the larger section are numbered R101 etc, and are repeated seven times on each card; the components in the smaller diagram are used only once per card.**



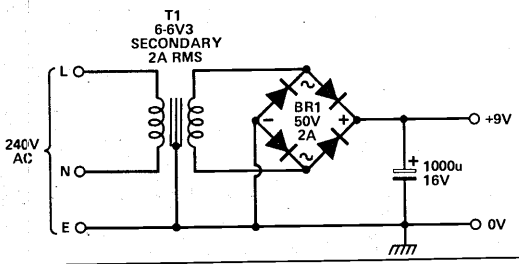



**Fig. 2 (Left) Circuit diagram of a power supply with a rating suitable for driving 10 of our character cards.**

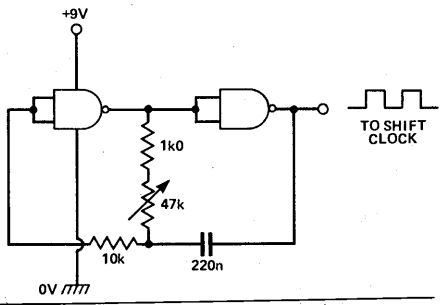



**Fig. 3 (Right) Circuit diagram of a simple clock generator which could be used to test the message panel.**

of the data input pins; this makes a logic high data bit enter the shift register and causes illuminated LEDs to move down the shift register at a speed determined by the shift clock. If the data input is kept as a high, all the LEDs in one row will illuminate, and then successively turn off when the data input goes low. All seven rows should be checked in this manner.

## Casing The Cards

The PCB has been designed to allow each card to be bolted into a rigid case. We used the standard aluminium extrusions available from DIY stores; the diagram of Fig. 5 shows how the panel can be assembled and covered with red filter plastic to improve the readability.

## Character Control

To scroll information from a standard computer port into the shift registers, the clock line should be connected as the MSB (the order of the other bits depends on the character generating patterns). Whatever binary number (less than 128) is required on the LED column should then be output to the port, ie

POKE PORT, number

Since the MSB of an eight bit binary number less than 128 is 0, the clock line is low. Then 128 is added to this number to toggle the shift clock, ie

POKE PORT, number +128

When the number is output again the lower seven bits remain unchanged but the MSB switches high, clocking the shift registers on one place. This process is repeated to scroll the message across the panel. Figure 4 gives a suggested character set based on the ASCII code (decimal equivalents shown beneath); however, we've replaced the hash symbol (# with a £ sign, which is probably more useful in this application. As an example of the procedure given above, to display the letter F the numbers 127, 9, 9, 1, 1 must be output from the port (corresponding to 1111111, 0001001, 0001001, 0000001, 0000001, the bit-patterns of the five columns making up the letter).

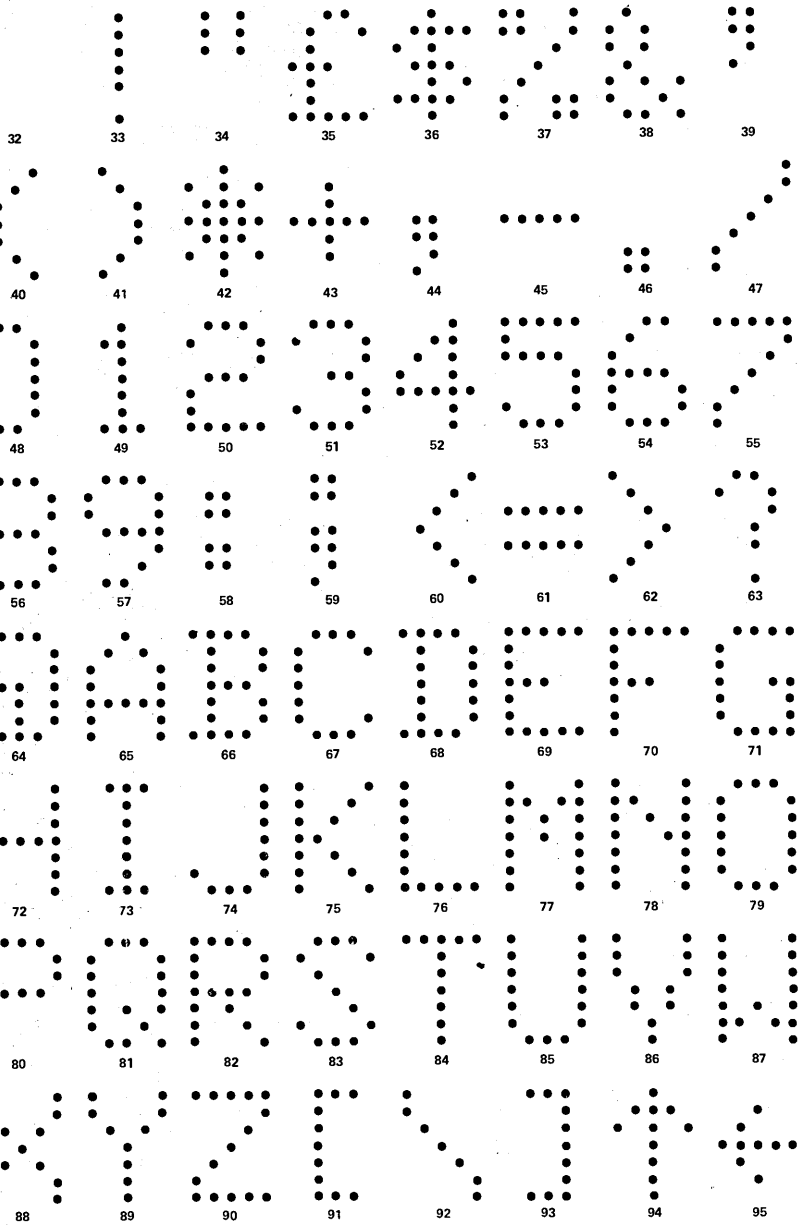



**Fig. 4 (Above) A suggested character set for this project. Although based on the ASCII standard set, we've replaced the hash sign (#) with a £ sign; when displaying messages it's probably more useful to have both £ and $ available.**

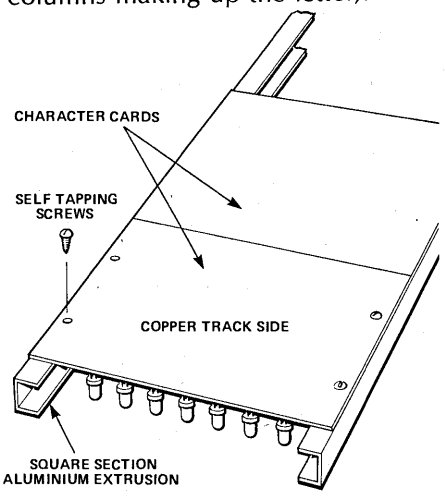



**Fig. 5 (Right) Assembly of the ETI Message Panel. The red filter plastic can be stuck to the other side of the aluminium extrusion.**

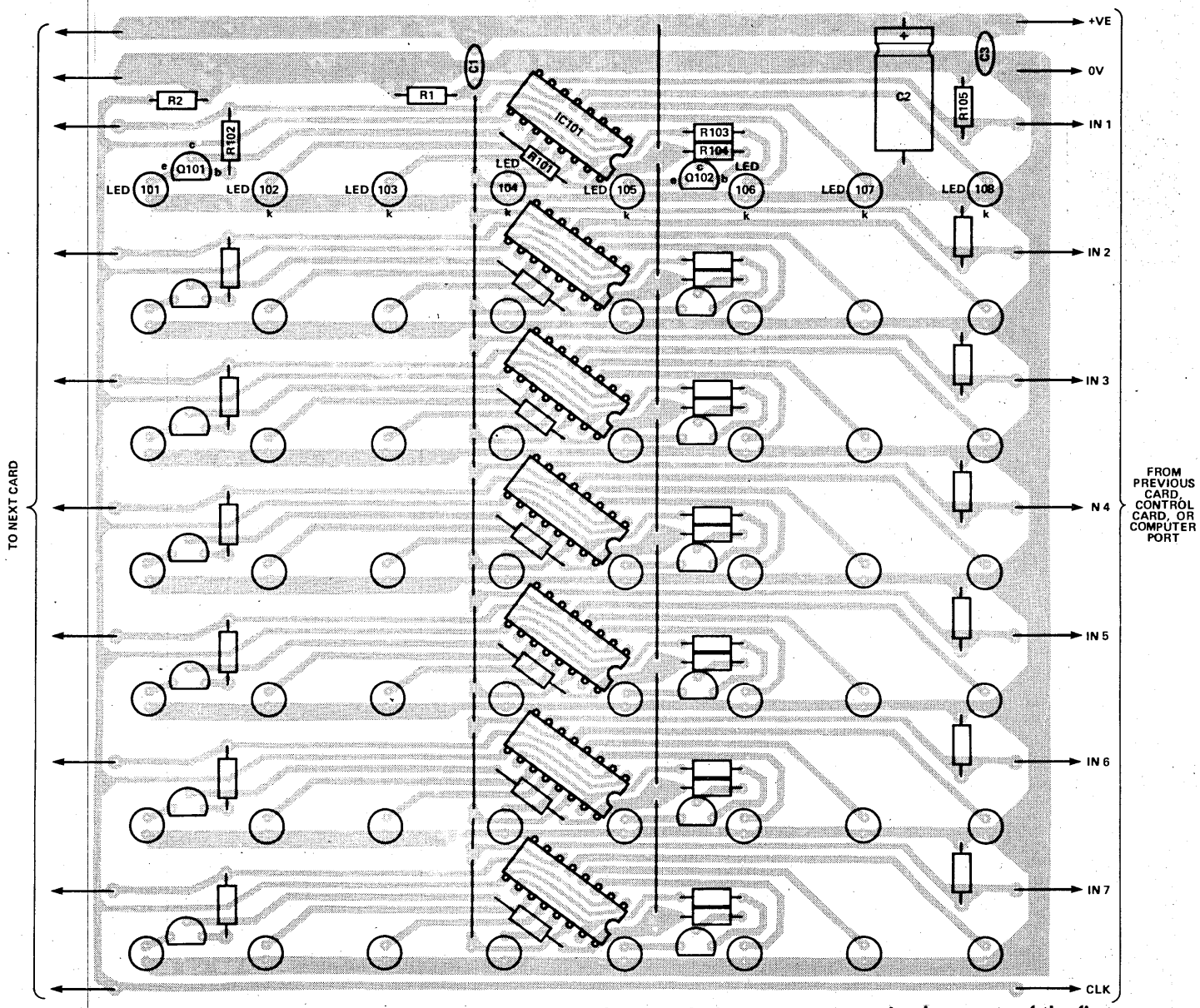
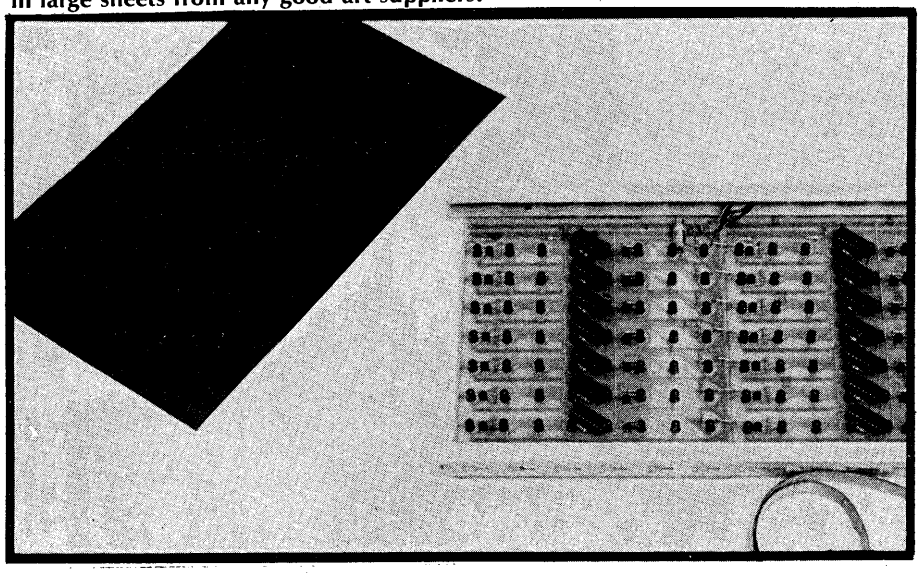



**Fig. 6 Component overlay for one Message Panel character card. The unmarked components are simply repeats of the first row.**

## PARTS LIST

| | |
|---|---|
| **Resistors (all ¼W, 5%)** | |
| R1 | 100k |
| R101, 104 | 680R |
| R102, 103 | 22k |
| R2, 105 | 1M0 |
| **Capacitors** | |
| C1, 3 | 1n0 ceramic |
| C2 | 220u 25V axial electrolytic |
| **Semiconductors** | |
| IC101 | 4015B |
| Q101, 102 | BC184L |
| LED101-108 | high efficiency 5 mm red LEDs (L53ID — see Buylines) |
| **Miscellaneous** | |
| PCB (see Buylines): aluminium extrusion for shelving systems; red filter plastic. | |

**The red filter plastic used to improve the contrast of the display should be obtainable in large sheets from any good art suppliers.**



# MESSAGE PANEL INTERFACE

## A handful of TTL, a small PCB and a socket, and you and your ZX computer can go for a scroll down memory lane. Design and development by Rory Holmes.

The interface described in this article supplements the Message Panel character cards featured last time. It allows any number of linked character cards to be driven from either the Sinclair ZX81 or the ZX Spectrum computer. Using the program published in this article, repeating messages drawn from a 7 × 5 ASCII character set can be displayed with a variable scroll speed across the message cards.

Essentially the interface is an eight-bit output port for the ZX expansion bus, whose logic levels are translated from TTL to the 9 V CMOS logic levels. The interface can be used as a universal CMOS output port for Sinclair computers (CMOS with supply voltage below 10 V).

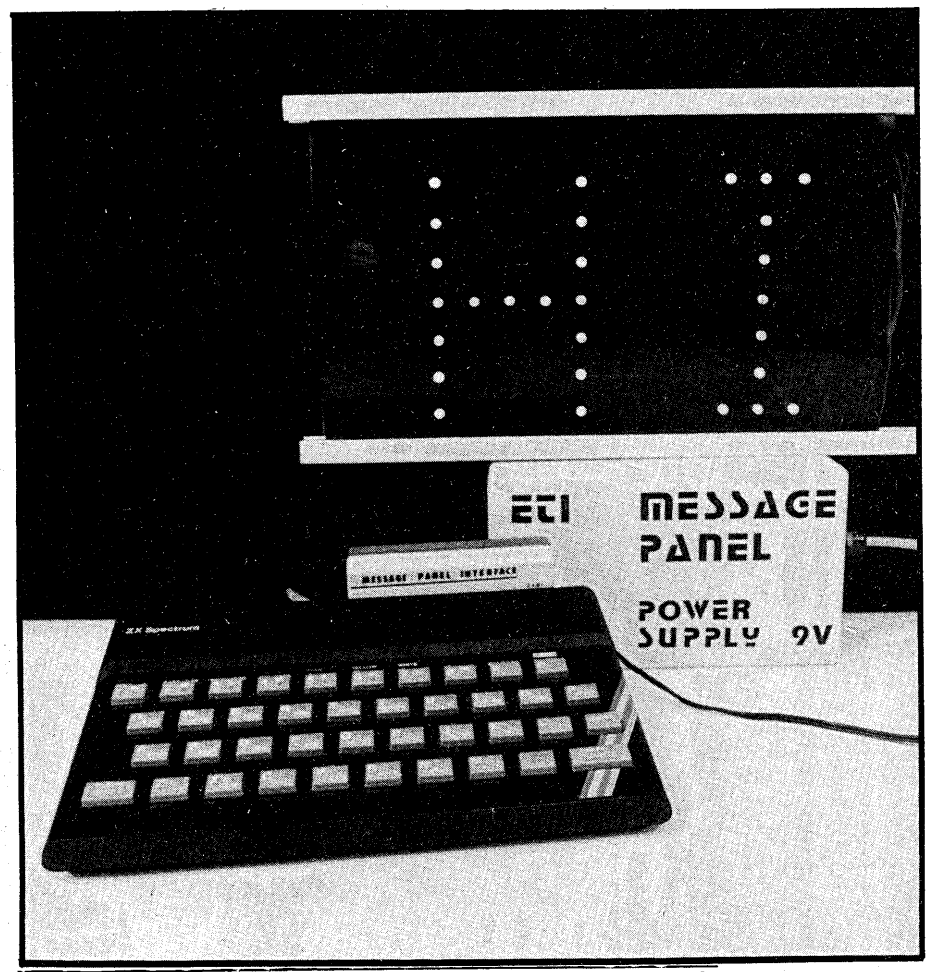
The port is memory mapped for the ZX81 and I/O mapped for the Spectrum; a DPDT slide switch is used to change between the two, though this may be replaced if desired by permanent links on the PCB.

In the interests of economy our circuit decodes only the top three address bits on the ZX81, placing the port over an 8K address range starting at 8192 decimal. Since this coincides with one of the 8K ROM 'echoes' a $\overline{\text{ROMCS}}$ signal is returned to the bus to de-select the ROM whenever the interface is addressed. Although the port responds to all the addresses in this 8K block it will not interfere with the basic memory or add-on RAM. With the Spectrum the port is I/O mapped at location 65503 and is accessible using the command OUT 65503,N. Again, it does not interfere with memory, and add-ons such as the printer may still be used.

### Boxing Clever

The interface is built into a two part Vero case as shown in the photographs. An edge-connector protrudes through the front to plug directly into the Sinclair expansion bus. Correct location is achieved for both computers with a polarising key at position three (on the connector, that is, not the bus). The wire-wrap type of edge-connector socket that we have used also allows an edge-connect expansion plug to be soldered directly to the back of the PCB against the protruding pins, allowing the ZX bus to follow through the interface box. Other add-ons can then be simply plugged in at the back. A 10-way cable, either ribbon or multi-core, connects the interface output to the first Message Panel character card.

The character cards have seven CMOS data inputs for the shift registers and one CMOS clock line. They all require logic drive signals compatible with 9 V CMOS logic levels. The clock line is treated as the most significant data bit to make up the eight-bit control bus which goes to the interface along with the positive supply rail and earth. These




**All wired up and ready to go — the complete Message Panel.**

supply lines provide the CMOS interface logic voltages. A sequence of eight-bit numbers can completely control the Message Panel.

In order to keep down the size and cost of the board, we have resorted to a compromise with the TTL-to-CMOS level translation. Using open collector TTL with pull-up resistors, a 0 to 7 V logic swing is produced to drive the 9 V CMOS. This works fine for the shift register data inputs (7 V being above the CMOS switching threshold), but isn't suitable for the CLOCK line, because it drives the clock inputs of all the shift registers in parallel. For this reason a one-transistor buffer is used on the eighth output bit to provide a sufficiently low impedance, full 9 V swing, for driving the clock line reliably.

## Construction

The interface is constructed on a PCB whose overlay is shown in Fig. 1. The PCB has been designed to take a 23-way double-sided edge-connector socket, which must have long wire-wrap pins to allow the connector to stand off from the board. The edge connector comes with a polarising key at position 3 and so must be fitted into the PCB the right way round. A 43-way connector could also be used, providing the unwanted ends are carefully sawn off so that the key remains in the same position. It's quite difficult to get all the edge connector pins through the PCB holes at once, but with the aid of a screwdriver and some patience it can be eventually be done. Before soldering the connector in place, ensure that it is square to the PCB, with the top edge being 21 mm from the component side of the board. This should leave the wire-wrap pins protruding about 2 mm from the track side of the board.

Fit the four links and IC sockets next, and then insert Veropins at all the connection points marked (they are also useful for connecting up to the slide switch). The vertical resistors can now be soldered in along with the other components, and finally the ICs can be plugged in, carefully observing their orientation. The DPDT slide switch should now be wired up with flying leads to the connection points as shown on the overlay.

Before final assembly the interface may be tested. Wire up the 0 V and 9 V Veropin connections to a PP3 9 V battery; this simulates the Message Panel power supply to determine the logic level output voltages. Now, simply plug the polarised edge connector into the back of your ZX81 or Spectrum computer. Switch on the computer, having first set the slide selector switch, and then monitor the data output pins with a voltmeter or scope. A logic low should be represented as 0 V while logic high will be about 6V8, but note that the data output pins are not in consecutive order.

In immediate mode, the OUT 65503, X command can be used for the Spectrum, and the POKE 8192, X command should be used on the ZX81. The value of X should appear as binary digits on the data outputs. For instance, POKE 8192,128 should result in the clock output pin going logic high and all the data pins going low. POKE 8192,3 will make the D1 and D2 outputs go to 6V8 (logic high), while all the other outputs remain at logic low.

A number of digital output patterns should be tested to ensure that all the output bits are working.

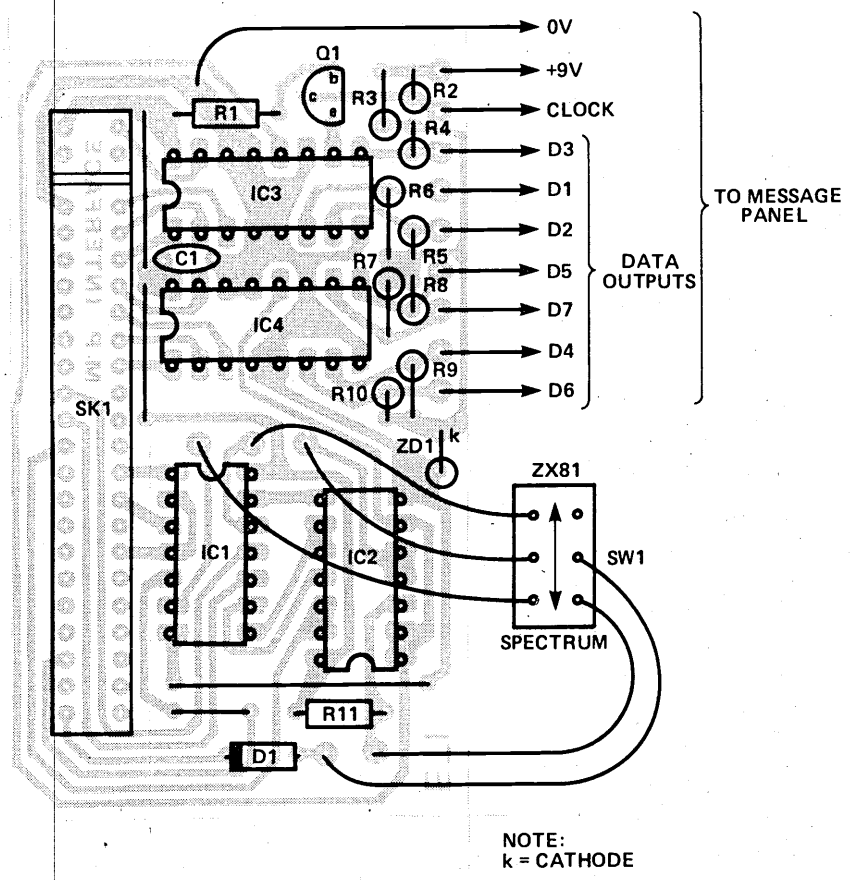



**Fig. 1 Component overlay for the Message Panel interface board.**

## BUYLINES

**Apart from the specialist hardware, nothing for this project should present undue problems. The 74LS TTL is now as common as dirt and can be obtained from such advertisers as Thames Valley Electronics, Cricklewood, Technomatic, Watford and Rapid Electronics. Wire-wrap edge connectors are sold by Timedata, 57 Swallowdale, Basildon, Essex, while the PCB can be obtained, as usual, from our PCB Service.**

## PARTS LIST

| | |
|---|---|
| **Resistors (all ¼W, 5%)** | |
| R1 | 10K |
| R2 | 1k0 |
| R3, 11 | 470R |
| R4-10 | 22k |
| **Capacitors** | |
| C1 | 10n ceramic |
| **Semiconductors** | |
| IC1 | 74LS27 |
| IC2 | 74LS00 |
| IC3, 4 | 74LS170 |
| Q1 | BC184L |
| D1 | 1N4148 |
| ZD1 | 6V8 400 mW zener |
| **Miscellaneous** | |
| SW1 | DPDT miniature slide switch |
| SK1 | 23-way double-sided edge connector with wire-wrap pins, key at position 3 for ZX81; 23-way double-sided edge plug for optional expansion (see Buylines) |

**PCB (see Buylines); Verocase (ref. 65-2514F); Veropins; nine-core screened cable.**

When satisfied with the operation of the interface, the PCB should be assembled into its case; it has been designed to fit a small two section Vero case, order ref. 65-2514F. The slide switch is mounted through an appropriate hole at one end, while the nine-way multi-core cable enters the box at the other end (see the internal photographs showing the PCB mounted in the case). The screen of the nine-way cable is used to carry the 0 V line to the message panel, and therefore if ribbon cable is used instead it should have 10 ways. The 10 outputs are all wired directly, via the cable, to the corresponding connections on the Message Panel. The case lid must have a rectangular hole cut to allow the edge-connector to protrude as far as possible through the front. The board should be raised from the base of the box on spacers, so that it is firmly sandwiched between the lid and the spacers when the case halves are screwed together; we used four large stick-on rubber feet as the spacers, to give a compression fit. After restraining the nine-core cable with a cable-tie or similar, it should be wired to earth and the nine terminal pins, preferably using colour coding for the data bits. A nine-way D-type cannon connector plug was found ideal for terminating the far end of the cable (make sure the earth passes through the plug screen).

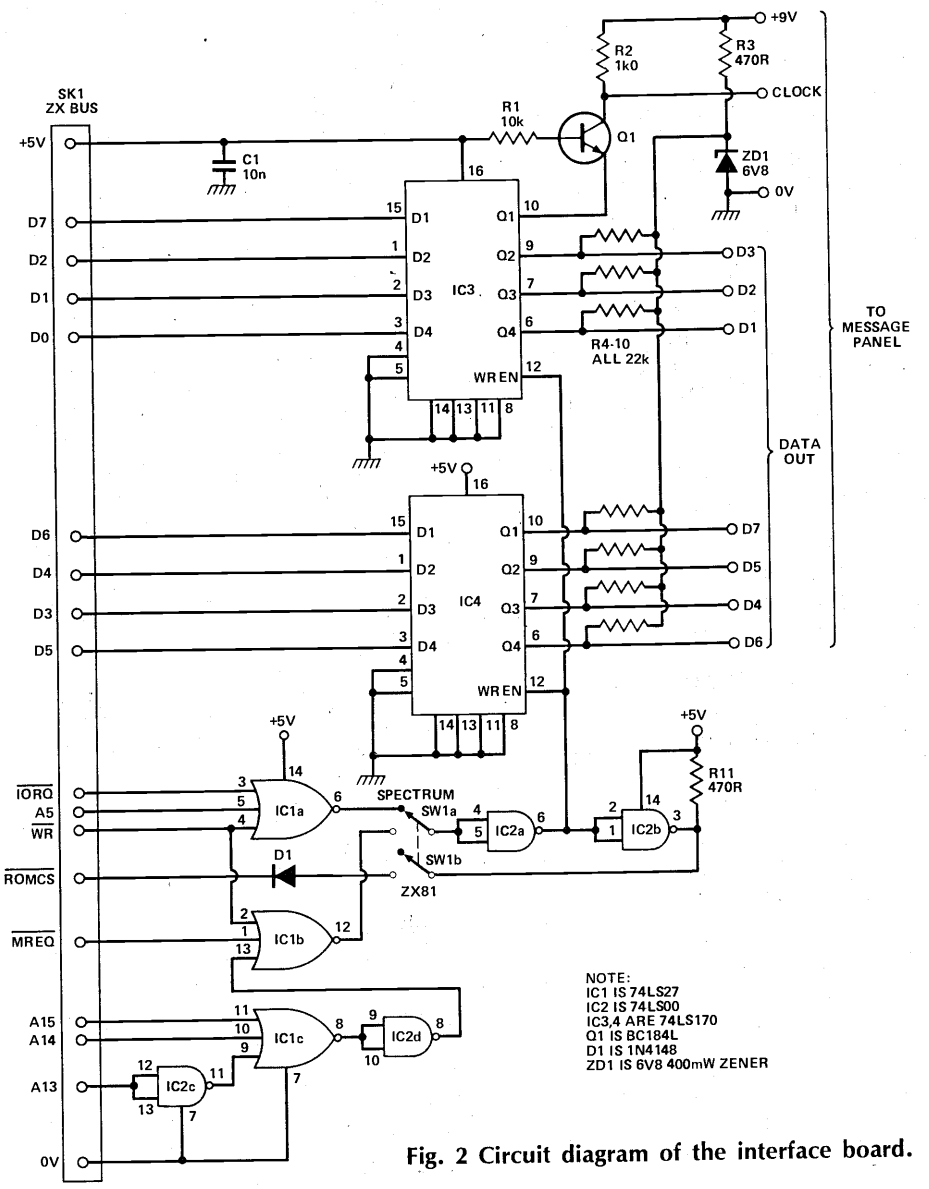



**Fig. 2 Circuit diagram of the interface board.**

## Setting Up The System

The system diagram of Fig. 3 shows how the separate parts are linked together to give a working Message Panel system. The 9 V power supply is the simple unregulated circuit shown last month. We mounted it in a separate box which plugs directly into the Message Panel via a two-way cable.

The program shown in Fig. 4 runs on a standard Sinclair Spectrum, and will allow messages entered on the keyboard to be displayed. It stores the character-generating data in a string array C$(7, 64) and uses this to display a message of any length entered in M$. The subroutine at line 3000 performs the data output and clocking of the message panel, with a variable scroll rate determined by the PAUSE statement. Subroutine 9000 sets up the character generator array C$ by using the Spectrum's ASCII characters to represent the data (this is done for convenience and to save memory space). The value returned for each character in

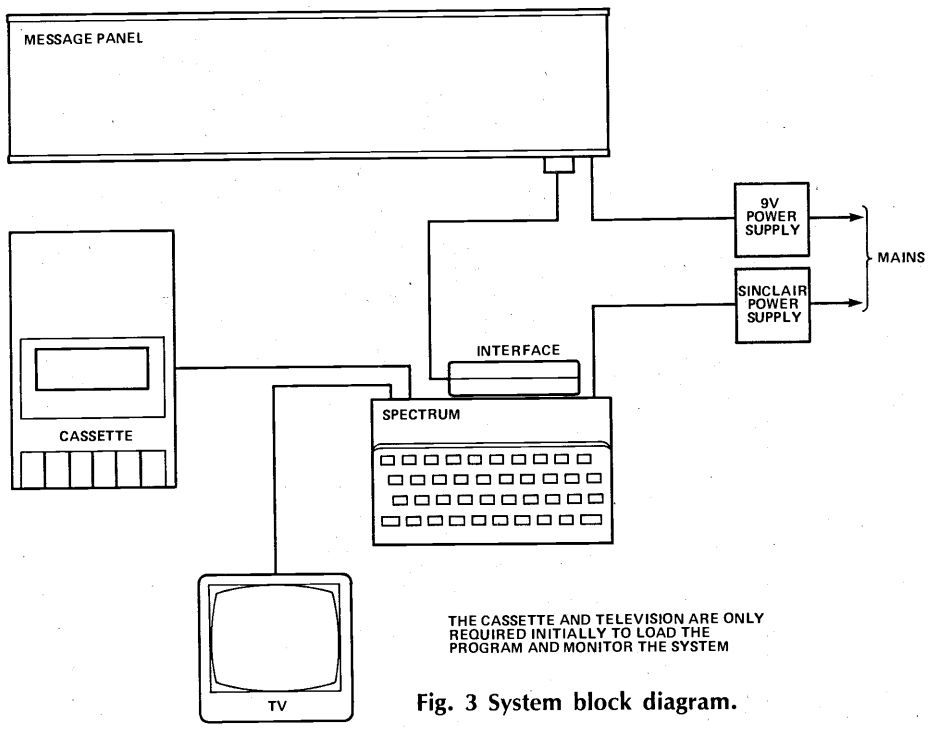



**Fig. 3 System block diagram.**

## HOW IT WORKS

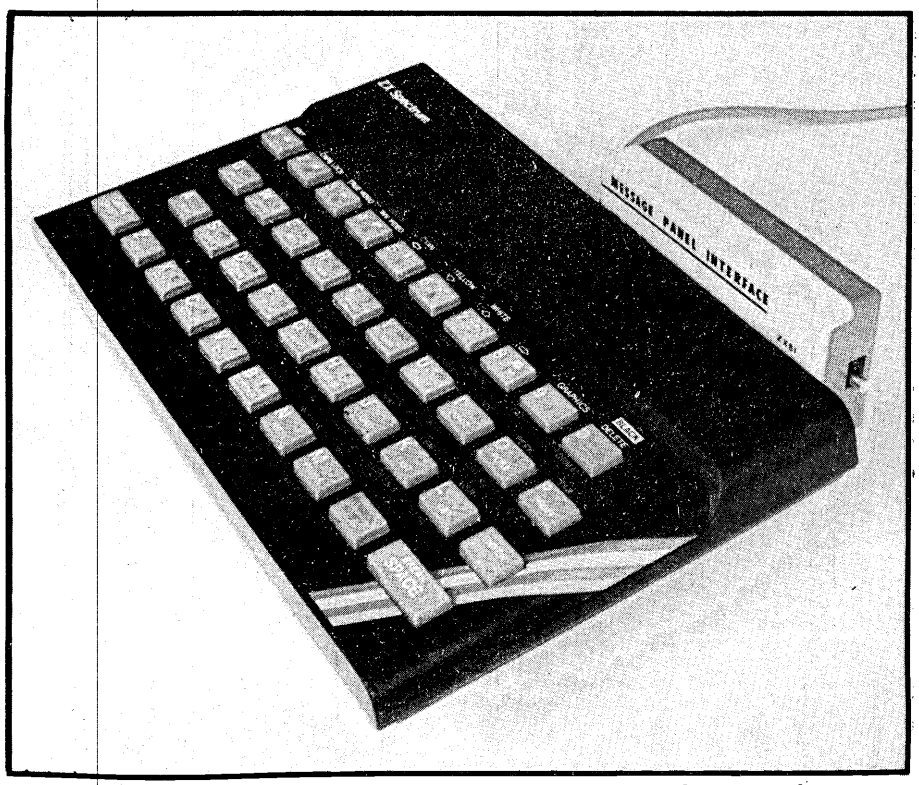
The port is essentially an eight-bit latch that will accept and store data from the ZX data bus as long as the address bus is sending the correct address to enable the latch. The latch is built from two four-bit registers wired in parallel, IC3 and IC4; they are actually 'four by four' register files (the 74LS170) of which only one four-bit latch is used. When the WRITE ENABLE input on pin 12 is taken low the latches will store the data on the D inputs; it then appears at the Q outputs.

The Q outputs are open collector types and they are used with pull-up resistors R4 to R10, thus increasing the available logic-level swing to match the CMOS inputs on the character cards. The maximum collector voltage allowed on the TTL Q outputs is 7 V and so the available swing is limited to 6V8 by the zener diode ZD1. R3 drops the 9 V CMOS positive rail voltage to supply the pull-up resistors. The logic swing of 0 to 6V8 is quite suitable for driving single CMOS inputs with up to 10 V on their positive rail supply (to achieve reliable switching from logic 0 to logic 1, CMOS needs an input of around 60% of the supply voltage). So the shift register data inputs will be reliably driven from these 'compromise' outputs. However, the CLOCK line on the message panel will need to drive every shift register IC that is being used; for this reason a single-transistor level-translator and buffer (Q1) has been used.

The address decoding is all achieved using IC1 (a triple three input NOR gate), and IC2 (quad NAND gates). Slide switch SW1 selects between memory-mapped decoding for the ZX81, via IC1b, or I/O mapped decoding for the Spectrum, via IC1a. When switched for the Spectrum the states of the bus lines IORQ (I/O request), A5 (address bit 5), and WR (write signal) are continuously monitored for logic lows. If they all go low together, then the Spectrum is performing an OUT 65503,X command, and the ouput of NOR gate IC1a will go high. This output is inverted by IC2a, which in turn enables the latches to store the data. An eight-bit data word can thus be updated into the latch at any time. The rest of the gates are effectively ignored, and as far as the Spectrum is concerned the A13, 14, and 15 inputs are connected to the wrong bus pins anyway.

When plugged into a ZX81, though, with the selector switch in its other position, these gates become usefully active. Here, address bits A14 and 15 must be logic low and A13 logic high to produce a high on the output of IC1c; this decodes the second 8K address region. The output of IC1c is inverted by IC2d and fed to one input of NOR gate IC1b. The other inputs monitor logic low states on the MREQ (memory request), and WR bus lines. Thus, IC1b's output will only go high when the ZX81 is performing a 'memory write' operation at an address location between 8192 and 16383. This output is fed to the latch enable via the selector switch and inverter as before. IC2b inverts and buffers the enable signal to drive the ROMCS line (linked up via SW1b). Whenever the interface is addressed this line will go high through diode D1 to switch off the 8K ROM.

The 70 mA or so of supply current for the TTL is taken directly from the 5 V supply rail on the ZX bus; C1 is included for on-board decoupling.




Like all good add-ons for the Spectrum, our interface simply plugs onto the expansion port.

```
  1 REM * MESSAGE PANEL PROGRAM
  2 REM
  5 LET P=65503: LET CLK=128
 20 DIM C$(7,64)
 30 GO SUB 9000
 90 INPUT "SCROLL SPEED 0.5 TO
5 ?";S
100 INPUT "YOUR MESSAGE ?";M$
130 GO SUB 1000
140 GO TO 130
1000 REM DISPLAY MESSAGE IN M$
1010 FOR K=1 TO LEN M$
1020 LET C=(CODE M$(K))-31
1030 FOR L=1 TO 7: LET B=CODE C$
(L,C)
1040 GO SUB 3000
1050 NEXT L
1060 NEXT K
1090 RETURN
3000 REM * OUTPUT PATTERN DATA
          TO PORT

3010 OUT P,B
3020 OUT P,B+CLK
3030 PAUSE S
3090 RETURN
9000 REM * INITIALIZE CHARACTER
          DATA

9010 LET C$(1)="              $<5          £
> r  '<a6      A 2!A>A@@>@   @@@@>
@>@   ? @<  a"
9020 LET C$(2)="    w*  I   A   X £
AB!A EJ II6[      I @A@I A A@ @   A
 A E  @    C@ A"
9030 LET C$(3)="    _   k V     >8 £
A@II EI II6;    Yy  IAAI A @@ @   A
 @ I@@£   XIA A@*"
9040 LET C$(4)="    w*d   A
>@II@EI I)          A IAAA I A@ @   A
 !)@  @      EA @"
9050 LET C$(5)="          cP
  F6 90 6  A     >!6 >A y@ ?A@@@>
 *!F   ?  @<  C  £"
9100 FOR I=6 TO 7: FOR J=1 TO 64
: LET C$(I,J)=CHR$ 0: NEXT J: NE
XT I
9200 DIM D(5): DIM N$(5,2,36)
9210 LET D(1)=31: LET D(2)=31: L
ET D(3)=26: LET D(4)=32: LET D(5
)=36
9220 LET N$(1,1)="ABCDHIJKLMNORT
UZ[\1†£;tuwz!j"@ ": LET N$(1,2)=
"@@@T@@@UH@H@@bXF@@@TB@bA@C@C@DH
"
9230 LET N$(2,1)="ABCFHIKLNPUX]†
_£bgilnoqsuwyz}@ ": LET N$(2,2)=
"@@GSG\NHHPT@HTbAR1HHBBIIAXTDDB\
"
9240 LET N$(3,1)="ACDFHIJKNPUX]†
_bgilnoqsxy}": LET N$(3,2)="@@@H
Gbb_HHRITTT@IHTLDIYPHH"
9250 LET N$(4,1)="ABCHJKLMNOPX[\
]†_£bgilnoquwyz?@ ": LET N$(4,2)
="@@G@\NH@H@D@@@bTHFRAHbBHIAXTDP
BH"
9260 LET N$(5,1)="ABCDEHIJKLMNOP
QRUXZ[\†_£dgjqtuwz!"@ ": LET N$(
5,2)="@@@TR@@@UH@H@C@@PC†@@T@@bA
@F bAGC@@DH"
9280 FOR I=1 TO 5: FOR J=1 TO D(
I)
9290 LET C$(I,(CODE N$(I,1,J)-64
)*2)=CHR$ (CODE N$(I,2,J)-64)
9294 NEXT J: NEXT I
9900 RETURN
```

Fig. 4 Spectrum display program.

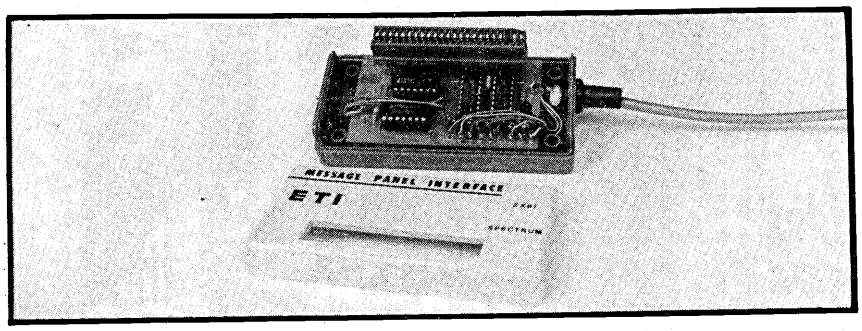
## TABLE 1

| CHARACTER | | COLUMN 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1 | | 0 | 0 | 0 | 0 | 0 |
| 2 | ! | 0 | 0 | 95 | 0 | 0 |
| 3 | " | 0 | 7 | 0 | 7 | 0 |
| 4 | # | 20 | 119 | 0 | 119 | 20 |
| 5 | $ | 36 | 42 | 107 | 42 | 16 |
| 6 | % | 99 | 19 | 8 | 100 | 99 |
| 7 | & | 54 | 73 | 86 | 32 | 80 |
| 8 | ' | 0 | 7 | 7 | 0 | 0 |
| 9 | ( | 0 | 28 | 34 | 65 | 0 |
| 10 | ) | 0 | 65 | 34 | 28 | 0 |
| 11 | * | 21 | 14 | 31 | 14 | 21 |
| 12 | + | 8 | 8 | 62 | 8 | 8 |
| 13 | , | 0 | 88 | 56 | 0 | 0 |
| 14 | - | 8 | 8 | 8 | 8 | 8 |
| 15 | . | 0 | 96 | 96 | 0 | 0 |
| 16 | / | 96 | 16 | 8 | 4 | 3 |
| 17 | 0 | 62 | 65 | 65 | 62 | 0 |
| 18 | 1 | 0 | 66 | 127 | 64 | 0 |
| 19 | 2 | 114 | 73 | 73 | 73 | 70 |
| 20 | 3 | 34 | 65 | 73 | 73 | 54 |
| 21 | 4 | 24 | 20 | 18 | 127 | 16 |
| 22 | 5 | 39 | 69 | 69 | 69 | 57 |
| 23 | 6 | 60 | 74 | 73 | 73 | 48 |
| 24 | 7 | 97 | 17 | 9 | 5 | 3 |
| 25 | 8 | 54 | 73 | 73 | 73 | 54 |
| 26 | 9 | 6 | 73 | 73 | 41 | 30 |
| 27 | : | 0 | 54 | 54 | 0 | 0 |
| 28 | ; | 0 | 0 | 59 | 0 | 0 |
| 29 | < | 0 | 8 | 20 | 34 | 65 |
| 30 | = | 20 | 20 | 20 | 20 | 20 |
| 31 | > | 65 | 34 | 20 | 8 | 0 |
| 32 | ? | 2 | 1 | 89 | 6 | 0 |
| 33 | @ | 50 | 49 | 121 | 65 | 62 |
| 34 | A | 124 | 18 | 17 | 18 | 124 |
| 35 | B | 65 | 127 | 73 | 73 | 54 |
| 36 | C | 62 | 65 | 65 | 65 | 34 |
| 37 | D | 65 | 127 | 65 | 65 | 62 |
| 38 | E | 127 | 73 | 73 | 65 | 65 |
| 39 | F | 127 | 9 | 9 | 1 | 1 |
| 40 | G | 62 | 65 | 65 | 73 | 121 |
| 41 | H | 127 | 8 | 8 | 8 | 127 |
| 42 | I | 0 | 65 | 127 | 65 | 0 |
| 43 | J | 32 | 64 | 64 | 64 | 63 |
| 44 | K | 127 | 8 | 20 | 34 | 65 |
| 45 | L | 127 | 64 | 64 | 64 | 64 |
| 46 | M | 127 | 2 | 12 | 2 | 127 |
| 47 | N | 127 | 2 | 4 | 8 | 127 |
| 48 | O | 62 | 65 | 65 | 65 | 62 |
| 49 | P | 127 | 9 | 9 | 9 | 6 |
| 50 | Q | 62 | 65 | 81 | 33 | 94 |
| 51 | R | 127 | 9 | 25 | 41 | 70 |
| 52 | S | 34 | 69 | 73 | 81 | 34 |
| 53 | T | 1 | 1 | 127 | 1 | 1 |
| 54 | U | 63 | 64 | 64 | 64 | 63 |
| 55 | V | 7 | 24 | 96 | 24 | 7 |
| 56 | W | 127 | 32 | 16 | 32 | 127 |
| 57 | X | 99 | 20 | 8 | 20 | 99 |
| 58 | Y | 3 | 4 | 120 | 4 | 3 |
| 59 | Z | 97 | 81 | 73 | 69 | 67 |
| 60 | [ | 0 | 127 | 65 | 65 | 0 |
| 61 | \ | 3 | 4 | 8 | 16 | 96 |
| 62 | ] | 0 | 65 | 65 | 127 | 0 |
| 63 | † | 4 | 2 | 127 | 2 | 4 |
| 64 | _ | 8 | 28 | 42 | 8 | 8 |

M$ is used to address a sub-string seven characters long in C$. The CODE value of each character in the sub-string corresponds to the binary pattern for each column of dots in a character on the character cards.

The program can easily be adapted for running on the ZX81 with extra RAM, but the character codes must be altered because the character set on the ZX81 does not have the same codes as the standard Spectrum ASCII character set. In the Spectrum program, a string array has been used for storing the pattern data merely to save space, and the string representation of this data must be changed to work with the ZX81's character codes. Table 1 provides all the eight bit numbers for each column of dots in each character of our standard character set, published last month. This table could be used directly as a numeric array or converted to a string array (as in C$(7, 64)) using the CHR$ equivalents.

The program can be altered in a variety of different ways; for example, so as to display messages stored in the program as literal strings and selectable on a menu




basis. Other character symbols are easily programmed in by adding more data. Inverse characters can be obtained by inserting the line:

3005 LET B = 127— B.

Once the program is in the computer and running, the television and cassette recorder can be dispensed with, allowing the computer to drive just the message panel in a continuous loop.

At first sight it might seem impossible to make the display scroll backwards; after all, the shift registers used in the character cards are not bidirectional and can only clock data from right to left. However, there's a cunning programming trick that can be used to overcome this problem. Suppose the message you're displaying has n characters. Each character is five columns wide and the program in Fig. 4 inserts a gap of two columns between letters. Hence each 'character' is effectively seven columns wide. If we now cycle through a modified 'message output' loop (GOSUB 3000) in the program 7n—1 times, and do it fast enough, the display will appear to have shifted back by one column. To output the clock pulses fast enough will probably require a machine code routine rather than the BASIC program listed here, but by sending a burst of clock pulses, followed by a pause, another burst, a pause and so on, the required effect should be obtained.

Incidentally, to hire a commercial display will cost you anything up to £250 per week!

# ZX81 MUSIC BOARD
## PART 1

**There have been a great many commercial and hobbyist designs for ZX81 peripherals, but we feel this one is something special. Full software listings will be given to help you use the board and the price is low. Design and development by M. P. Moore.**

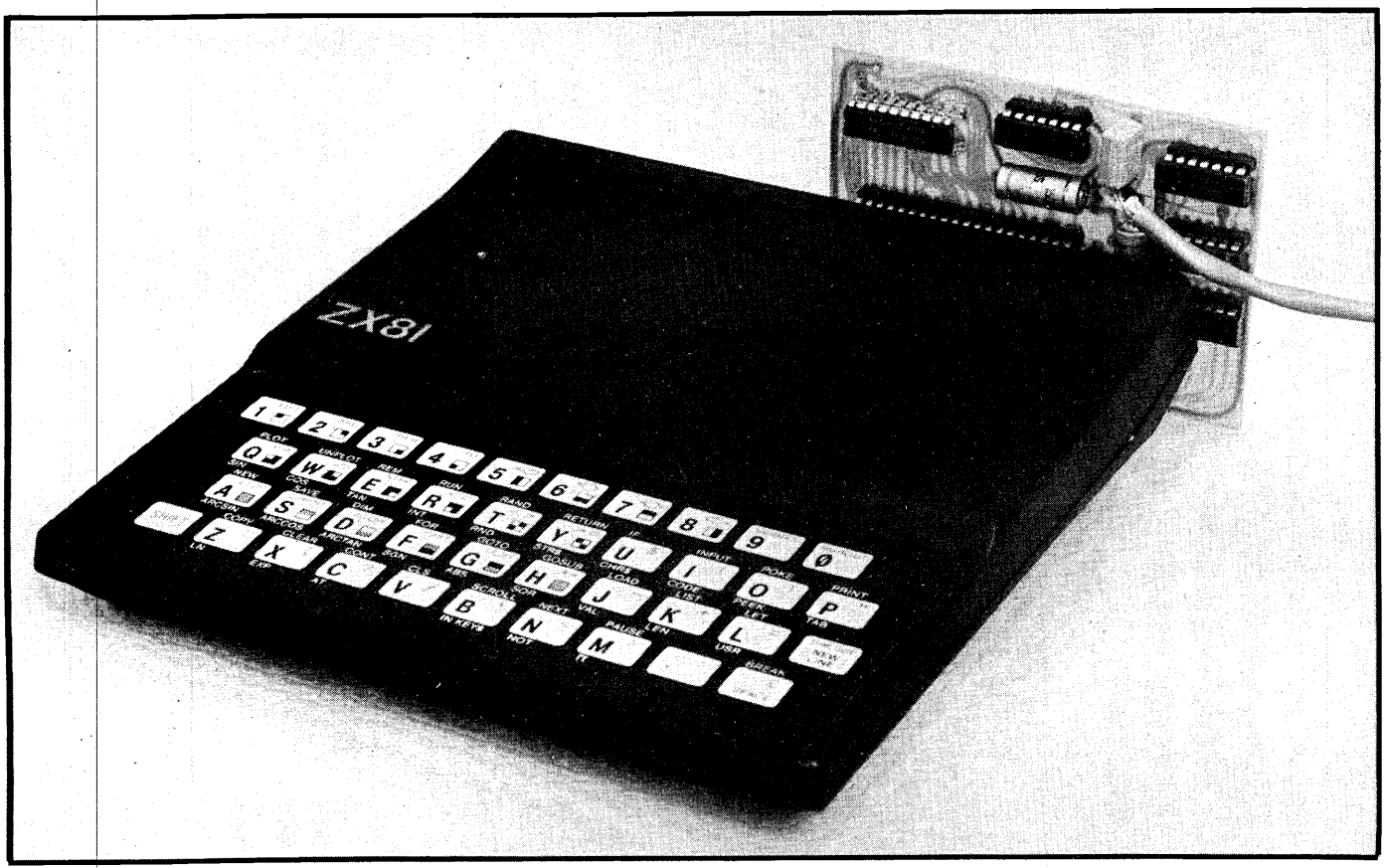
Give your space invaders program real 'zapp' — this add-on board enables you to hear those little green monsters being blasted away. Plug in the board, load the software cassette, and with two instructions you have a wide range of on-board sounds for your computer games; or you can copy music for your ZX81 to play, or devise your own sound effects for use in your own programs. You can also mix your own sound effects with the on-board sounds if you wish.

The unit is a sound generator with a fusible-link memory programmed with sounds varying from gunshots to spaceships, and with a basic octave of notes from which a range of seven or more octaves of music is obtained. When used with the software supplied it will bring ZX81 games to life with startling realism. The board will produce sounds with the basic 1K ZX81 but its full potential is realised with a 16K expansion, when the music program can provide a completely new use for those who are wondering what to do with their ZX81 now they have it.

A complete kit of parts is available (see Buylines), which also includes a comprehensive user's manual and software cassette. A demonstration cassette containing on-board sounds and music generated by the add-on sound board is available at an all-inclusive price of 95p. Petron Electronics have been good enough to grant us permission to publish both their PCB design and the complete software listings, including the PROM hex dump, to satisfy those diehard readers who insist on doing everything themselves. However, given the low price of Petron's kit, which contains all the hardware required plus documentation, we think that this is the best way to go for cost-effectiveness, ease of construction and convenience.

### Construction

All components in the circuit are mounted on a single-sided PCB (see overlay): IC sockets are supplied for all ICs. Two screened leads provide the connection from the PCB to your amplifier; all other connections to the board are made via an edge connector which plugs straight into the back of the ZX81




**The ZX81 music and sound effects board, like most other ZX peripherals, plugs directly into the computer.**

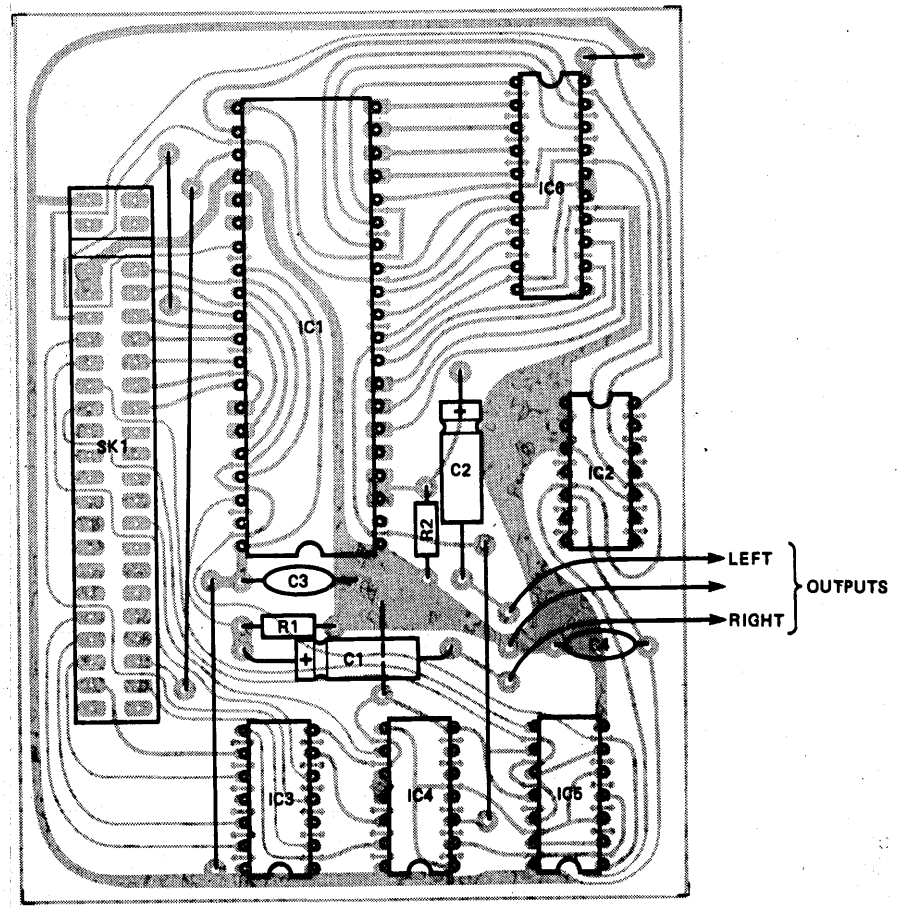



Fig. 1 Component overlay for the ZX81 sound board.

## PARTS LIST

| | |
|---|---|
| Resistors (all ¼W, 5%) | |
| R1, 2 | 1k2 |
| Capacitors | |
| C1, 2 | 100u 16 V axial electrolytic |
| C3, 4 | 100n polyester |
| Semiconductors | |
| IC1 | AY-3-8910 |
| IC2 | 74LS93 |
| IC3 | 74C20 |
| IC4 | 74C32 |
| IC5 | 74C02 |
| IC6 | TBP28L22N |

Miscellaneous
PCB; edge connector; IC sockets; two off phono plugs; 2 m of screened cable.

(or 16K RAM pack if used).

First of all solder the six IC sockets and then the six links: some of these are close to each other or to other components and the use of insulated wire is recommended. Now solder resistors R1 and R2 — these resistors can be of any value between 1k0 and 1k8. Solder the electrolytic capacitors C1 and C2, taking care to mount them the right way round (see overlay), and then capacitors C3 and C4. Finally, carefully insert and solder the edge connector leaving a gap of approximately 7 mm between the connector and the PCB. The pin corresponding to 9 V on the connector is not required and, for safety purposes, has been cut. Now carefully check all your soldered joints, preferably with a magnifying glass, and make sure that there are no bridges across any of the tracks.

If the board is to be used with a stereo amplifier, cut the length of screened cable supplied in half and solder the inner cores to one end to the left and right outputs, and connect the outer cores (screen) to the point marked GND. Take care to insulate these wires so that they will not short across other component leads. If you wish to use the board with a mono amplifier, connect a wire link between the two outputs and to this link connect the inner core of one of end of the screened cable, taking the screen to 0 V and insulating the cable as before. Connect the phono plugs (or one of them if you are using a mono amplifier) to the other end of the screened cable.

Now, carefully checking the orientation of the ICs, insert them into the IC sockets. Note that IC2 and IC6 are mounted in the opposite direction to the other ICs. With your ZX81 switched *off,* carefully plug the board into the back of the ZX81. If you have a 16K RAM pack, plug this on *first:* the sound board will plug onto the back of your RAM pack. Switch on your ZX81 and wait for the inverse K prompt to appear on your screen.

### On-Board Sound Program

This program enables you to include the on-board sounds listed in Table 2 in your own programs. To use these sounds all you have to do is to load the first short program from the software cassette and connect up your amplifier, keeping the volume fairly low. The following program will allow you to review the range of principal on-board sounds available before incorporating them in your own programs.

```
10  PRINT "SOUND NO.?"
20  INPUT S
30  POKE 16531,S
40  RAND USR 16514
50  CLS
60  GOTO 10
```

In order to run this program type **GOTO 10.**

The computer will now ask you the number of the sound you wish to hear: **SOUND NO. ?** As an example, type **153 NEWLINE.** The computer will repeat this question after each sound. A continous sound (eg helicopter) must be silenced by typing 0 or another sound number.

In order to use these sounds in your own programs, enter your program without altering line 1 of program "S". At each point in your program where you require a sound to be generated, you simply include the following program lines:-

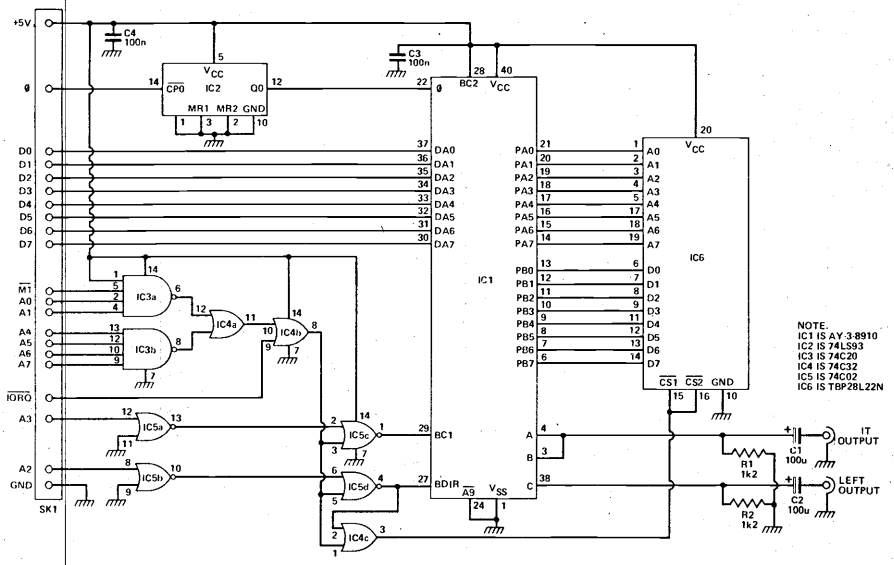



Fig. 2 Complete circuit diagram for the ZX81 sound board.

## HOW IT WORKS

IC6 is a fusible-link read-only memory (PROM) programmed with the data for all the on-board sounds and a basic octave of notes for music. This memory is accessed through the ports on IC1.

IC1 is a programmable sound generator (PSG), an AY-3-8910 which can be programmed to generate a wide range of sounds. Once data is written to this chip it produces and maintains the sound without continuous CPU maintenance, thus making it ideal for use with computer programs.

The PSG has three analogue outputs: outputs A and B are connected directly together and, via C1, connect to one channel of your amplifier; output C is connected via C2 to the other amplifier channel. The board will, therefore, give a dual image effect when used with a stereo amplifier. If you wish to use a mono amplifier, the analogue output C is connected directly to A and B.

IC3 and IC4a are used as an address decoder: the output of IC4a will be logic 0 when address lines A0, A1 and A4 - A7 are 1; $\overline{M1}$ must also be logic 1. IC4b is used to provide a chip select signal for the PSG only when the Input/Output request ($\overline{IORQ}$) is at logic 0. Thus the output of IC4b will be 0 only when a read or write operation on the PSG is to be performed. Whenever the output of IC4b is logic 1, the outputs of IC5c and IC5d will be 0, BC1 aand and BDIR will both be 0, and the PSG will be in the inactive state: see Table 2. (Since whenever it is deselected the 'inact' signal is sent, it is not necessary for the ZX81 program to send 'inact' to the PSG.)

IC5a and IC5b are used, together with IC5c and IC5d, to provide the necessary combinations of 0 and 1 for BC1 and BDIR. The output of IC4c drives the fusible-link PROM chip select input: this is to minimise the possibility of data bus contention between the PSG and the PROM should PSG port D accidentally be programmed as an output port, since IC4c output will only be 0 during a PSG read cycle.

The maximum clock frequency to the PSG is 2 MHz. IC2 is a low power Schottky version of the 7493 counter and is used here to divide the ZX81 clock frequency by 2.

## TABLE 1

| BC1 | BDIR | FUNCTION |
|---|---|---|
| o | o | INACTIVE |
| 0 | 1 | WRITE TO PSG |
| 1 | 0 | READ FROM PSG |
| 1 | 1 | LATCH ADDRESS |

## EPROM DATA

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 1F | 07 |
| 10 | 10 | 10 | FF | 28 | 09 | 69 | 00 |
| 00 | 3B | 00 | 00 | 10 | FF | 32 | 08 |
| 96 | 02 | C8 | 02 | 64 | 02 | 0F | 00 |
| 10 | 10 | 10 | 3C | 00 | 08 | 00 | 32 |
| 03 | 0F | 00 | 10 | 10 | 10 | 3C | 00 |
| 08 | 07 | 10 | 10 | 10 | FF | 05 | 09 |
| 07 | 08 | 10 | 10 | 96 | 03 | 08 | 07 |
| 10 | 00 | 10 | FF | 0C | 0F | BE | 00 |
| BE | 00 | BE | 00 | 00 | 38 | 10 | 10 |
| 10 | 00 | 01 | 09 | 5F | 00 | 5F | 00 |
| 5F | 00 | 00 | 38 | 10 | 10 | 10 | 00 |
| 01 | 09 | 2F | 00 | 2F | 00 | 2F | 00 |
| 00 | 38 | 10 | 10 | 10 | 00 | 01 | 09 |
| 17 | 00 | 17 | 00 | 17 | 00 | 00 | 00 |
| 10 | 10 | 10 | 00 | 03 | 09 | 19 | 00 |
| 32 | 00 | 41 | 00 | 1E | 00 | 0A | 0A |
| 0A | 6E | 00 | 6E | 00 | 00 | 00 | 09 |
| 10 | 0C | 0F | 10 | 96 | 03 | 08 | 1F |
| 07 | 10 | 10 | 10 | FF | 64 | 09 | 5A |
| 01 | 5A | 01 | 5A | 01 | 00 | 38 | 0F |
| 0F | 0F | 05 | 01 | 05 | 01 | 05 | 01 |
| 00 | 38 | 10 | 10 | 10 | 00 | 19 | 09 |
| FA | 03 | 00 | 00 | 03 | 05 | 10 | 0A |
| 10 | FF | 01 | 0C | A0 | 01 | 64 | 01 |
| 96 | 00 | 0F | 30 | 10 | 10 | 10 | 32 |
| 02 | 09 | FF | 3F | 10 | 10 | 10 | FF |
| 32 | 08 | 5C | 0F | 70 | 0E | A0 | 0D |
| DC | 0C | 28 | 0C | 28 | 0C | 68 | 0B |
| D2 | 0A | 46 | 0A | 9A | 09 | 22 | 09 |
| 22 | 09 | A0 | 08 | 28 | 08 | AE | 07 |

## PROGRAM 'S'

```
1       REM (Our machine code).

Machine code at line 1:
3E 07 D3 FF 3E 78 D3 F7
21 B4 40 E5 01 FB 0E C5
16 xx 3E 0E D3 FF 7A D3
F7 14 3E 0F D3 FF ED A2
20 F0 C1 E1 16 00 7A D3
FF 5E 7B D3 F7 14 10 F5
C9 xx xx xx xx xx xx xx
xx xx xx xx xx xx xx
```

**POKE 16531, x**
**RAND USR 16514**

where x is the number of the sound required from table 2.

The sound POKEd to 16531 remains the same until changed. Therefore, if you wish to repeat the same sound, there is no need to repeat **POKE 16531,x** — all you need to do is repeat the line **RAND USR 16514.**

Fast repetition of single sounds can be used to give a different effect. For example, the following program uses the rifle shot (sound 50) to generate a machine gun sound:

```
70   POKE 16531,50
75   FOR D=1 TO 40
80   RAND USR 16514
85   PAUSE 1
90   NEXT D
```

Now type **GOTO 70** and the computer will generate a burst of machine gun fire.

Table 2 gives the principal sounds that may be obtained, but there are many other interesting sounds which you can find by experimenting with other numbers not listed in this table.

## TABLE 2

| Sound N | Description | Continuous? | Sound N | Description | Continuous? |
|---|---|---|---|---|---|
| 0 | Silence | — | 92 | Mid blip | No |
| 8 | Cannon fire | No | 106 | High blip | No |
| 9 | Pistol shot | No | 204 | Musical blip | No |
| 50 | Rifle shot | No | 57 | Steam engine | Yes |
| 64 | Missile | No | 145 | Steam engine with whistle | Yes |
| 18 | Sonar | Yes | 167 | Train horn lower note | Yes |
| 153 | Explosion | No | 178 | Train horn upper note | No |
| 190 | Helicopter | Yes | 32 | Propellor aeroplane | Yes |
| 28 | Fog horn | Yes | 39 | Jet plane on the ground | Yes |
| 29 | Fog horn | Yes | 134 | Jet plane flying | Yes |
| 21 | Compressor | Yes | 52 | Mechanical hammer | Yes |
| 99 | Waterfall | Yes | 49 | UFO | Yes |
| 101 | Waterfall | Yes | 131 | UFO | Yes |
| 121 | Low bong | No | 213 | UFO | Yes |
| 33 | Mid bong | No | 214 | UFO | No |
| 45 | High bong | No | 231 | UFO | Yes |
| 78 | Low blip | No | | | |

# ZX81 MUSIC BOARD

## PART 2

**With the circuit and construction covered in the last article, we now turn to the software routines that enable you to use this project to the full. Design and development by M. P. Moore.**

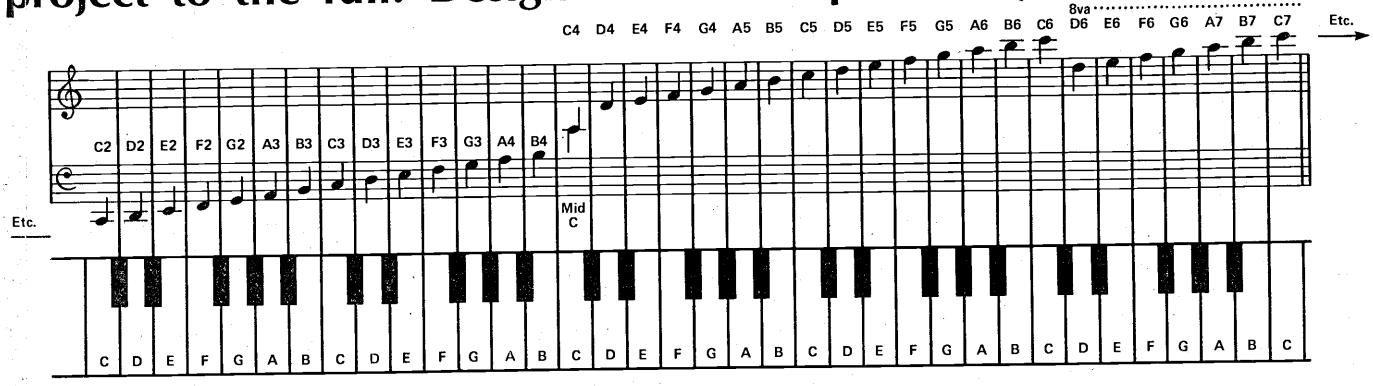



**Fig. 1 Table of notes as used by the input routine.**

## THE MUSIC PROGRAM

```
1     REM (our machine code)
2     CLS
5     DIM H(7)
10    FOR N = 1 TO 7
15    LET H(N) = 76
20    NEXT N
25    PRINT "CLEAR MUSIC SPACE
      (Y OR N) OR PLAY (P) OR
      LIST (L)?"
30    IF INKEY$ = "N" THEN
      GOTO 55
31    IF INKEY$ = "P" THEN
      GOTO 200
33    IF INKEY$ = "L" THEN
      GOTO 350
35    IF INKEY$ < >"Y" THEN
      GOTO 30
36    FAST
40    FOR N = 16670 TO 21670
45    POKE N,255
50    NEXT N
51    SLOW
55    PRINT "SHARPS?";
60    LET Z = 78
65    GOSUB 1150
100   PRINT "FLATS?";
105   LET Z = 74
110   GOSUB 1150
136   CLS
137   PRINT "EDIT FROM LINE
      NO.";
140   INPUT Z
141   PRINT Z
142   PRINT AT 21,0;Z; " [3 SPC]";
143   LET X = 0
144   FOR D = (Z - 1)*6 + 16670
      TO 21670 STEP 2
148   SLOW
149   GOSUB 1000
150   IF N$< >"5" THEN GOTO
      155
151   LET Z = Z - 2
152   LET D = D - 8
153   LET X = 2
154   GOTO 170
155   IF N$ = "R" THEN GOTO
      510
156   IF N$ = "E" THEN GOTO
      136
157   IF N$ = "P" THEN GOTO
      200
158   IF N$< >"0" THEN GOTO
      163
159   POKE D,0
160   POKE (D + 1),0
161   PRINT "[6 SPC]";
162   GOTO 170
163   IF N$ = "L" THEN GOTO
      350
164   POKE D,N
165   POKE (D + 1),H
170   GOSUB 2000
185   NEXT D
200   CLS
201   PRINT "SET VOLUMES (Y OR
      N), EDIT (E)", "OR LIST (L)?"
202   SLOW
205   IF INKEY$ = "Y" THEN
      GOTO 215
206   IF INKEY$ = "E" THEN
      GOTO 136
207   IF INKEY$ = "L" THEN
      GOTO 350
208   IF INKEY$ = "N" THEN
      GOTO 265
210   GOTO 205
215   PRINT "A=";
220   INPUT S
225   POKE 16540,S
230   PRINT S; "B=";
235   INPUT S
240   PRINT S;" C=";
245   POKE 16541,S
250   INPUT S
255   POKE 16542,S
260   PRINT S
265   PRINT "SPEED?"
270   INPUT S
271   PRINT "PLAY FROM LINE
      NO.?"
272   INPUT D
275   LET D = (D-1)*6+16670
276   CLS
280   RAND USR 16514
281   FAST
282   IF PEEK D = 255 THEN
      GOTO 299
283   POKE 16561, (PEEK D)
284   LET D = D + 1
285   POKE 16562, (PEEK D)
286   LET D = D + 1
287   POKE 16563, (PEEK D)
288   LET D = D + 1
289   POKE 16564, (PEEK D)
290   LET D = D + 1
291   POKE 16565, (PEEK D)
292   LET D = D + 1
293   POKE 16566, (PEEK D)
294   LET D = D + 1
295   RAND USR 16543
296   PAUSE S
297   POKE 16437,22
298   GOTO 282
299   LET X = 16561
300   FOR D = 1 TO 6
305   POKE X,0
306   LET X = X + 1
307   NEXT D
308   SLOW
310   RAND USR 16543
315   GOTO 200
350   CLS
351   PRINT "LIST FROM LINE
      NO.";
355   INPUT Z
360   PRINT Z
365   PRINT AT 21,0;Z"[3 SPC];
366   FAST
370   LET X = 0
371   LET K = (Z - 1) * 6 + 16734
372   FOR D = (Z - 1) * 6 +
      16670 TO (Z - 1) * 6 +
      16734 STEP 2
375   LET Y = 1
380   LET L = PEEK D
385   LET M = PEEK (D + 1)
386   IF L = 255 AND M = 255
      THEN PRINT " STOP "
387   IF L = 255 AND M = 255
      THEN GOTO 491
388   IF L = 0 AND M = 0 THEN
      PRINT "[3 SPC];
389   IF L = 0 AND M = 0 THEN
      GOTO 470
390   LET L = L + M * 256
395   IF L< 1966 THEN LET Y = Y
      + 1
400   IF L< 1966 THEN LET L = L
      * 2
405   IF L< 1966 THEN GOTO 395
410   LET M = L/256
415   LET M = INT M
```

The music program allows the ZX81 to play up to three notes simultaneously. The range is from A octave 1 upwards, where middle C is C4 (see Fig. 1). There is sufficient memory space with a 16K expansion to enter 833 chords of music. Everything possible has been done to facilitate the entering of music. The key signature (sharps or flats) is set to begin with and remains set until changed; changes may be made during the entering of music; each of the three channels has an independently set volume; the same note repeated on one channel will give a continuous note, but if played on alternating channels, will give a repetitive note.

The symbols used are + (sharp), − (flat) and = (natural). The functions available are **EDIT,** which is used for entering and editing music already entered, and includes **BACKSPACE** and **REPEAT** functions; **LIST,** which allows you to read the music entered, and **PLAY.**

Program "M" is very long and takes about five minutes to load. Having loaded the program the sequence of operations is as follows:-

Type **GOTO 2 NEWLINE.**

The computer will ask: **CLEAR MUSIC SPACE (Y OR N) OR PLAY (P) OR LIST (L)?**

The second and third functions don't interest us at the moment, and since the music space is clear to start with, type **N.**

The computer now asks **SHARPS?**

If the key signature contains sharps, type them in (in any order) followed by **NEWLINE.**

If there are no sharps type **0 NEWLINE.**

The computer now asks **FLATS?**

Deal with the question as for sharps.

The computer asks **EDIT FROM LINE NO.**

Since you are starting from scratch enter **1 NEWLINE.**

The computer is now ready to accept up to 833 'lines' of music. A

## BUYLINES

**Since this article was first published, the board has been through a series of upgrades and developments; these include the replacement of the fusible-link PROM with an EPROM and the removal of a memory map conflict with the ZX printer. Newtech Micro Developments Ltd, of 1 Courtlands Road, Newton Abbott, Devon can supply a programmed EPROM, revised PCB, software cassette and manual for the updated version for £11.95 including VAT and p&p.**

**If you want to make the original version, then you will have to make the PCB yourself as this version is now out of production. The PROM is available from the Midwich Computer Company Limited, Rockinghall House, Hinderclay Road, Rickinghall, Suffolk IP22 1HH, tel 0379-898751; you will, of course, have to program it before use (people do forget!).**

**Also available from Newtech is a converter that allows the board to be used with the Spectrum; this costs £6.50 inc VAT and p&p.**

## THE MUSIC PROGRAM

```
420   IF L>=256 THEN LET L=L-256
425   IF L>=256 THEN GOTO 420
430   POKE 16619,L
435   POKE 16620,M
440   LET L=USR 16608
445   LET L=L/4
450   LET M=INT L-19
454   LET M=INT M
455   IF M=37 AND INT L<>L THEN LET Y=Y-1
456   IF M=37 THEN LET M=44
460   PRINT CHR$ M;
465   IF INT L<L THEN PRINT "+";
466   IF INT L=L THEN PRINT "=";
468   PRINT Y;
470   PRINT "[3 SPC]";
471   IF D=K THEN GOTO 491
472   GOSUB 2000
490   NEXT D
491   SLOW
495   PRINT AT 0,0;
496   IF INKEY$="E" THEN GOTO 136
500   IF INKEY$="P" THEN GOTO 200
501   IF INKEY$="L" THEN GOTO 350
505   GOTO 496
510   PRINT "REPEAT FROM LINE NO.?"
511   SCROLL
515   INPUT C
516   LET E=D
517   FAST
520   FOR C=C TO (Z-1)
525   LET Q=(C-1)*6+16670
530   FOR E=E TO E+5
531   IF E=21670 THEN GOTO 200
535   POKE E,PEEK Q
540   LET Q=Q+1
545   NEXT E
550   PRINT "DITTO";
552   LET X=2
555   GOSUB 2000
560   NEXT C
561   LET D=(E-2)
562   SLOW
565   GOTO 185
1000  INPUT N$
1001  IF N$="R" OR N$="5" OR N$="0" OR N$="P" OR N$="L" OR N$="E" THEN RETURN
1002  IF N$(1)<"A" OR N$(1)>"G"THEN GOTO 1000
1005  IF LEN N$<2 THEN GOTO 1000
1008  FAST
1009  LET H=CODE N$-37
1010  IF LEN N$ 2 THEN GOTO 1030
1015  IF N$(2)="-" THEN LET H(H)=74
1020  IF N$(2)="+" THEN LET H(H)=78
1025  IF N$(2)="=" THEN LET H(H)=76
1030  LET N=CODE N$
1035  LET N=N*4
1036  IF N=160 AND H(H)=74 OR N=172 AND H(H)=74 THEN LET N=N-2
1040  LET N=N+H(H)
1045  POKE 16581,N
1050  LET N=USR 16567
1055  PRINT N$(1);
1057  IF H(H)=78 THEN PRINT "+";
1058  IF H(H)=76 THEN PRINT "=";
1059  IF H(H)=74 THEN PRINT "-";
1060  IF LEN N$=2 THEN LET H=CODE N$(2)
1065  IF LEN N$=3 THEN LET H=CODE N$(3)
1066  PRINT CHR$ H; "(3 SPC)";
1067  LET H=H-28
1070  LET H=H-H-H
1075  FOR H=H TO -1
1080  LET N=N/2
1085  NEXT H
1086  LET N=N*2
1090  LET N=INT N
1095  LET H=N/256
1100  LET H=INT H
1105  IF N>=256 THEN LET N=N-256
1110  IF N>=256 THEN GOTO 1105
1115  RETURN
1150  INPUT H$
1155  IF H$<"A" AND H$<>"0" OR H$>"G" THEN GOTO 1150
1160  IF H$="0" THEN RETURN
1165  FOR N=1 TO LEN H$
1170  LET H=CODE H$(N)
1175  LET H=H-37
1180  LET H(H)=Z
1185  NEXT N
1190  PRINT H$
1195  RETURN
2000  LET X=X+1
2005  IF X<3 THEN RETURN
2010  LET X=0
2015  LET Z=Z+1
2020  SCROLL
2025  SCROLL
2030  PRINT Z; "[3 SPC]";
2035  RETURN
```

Machine code at line 1:-

```
3E 07 D3 FF 3E 38 D3 F7
21 9C 40 01 08 03 79 D3
FF 0C 5E 7B D3 F7 23 10
F5 C9 xx xx xx 21 B1 40
01 00 06 5E 79 D3 FF 7B
D3 F7 23 0C 10 F5 C9 xx
xx xx xx xx xx 3E 07 D3
FF 3E 78 D3 F7 3E 0E D3
FF 01 FB xx 78 04 D3 F7
3E 0F D3 FF ED 58 3E 0E
D3 FF 78 D3 F7 3E 0F D3
FF ED 50 D5 C1 C9 3E 07
D3 FF 3E 78 D3 F7 06 FE
21 xx xx 7D E6 80 6F 3E
0E D3 FF 78 D3 F7 3E 0F
D3 FF DB FB E6 80 BD 28
05 05 05 18 EA 00 3E 0E
D3 FF 04 78 D3 F7 3E 0F
D3 FF DB FB BC 20 E9 05
48 06 00 C9
```

Approximately 5,000 memory positions are reserved here for music.

## HOW IT WORKS — MUSIC PROGRAM

Array H is a one-dimensional seven-position array which is used to keep a record of whether each note A - G is natural, sharp or flat. Lines 5 to 20 load the value for natural (76) into each position of array H. Lines 25 to 35, depending on the answer typed in, make the computer jump accordingly or continue from line 36. Lines 36 to 51 clear the memory space set aside for music by POKEing the stop code 255 to each memory location reserved. In conjunction with the BASIC subroutine at lines 1150 to 1195, lines 55 to 110 set the initial key signature by making the value of the appropriate position of H equal 78 for sharps and 74 for flats. Lines 137 to 142 make Z equal to the current line number for entering and editing music. Variable X is used to keep a record of which channel note you are currently entering. Line 144 sets up a loop using D where D starts with the address of the memory space corresponding to line number (Z) and ends with the value 21670, which is the last available music space. Line 149 calls the BASIC subroutine at line 1000. This subroutine inputs a key-press or series of key-presses which will either be note data, silence or one of the five available functions: REPEAT (R), BACKSPACE (5), PLAY (P), LIST (L) or EDIT (E). If silence or one of the functions is entered the computer returns to line 150.

If note data was entered the computer continues at line 1002 with a check that the data entered was in fact valid note data. Line 1009 makes variable H equal to the code of the note entered minus 37. If the data entered does not contain an appending sharp, flat or natural sign, line 1010 makes the computer jump to line 1030. Lines 1015 to 1025 adjust array H accordingly (sharp, flat or natural) depending on the second character of N$ (ie N$ (2)). Lines 1030 to 1040 load the variable N with the code of the note entered (ie N$(1)) and adjust this value together with corrective maths depending on the note (sharp, flat or natural) stored in array H to provide the address of the basic note data in the pre-programmed PROM. Lines 1045 to 1050 POKE this data to memory position 16581 and a machine code subroutine based at 16567 returns N with the basic tone period value of this note, ie the lowest octave. Lines 1055 to 1059 print the note and its sign on the screen. Lines 1060 to 1065 make variable H equal to the code of the octave number entered depending on whether N$ is two or three characters long. Line 1066 prints the octave. Lines 1067 to 1075 correct the value of H and set up a loop using H where H = H to − 1. This loop is used to divide the basic data in N by 2 (this has the effect of raising the note one octave each time N is divided by 2) until the correct tone period data is obtained. Line 1086 corrects the tone period value of N, otherwise it would be one octave too high. Lines 1095 to 1110 set H to equal the most significant byte of the note and N to equal the least significant byte.

Line 115 returns the computer from this subroutine at line 1001 because:-

1) BACKSPACE function (5) was entered; it runs through lines 151 to 154 adjusting the values of Z, D and X to effect a backspace. Lines 155 to 157 check to see if the computer was returned with functions R, E or P and if so, it jumps accordingly.
2) If 0 was entered, the computer runs through lines 159 to 162 entering silence in the current note position and printing spaces in that note position on the screen.
3) If L was entered, the computer jumps to line 350 to list data.
4) If P was entered, the computer jumps to line 200 to play the music.
5) If E was entered, the computer goes back to line 136 to restart the edit function.
6) If R was entered, the computer jumps to line 510.
7) If note data was entered, the computer runs through lines 164 and 165 which POKE the tone period data in the current memory position.

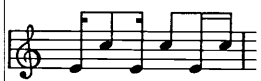
Line 170 calls a BASIC subroutine located at 2000. This subroutine simply checks whether or not the data just dealt with was the third note of a chord, and if so, scrolls the screen up two lines and prints a new line number before returning. Line 185 causes the computer to loop back to line 144 ready to enter the next string of music data. If all the available memory space were taken up, the computer would fall through line 185 and actuate the section of the program which plays music from line 200.

Had the R function returned the computer from the subroutine at 1002 the computer would have jumped to line 510 which puts the question FROM CHORD NO. ? Lines 511 to 565 perform a block copy of the lines specified and adjust the display accordingly. Lines 200 to 210 ask whether you want to set channel volumes, or edit or list. If the answer is E the computer jumps back to the EDIT program at line 136; if L is entered the computer jumps to the LIST program at line 350; if you didn't want to set the volumes so that the answer was N, the computer jumps to line 265. Lines 215 to 260 input volumes for the three channels A, B and C and POKE the volumes required to memory locations 16540, 16541 and 16542 respectively. Lines 265 to 270 input the value of the pause used in line 296 to regulate the speed at which music is played. Line 276 sets D to the address of the first memory position containing music data. Line 280 calls a machine code subroutine based at line 16514 which initialises the PSG for three channels of sound. This subroutine programs the PSG with the volumes held in memory locations 16540 to 16542. Line 282 checks the current music memory position for the STOP code (255) and if this position equals 255 it identifies this as the end of the music, and the computer jumps to 299. Lines 283 to 294 POKE to memory positions 16561 to 16566 the six bytes of tone period data for the next chord to be played.

It may be thought that memory space could have been saved by making lines 283 to 294 a loop. This proved, however, to have an unacceptable slowing effect on the maximum speed available (ie 0). Line 295 calls a machine code subroutine which relays the data in memory position 16561 to 16566 to the PSG, thus producing the next chord. Line 296 is the pause regulating the speed using the variable S and, since this program section is run in the FAST mode, line 297 POKEs Sinclair's obligatory 255 to memory position 16437. Line 298 causes the computer to jump back to line 282 to continue with the next chord. As we have seen, when the end of the music is reached, the computer jumps to line 299. Lines 299 to 307 load the silence value 0 to memory positions 16561 through 16566 and line 310 outputs this last set of data to the PSG using the above-mentioned machine code subroutine located at 16543. Line 315 then causes the computer to jump back to line 200.

The LIST (L) function starts at line 350. Lines 351 to 365 input and print the line number that is being listed which is held in variable Z. Variable X is used to keep a record of which channel data is being calculated. Line 371 sets the variable K to the memory address of the last note data to be listed in one full screen ( providing a STOP code 255 is not encountered first). Line 372 sets variable D up in a loop, where D starts with the value of the first music location to be listed and thereafter holds the current memory position of data being calculated. Variable Y is used to keep a record of the octave as it is being calculated in lines 395 to 405. Lines 380 to 385 set variables L and M to the value of the data for the current note being listed. Lines 386 and 387 check for the STOP code 255 in variables L and M, and if it is detected the computer prints STOP and jumps to line 491. Lines 388 and 389 check for silence (0) and if detected the computer prints spaces and then jumps to line 470. Line 390 sets variable L to the value of the complete tone period. Lines 395 to 405 calculate the octave in Y by multiplying the value of L by 2 until it is within range of the basic octave values (ie greater than or equal to 1966).

Lines 410 to 425 reconstitute this new data into variables L and M. At this point the number held in L and M will be the same as a number in the basic octave of notes in the pre-programmed PROM. Lines 430 to 435 POKE this data to memory positions 16619 and 16620. Line 440 calls up a machine code subroutine which returns with L set to the memory position of the PROM where the note data POKEd in lines 430 and 435 is to be found. Lines 445 to 455 reconstitute the value of L in variable M so that M contains the Sinclair code for the correct note A to G. Line 456 runs a check on variable M, so that if M = 37 (ie G sharp) the computer, rather than printing 9, prints G (code 44). Line 460 prints the note thus calculated. Line 465 checks the value of L to see if it is a whole number. If it is not, due to the maths in lines 445 to 455, the note will be a sharp and line 465 prints + (sharp). Line 466 likewise checks to see if L is a whole number, and if it is, prints the sign for natural (=). Line 467 checks for the note being G sharp and corrects the octave value in Y accordingly. Lines 468 to 470 print the octave and the next three spaces. When a screen-full of data has been listed, D will equal K. Line 471 checks for this and if D = K the computer jumps to line 491. As with the EDIT function (E), the LIST function uses the subroutine at line 2000 to keep the VDU display correct. When it has finished listing music the computer continues at line 491. At lines 495 to 505 the computer waits for a further command EDIT (E), PLAY (P) or LIST (L), and jumps accordingly.



THIS BAR SHOULD BE ENTERED AS FOLLOWS

| LINE No. | Ch1 | Ch2 | Ch3 |
|---|---|---|---|
| 1 | E4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 2 | C5 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 3 | C5 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 4 | E4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 5 | C5 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 6 | C5 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 7 | E4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 8 | C5 NEWLINE | 0 NEWLINE | 0 NEWLINE |

**Fig. 2 Calculation of the number of lines in a bar.**

'line' of computer music is a note (or silence) entered in each of the three channels A, B and C. Channels A and B are played through one audio output and C through the other if a stereo amplifier is used.

When copying music you must find the shortest note in the score: longer notes must be converted to the equivalent multiple of the shortest note to find the number of computer 'lines' in a bar (see Fig. 2). Notes 1, 3, 5 and 6 in the example are half the length of notes 2 and 4. The total number of computer 'lines' in the bar is therefore 8.

A note repeated on the same channel will sound as one continuous note. If the notes are to sound distinct, you must change from one channel to another (see Fig. 3).

Let us go back to our instruction **1 NEWLINE.**
The computer is now displaying 1 at the bottom left-hand corner of the screen. It is waiting for notes to be entered in channels A, B and C. Supposing you wish to enter A below middle C in channel A, silence in channel B, and E below middle C in channel C. Type:-

**A4 NEWLINE**
**0 NEWLINE**
**E3 NEWLINE**

The computer now displays on the screen:-

**1 A = 4 E = 3**
**2**

It is now waiting for the second 'line' of music.
Supposing the key signature has been set as 2 sharps (F and C) and the following is now entered:-

**C4 NEWLINE**
**A3 NEWLINE**
**F3 NEWLINE**

The computer now displays on the screen:-

**1 A = 4 E = 3**
**2 C + 4 A = 3 F + 3**
**3**

The computer has automatically made F and C into sharps, and is now waiting for 'line' 3 of music. If you now want to make C and F natural, type:-

**C=4 NEWLINE**
**A3 NEWLINE**
**F=3 NEWLINE**

The computer now displays on the screen:-

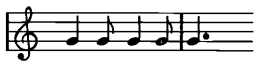
**1 A = 4 E = 3**
**2 C + 4 A = 3 F + 3**
**3 C = 4 A = 3 F = 3**
**4**

The computer is waiting for 'line' 4 of music. The same procedure applies, of course, for changing notes from flat to natural and vice-versa.

The **BACKSPACE** function enables you to backspace as required from any point in the music. The instruction is:-
**5 NEWLINE.**
The **REPEAT** function allows the repetition of line(s) from the line specified to the line you have reached. Instruction:- **R NEWLINE.**
The computer will ask:-
**FROM CHORD NO.**
Type in the line number from which you wish to repeat, followed by **NEWLINE.**
The computer will now repeat the line(s) requested.

**PLAYING** of music is effected through the instruction:-
**P NEWLINE**
Wait for the screen to clear. The computer will now print:-
**SET VOLUMES (Y OR N) OR EDIT (E) OR LIST (L)?**
If you want to play the music and have not already set the volumes, type **Y.**
The computer now prints:- **A=**
Volumes range from 0 (silence) to 15 (loudest).
Type in the volume you require for channel A, followed by **NEWLINE.**
The computer repeats the question for channel B, and when dealt with, C. It will then print:- **SPEED?**
Speeds vary from 0 (fastest) upwards where 50 is approximately one second per computer 'line' or chord. Type in the speed you require, followed by **NEWLINE** and the computer will ask **PLAY FROM LINE NO. ?**
This question is answered by typing in the line number you wish the computer to start playing from. To start at the beginning type **1** followed by **NEWLINE:** the computer will now play the music you have entered.

When the music ends, the computer will ask:-
**SET VOLUMES (Y OR N) OR EDIT (E) OR LIST (L)?**



| LINE No. | Ch1 | Ch2 | Ch3 |
|---|---|---|---|
| 1 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 2 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 3 | 0 NEWLINE | G4 NEWLINE | 0 NEWLINE |
| 4 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 5 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 6 | 0 NEWLINE | G4 NEWLINE | 0 NEWLINE |
| 7 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 8 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |
| 9 | G4 NEWLINE | 0 NEWLINE | 0 NEWLINE |

**Fig. 3 Example of repetitive notes.**

If you want to play the music again without changing the volumes, type **N** and the computer asks:- **SPEED?** Enter the speed as before and the music will be played again.
If however you wish to change the volumes or one or more channels, type **Y** and proceed as before.

**LISTING** of music entered is effected by typing **L NEWLINE** and the computer will ask:- **LIST FROM LINE NO.** Type in the line you wish to start listing from, followed by **NEWLINE.**
The screen will go grey for about 20 seconds and then the computer will have printed 11 lines of music, starting from the line number you typed. You may now type **P** or **E** or **L** depending on which function you wish to use; to continue listing type **L** and give the line number that would carry on from where the first listing ended; to make alterations to the music or to enter more music, type **E,** and to play the music, type **P.**

## Devising Your Own Sound Effects

The third program on the software cassette effectively gives you direct access to the registers in the PSG. These registers are programmed with data to build up sound effects. The following is a short summary of the registers used and their functions.

Registers 0 to 5 determine the pitch (frequency) of the three notes on channels A, B and C. Registers 0, 2 and 4 are used to fine-tune the frequency of A, B and C respectively. Registers 1, 3 and 5 coarse-tune the frequencies. Data to the fine tune registers can vary from 0 to 255, and data to the coarse tune registers from 0 to 15.
Register 6 can vary from 0 to 31. This sets the pitch of any white noise to be included. Register 7 is the enable register. Bits D0 to D5 enable noise and/or sound on channels A, B and C. 0 in a bit of

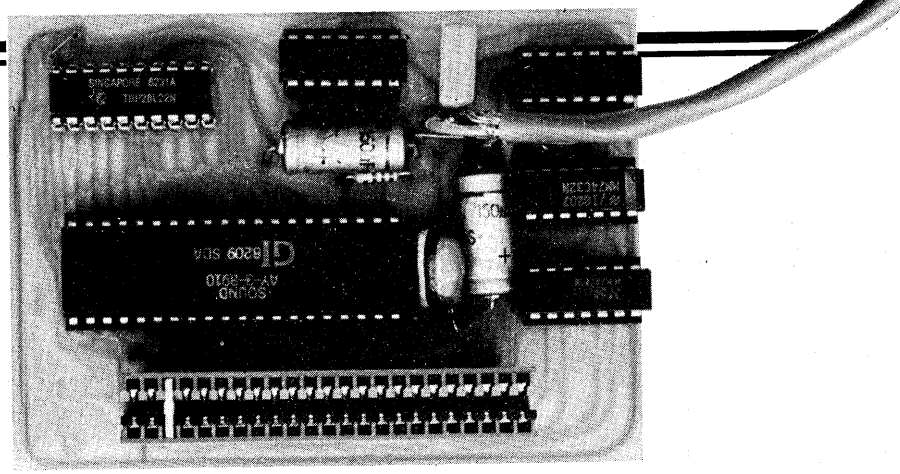
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | C | B | A | C | B | A |
| | | WHITE NOISE | | | NOTES | | |

**Fig. 4 The functions of the bits in PSG register 7.**

this register enables a function, 1 disables it (see Fig. 4).
Registers 8, 9 and 10 set the volumes of channels A, B and C respectively. These can vary from 0 (off) to 15 (loudest). If you send the number 16 to any of these registers, the volume of channels thus set will be varied according to data in registers 11, 12 and 13, the envelope generator, as follows.
Register 11 is used to fine-tune the envelope period; data sent to this register can vary from 0 to 255.
Register 12 is used to coarse-tune the envelope period and data to this register can also vary from 0 to 255.
Register 13 selects the shape of the envelope generator's waveform (Fig. 5).

Now, connect up your amplifier, keeping the volume fairly low, and load the third program ("D") from the software cassette.
Type **GOTO 2 NEWLINE.** The computer will now print:-
**YOU HAVE A CHOICE OF:**
**A (TO HEAR THE SOUND AGAIN)**
**R (TO ENTER ALL NEW DATA)**
**C (TO CHANGE SPECIFIC DATA)**
**L (TO LIST DATA)**
**S (TO SILENCE THE PSG)**
Since you have not yet entered any data, type **R.** The screen will clear and the computer now prints:-
**REGISTER 0 DATA?**
Type in **0** as the data for register 0, followed by **NEWLINE.**
The computer will respond:-
**REGISTER 1 DATA?**
Type in **0** as the data for register 1, followed by **NEWLINE.** The computer will continue to request data for all registers in this manner.
As an example, give it the following:-
Reg. 0 1 2 3 4 5 6 7 8 9 10 11-12 13
Data
0 0 0 0 0 0 31 7 16 16 16 255 40 9
If you examine the data you have just entered, you will see how the computer generated the sound of cannon fire. This data for cannon fire is contained in the PROM. The computer now repeats the question at the beginning of the program. If you type **A**, the computer will repeat the sound and will again ask the same question.

Now type **C**. The computer will respond with:- **REGISTER?**



**This little board does all the hard work.**

Type **6 NEWLINE.** The computer will now ask:- **DATA?**
Type **8 NEWLINE.** In response to the next question **REGISTER?** type **12 NEWLINE** and in response to **DATA?** type **5 NEWLINE.**
Now type **99 NEWLINE.**
You have just changed the data in the computer to generate a rifle shot. If you now type **L** the computer will list the data for you.
You can silence a continuous sound by entering 0 into all register locations. Since doing this would wipe out your sound data, the function **S** for silence is included which will switch the sound off without altering your data.
If you wish to start the sound again, type **A.**

When you have perfected your sound, use the LIST function and copy out the data. You can use your own sounds in your own programs using the fourth program ("G") on the software cassette.

## Mixing User And PROM Effects

Having loaded program "G", type **GOTO 10 NEWLINE.** This runs a short program which simplifies the entering of your sound data. The computer asks **HOW MANY SOUNDS?** Type in the number of different sounds of your own that you wish to include in your program, followed by **NEWLINE.** The computer display will now look like this:-
**HOW MANY SOUNDS?**
**SOUND No. 1**
**REG.0 DATA?**
Type in the data for register 0 in your first sound followed by **NEWLINE.** The display will now read:-
**HOW MANY SOUNDS?**
**SOUND No. 1**
**REG.1 DATA?**
Type in the data for register 1, sound 1 as before. When you have entered data for all 14 registers, the display will read:-
**HOW MANY SOUNDS?**
**SOUND No.2**
**REG.0 DATA?**
Continue entering data as before. When all the data for all your sounds has been entered the program will stop.

The user sound data is held in an array called A; in order not to lose this data do *not* use **CLEAR** or **RUN.** When you wish to run a program use the **GOTO** function. Do not use an array called A in your program and don't use variables Y and Z.

The fourth program consists of lines 1 to 9 and 10 through 21. DO NOT ALTER LINES 1 through 9, though if you wish, when you have entered your sound data, you can delete lines 10 through 21. When you wish to use your own sounds in your program simply insert the following two lines:-
**LET Z = x**
where x is the number of your sound,
**GOSUB 2**
when your sound will be heard.
You can of course mix your own sounds with the on-board sounds in your programs. Use the on-board sounds in the manner previously described — it is not necessary to load program "S" to do this if you have loaded program "G".

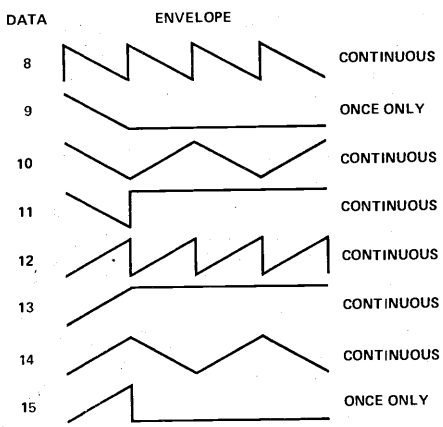



**Fig. 5 Envelope diagram for register 13.**

## HOW IT WORKS — USER EFFECTS

Line 2 dimensions a one dimensional array with 14 positions called A. This array is used to hold the data for the sound effects you are devising. Line 4 makes the computer jump to 200 where it is instructed to print the selection menu. Line 225 makes the computer jump to line 60. Lines 60 to 75 wait for an answer to this question and the computer jumps accordingly. If the function selected was R the computer jumps to line 5. Line 6 sets up a loop using variable D where D = 1 to 14. Line 10 prints the current register number which is held in D and asks what data you wish to go to that register. Lines 15 to 20 input and print the data. Line 25 causes the computer to loop back to line 6 until data for all 14 registers (0 to 13) has been entered. Line 35 again sets up a FOR NEXT loop using D where D equals from 1 to 14 (the PSG register numbers). Line 40 POKEs the register number to memory position 16515 and line 45 POKEs the data for that register to 16519. Line 46 calls a machine code subroutine based at 16514 which outputs to the PSG register (16515) data (16519). Line 50 causes a loop back to line 35 which continues until the data for all 14 registers (0 to 13) has been relayed to the PSG. Line 55 then causes a jump to 200 and a repeat of the initial question.

If the answer to the question at 200 is A, the computer again runs through lines 31 to 55 causing the PSG to repeat the sound. If the answer is C, the computer jumps to line 100. Lines 102 to 110 ask you the number of the register whose data you wish to alter. Line 110 inputs your answer in D and line 111 checks to see if your answer was 99, which would indicate that you had finished altering data for the time being and wished to hear the sound again — in which case the computer would jump to line 30. If your answer was not 99, the computer continues and lines 115 to 125 input and print your new data for the register you gave. Line 130 causes the computer to jump back to line 102 for you to change more data. If your answer was L, you wished the computer to list register data and it would jump to line 150. At line 155 a loop is again set up using D. Line 160 prints the register number (D-1). Line 165 prints the data for that register. Line 170 causes the computer to loop back to line 160, which it continues to do until data for all 14 registers have been displayed. Line 180 makes the computer wait for you to type F, when it will jump to line 30 and your sound will again be heard. If your answer was S, ie the PSG was maintaining a continuous sound and you wished to silence it without losing your data, the computer would jump to 230. Lines 230 to 234 comprise yet another loop using D to output 0 to all PSG registers causing the PSG to become silent.

NOTE: (5 SPC) MEANS "5 SPACES"

```
1    REM (our machine code)
2    DIM A(14)
3    CLS
4    GOTO 200
5    CLS
6    FOR D = 1 TO 14
10   PRINT "REGISTER";
     D-1;"DATA?";
15   INPUT A(D)
20   PRINT A(D)
25   NEXT D
31   CLS
35   FOR D = 1 TO 14
40   POKE 16515,(D-1)
45   POKE 16519,A(D)
46   RAND USR 16514
50   NEXT D
55   GOTO 200
60   IF INKEY$ = "A" THEN
     GOTO 30
61   IF INKEY$ = "S" THEN
     GOTO 230
65   IF INKEY$ = "R" THEN
     GOTO 5
70   IF INKEY$ = "C" THEN
     GOTO 100
71   IF INKEY$ = "L" THEN
     GOTO 150
75   GOTO 60
100  CLS
101  PRINT "TYPE 99 AS A
     REGISTER
     NUMBER [5 SPC] WHEN
     YOU WISH TO HEAR THE
     SOUND."
102  PRINT "WHICH REGISTER?";
110  INPUT D
111  IF D = 99 THEN GOTO 30
112  LET D = D + 1
115  PRINT D - 1; " DATA? ";
120  INPUT A(D)
125  PRINT A(D)
130  GOTO 102
150  CLS
155  FOR D = 1 TO 14
160  PRINT "REGISTER ";D - 1,
165  PRINT "DATA ";A(D)
170  NEXT D
175  SLOW
176  PRINT "PRESS ""F"" WHEN
     YOU HAVE FINISHED"
180  IF INKEY$ < >"F" THEN
     GOTO 180
190  GOTO 30
200  PRINT "YOU HAVE A
     CHOICE OF:"," A (TO HEAR
     THE SOUND AGAIN)","," R
     (TO ENTER ALL NEW
     DATA)","," C (TO CHANGE
     SPECIFIC DATA)","," L (TO
     LIST DATA)","," S (TO
     SILENCE THE P.S.G.)"
225  GOTO 60
230  FOR D = 1 TO 14
231  POKE 16515, (D-1)
232  POKE 16519,0
233  RAND USR 16514
234  NEXT D
235  CLS
236  GOTO 200
```

Machine code in line 1:

```
3E 07 D3 FF 3E 78 D3 F7
21 B4 40 E5 01 FB 0E C5
16 xx 3E 0E D3 FF 7A D3
F7 14 3E 0F D3 FF ED A2
20 F0 C1 E1 16 00 7A D3
FF 5E 7B D3 F7 23 14 10
F5 C9 xx xx xx xx xx xx
xx xx xx xx xx xx xx xx
3E xx D3 FF 3E xx D3 F7
C9
```

## HOW IT WORKS — USER/PROM SOUNDS PROGRAM

This program comprises a line of machine code, a subroutine at line 2 and a program to facilitate the entering of user sound data. This latter program is from line 10 to line 21. It sets up an array called A whose size depends on the number of different user sounds to be provided for. Lines 13 through 19 comprise a double loop which inputs the user's sound data into the appropriate position of array A.

The subroutine at line 2 is called when a sound, whose data is in array A, is to be heard. As has been explained, variable Z is set to the number of the user's sound. Line 2 sets up a loop using variable Y where Y = 1 to 14. Line 4 POKEs the register number (Y - 1) to memory position 16579 and line 5 POKEs the data for that PSG register which is already in array A(Z,Y) to memory position 16583. Line 6 calls a machine code subroutine based at 16578. This subroutine outputs the number at memory location 16579 to the PSG to select a register and the number at memory location 16583 to the PSG as data for this register. Line 7 causes the computer to loop back to line 4 until data for all 14 PSG registers has been output. The user can insert instructions 3 FAST and 8 SLOW, but we recommend leaving these out since the operation of a program in FAST mode causes the computer to discontinue maintaining the video display. This would be annoying, especially in games programs.

```
1    REM our machine code
2    FOR Y = 1 to 14
4    POKE 16579, Y - 1
5    POKE 16583, A(Z,Y)
6    RAND USR 16578
7    NEXT Y
9    RETURN

10   PRINT "HOW MANY
     SOUNDS?"
11   INPUT S
12   DIM A (S,14)
13   FOR N = 1 TO S
14   FOR R = 1 TO 14
15   PRINT AT 1,0;"SOUND
     NO.";N
16   PRINT AT
     2,0;"REG."R-1;" DATA?"
17   INPUT A(N,R)
18   NEXT R
19   NEXT N
20   CLS
21   STOP
```

Machine code at line 1:

```
3E xx D3 FF 3E xx D3 F7
C9
```

# IMPROVING YOUR ZX81

## Ian Ridout gives details of how you can improve the reliability of SAVEing programs on tape.

Since the ZX81 Sinclair personal computer was introduced, over 400,000 have been sold throughout the world. In technical journals and magazines it has been acclaimed by most critics as ideal for self-tuition in the BASIC language and an excellent introduction to computing. By all accounts it represents unequalled value for money both on its own, with just 1K of RAM memory, and also when the 16K RAM plus printer are added.

However, there are three main complaints about the ZX81:

- the mechanical instability of the add-on units can cause a brief disconnection leading to the loss of either TV picture synchronisation and/or loss of the stored program and data;
- an unexplainable loss of program and data (ie a crash);
- unreliable tape storage of programs.

The first complaint can be overcome by making an extension lead so that movements of the individual units can be tolerated without causing momentary memory pack disconnections. However, only the 16K RAM pack should be plugged into the extension; the printer should still be plugged directly into the back of the ZX81. It is necessary to keep the extension lead short (less than 6") or the effect will be to make system crashes occur more often, probably owing to increased capacitive loading and hence memory access delay. Also it is essential to keep the rear edge connector on the computer board clean, using white spirit and cotton buds.

The second problem could well be an internal software fault which is therefore not alterable by ZX81 users. Alternatively, the power supply regulator could be getting too hot because the unregulated input voltage is too high. I decided that it is better to bolt a much bigger aluminium heat sink to the existing heat sink rather than modify the mains power supply. The reason for this is that reducing the regulator's overhead can spoil its functioning, and therefore you end up with the same problem. The size of the additional heatsink was such that it fitted neatly under the entire keyboard area, hence also eliminating a hot-spot under the left-hand end of the keyboard.

This article describes the steps taken to make the recording of programs onto tape and the loading from tape considerably more reliable.

### The Tape Storage System

The IC pin which is used to suply a signal to the TV modulator is also used to supply the signal for tape storage purposes. Consequently, if you connect an audio amplifier and loudspeaker to the 'tape out' (ie MIC) socket, during normal computer usage you will hear a continuous buzzing sound. The video signal from the IC to the modulator consists primarily of high frequencies but most of what you hear is the television frame rate and its harmonics. The output level of this buzzing sound can be used as a very rough indication of whether there is sufficient signal coming out of the MIC socket to allow reliable recordings to be made with your particular tape recorder. However, this method is rather poor because the true signal that is recorded is of much higher frequency than that of the buzzing sound.

When you press SAVE (plus NEWLINE) there is a period of several seconds during which the TV screen goes blank and nothing appears at the MIC socket. After the pause, audio frequency tones are emitted from the IC output pin for recording onto the tape. The picture on the TV screen becomes alternate bands of black and white dashes (data), and black (pauses).

At the end of the recording sequence the IC output signal returns to being a compatible TV video signal, hence a buzz is recorded and the picture on the TV screen returns to normal.

### The Present Circuit

The tape recording circuit used in the ZX81 consists of just two resistors and two capacitors (Fig. 1) forming a band-pass filter with a band peak at about 3.4 kHz. The filter response rolls off each side of this centre frequency at 6 dB/octave (20 dB/decade).

The response shape of this filter is perfectly acceptable for this

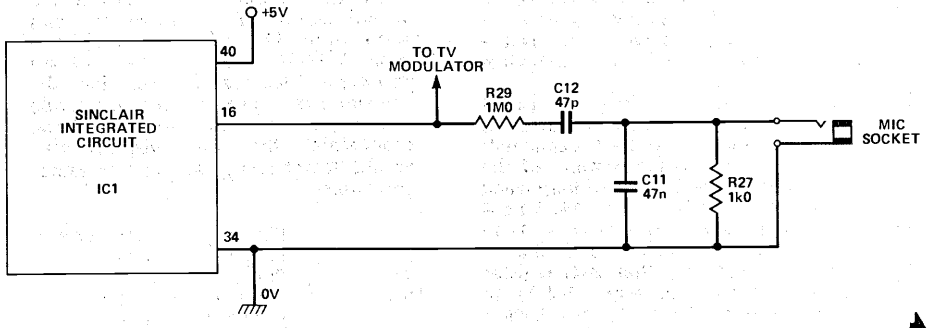



**Fig. 1 Original tape-recording circuit.**

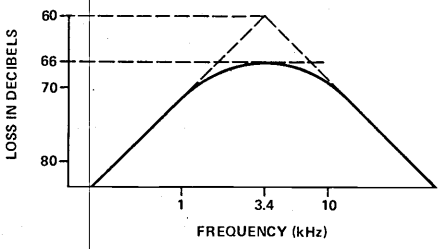



**Fig. 2 Response of tape output filter.**

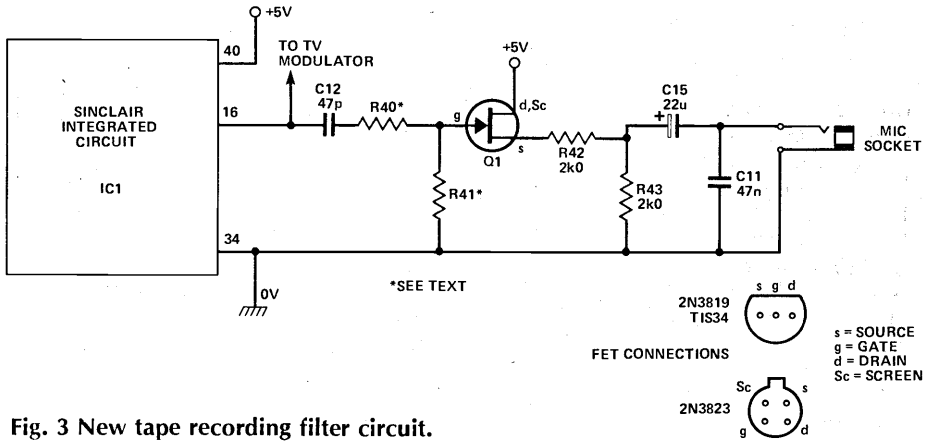



**Fig. 3 New tape recording filter circuit.**

application but the characteristic that makes recording unreliable is the low output signal level at the MIC socket. With about 4 V peak-to-peak coming out of the IC, a signal of only 2 mV appears at the MIC output socket even with this socket open-circuit. This is because the filter components give a resistive attenuation of 1000 to 1 and because a further halving of the signal takes place owing to the filter shaper. The response of this filter is given in Fig. 2, the axes being plotted with logarithmic scales.

A signal of 2 mV is insufficient to drive most recorders with manual recording-level control and is not high enough to give an adequate signal to tape-noise ratio on recorders with automatic level control.

## Circuitry Modification

To ensure reliable tape recording, it is necessary to amplify the signal by at least 10 times and preferably 100–200 times. This could be achieved by connecting a suitable amplifier of the appropriate gain on the output of the Sinclair filter. However, with such a low signal level, it is possible to pick up interference along with the signal, so one would need to be very careful with the layout of the amplifier input wiring and the power supply to the amplifier.

Because of the high signal level and the large bandwidth of the signal at the IC output, it is not practicable to put an amplifier before the filter. It is therefore necessary to redesign the filter connected between the IC output and the MIC socket. The design has to present the original high impedance load to the IC output pin, and preferably should present the original output impedance to the MIC socket although the latter is not essential. Also, the same filter shape has to be preserved but the output signal level has to be considerably higher than the 2 mV presently available. Naturally, it is also an aim to keep the cost of the modification to a minimum.

The result of fulfilling these requirements was a circuit using a cheap junction field-effect transistor (FET) as a source follower, and a few extra resistors, costing in all less than £1. The circuit also has the advantage that the output signal at the MIC socket is not inverted, although this is not essential.

The new tape-recording circuit is shown in Fig. 3. R40 and R41 ensure that the IC is presented with the same resistive load as in the original circuit and C12 gives the same low-frequency roll-off. The DC biasing requirements of the FET dictate that with the gate connected to the negative supply rail, the value of the source resistor should be about 4 k to obtain about 2 V DC on the source. Consequently, to achieve an output resistance in the order of the original 1k0, the output is taken from a tap halfway down the source resistor.

The output needs to be AC coupled, hence the need for the 22u electrolytic capacitor. The 47n capacitor C11 could have been connected directly across the 2k0 resistor R43 but it was more convenient to leave it in its original physical position on the PCB directly across the MIC socket.

It was not found necessary to feed the drain of the FET via an RC decoupling network, so the drain was connected directly to the regulated +5 V supply, the drain current being about half a milliamp.

The values of R40 and R41 need to be set according to whether the cassette machine is of the automatic or manual record level type (see Table 1). Most machines likely to be used for this application would be of the automatic record level type so R40 and R41 should be set to 910k and 91k respectively. If the record level is too high and overloading takes place, reduce R41 to about 15k. For manual record level machines these resistors should be set to 510k each. The sum of the values of R40 and R41 should be between 900k and 1M1 in order to preserve the filter characteristic.

**TABLE 1**

| | R40 | R41 |
|---|---|---|
| **Manual recorders** | 510k | 510k |
| **Auto level recorders** | 910k | 91k* |

*See text

## Doing The Modification

Advantage has been taken of spaces for components on the PCB which are not used in the UK version, namely R30, 31, 32 and D9. Three of the original components need to be removed and only one of these will be used again. The new layout is shown in Fig. 4 but reference to the photograph will assist.

Make the modification as follows: pull off three of the four rubber feet, leaving the one nearest to the TV output socket. Remove the five screws on the back of the case and note that the two short screws come from under the feet under the keyboard. Remove the back of the case.

Remove the two screws holding the printed circuit board to the front half of the case. Carefully raise the PCB ensuring that the strip-connections from the keyboard are not strained and are disturbed as little as possible. DO NOT remove the keyboard connection strips from the PCB sockets because they are not easy to reinsert.

Locate and remove R27 (1k0) and R29 (1M0). These two resistors will not be used again. The position that R27 occupied will be left empty. Locate and carefully remove C12 (47pF, positioned between C10 and C11). This capacitor will be used in the new circuit.

Before continuing, refer to Fig. 4. Insert the two 2k0 resistors R42 and R43 into the positions marked on the PCB as R31 and R32. Insert one end of R40 (see Table 1 for the value to be used) into the left-hand hole vacated by the removal of R29. Insert one lead of C12 into the right-hand hole vacated by the removal of R29. Solder together the remaining leads of R40 and C12 above the PCB, keeping their leads short.

Insert the positive end of C15 (22u) into the hole shown in Fig. 4. This hole is the one between EAR socket and the modulator which has a PCB track joining it to the two left-hand ends of R42 and R43. Solder the negative end of C15 to the left-hand hole vacated by C12. C15 should be positioned above, but not touching, the EAR socket. Do not be tempted to position C15 in the space between C10 and the modulator because one of the PCB support pillars occupies this space.

Solder the drain of the FET to the hole next to the right-hand end of C11. This is a plated-through hole which is connected to +5 V. Solder the source of the FET to the right-hand end of the position marked on the PCB as the cathode end of D9 (not used in the UK version). Position R41 (see Table 1 for its value), so that is to the right of the FET. Its lower lead is soldered into the left-hand hole of the component designated as R30 (not used) and its upper lead is soldered into the right-hand hole of the position previously occupied by C12. Solder the gate of the FET to the upper lead of R41.

After cutting off all excess leads and thoroughly inspecting all new solder joints for solder blobs and tracking between adjacent holes, put back the two short screws holding the PCB to the front cover. They are the ones near the regulator and half-way along the rear accessory socket. (You might like to use this opportunity to make the heatsink modification described earlier).

Before screwing on the back cover, connect the power, the television and the cassette recorder, write a very short program and ensure that it SAVEs and LOADs properly.

The two short screws go into the holes under the feet behind the keyboard. The new setting for the cassette player volume control will have to be obtained by experimenting.

## Fault Finding

If the television picture fails to appear, check that the television is tuned to the correct channel (36) and that the regulator is giving out 5 V. Next, check that the FET is wired correctly and that it has 5 V on its drain and 0 V on its gate. The biasing is such that the source should be at about 2 V. If the source is at 5 V check the connections to R42 and R43 and their values with a multimeter, after disconnecting the computer power. If the source voltage is higher than 3V5 or less than 1 V you have a FET which is on the tolerance limits and the combined value of the source resistors will have to be changed to compensate. Initially, change only R42, but if it has to be varied significantly to achieve 2 V DC on the source then I would suggest replacing the FET.

The type of FET used in this circuit is a junction FET and consequently there are no handling precautions necessary beyond those normally used for bipolar transistors. A 2N3823 was used in the original and the case/screen lead was soldered to the drain before insertion into the PCB. A TIS34 or the very cheap, plastic encapsulated 2N3819 could be used instead.

When a tape has been recorded from the computer, listen to it to see if it is distorted due to overloading. Naturally, it will be necessary to experiment with the playback level when loading the program back into the computer because the record level on the tape will be a little higher even if recorded by an automatic record level machine.

## PARTS LIST

**Note that this is a list of the *new* components required**

| | |
|---|---|
| **Resistors (all ¼W 5%)** | |
| **R40** | **See Table 1** |
| **R41** | **See Table 1** |
| **R42,43** | **2k0** |
| **Capacitors** | |
| **C15** | **22u 16 V axial electrolytic** |
| **Semiconductor** | |
| **Q2** | **2N3819 or TIS34 or 2N3823** |

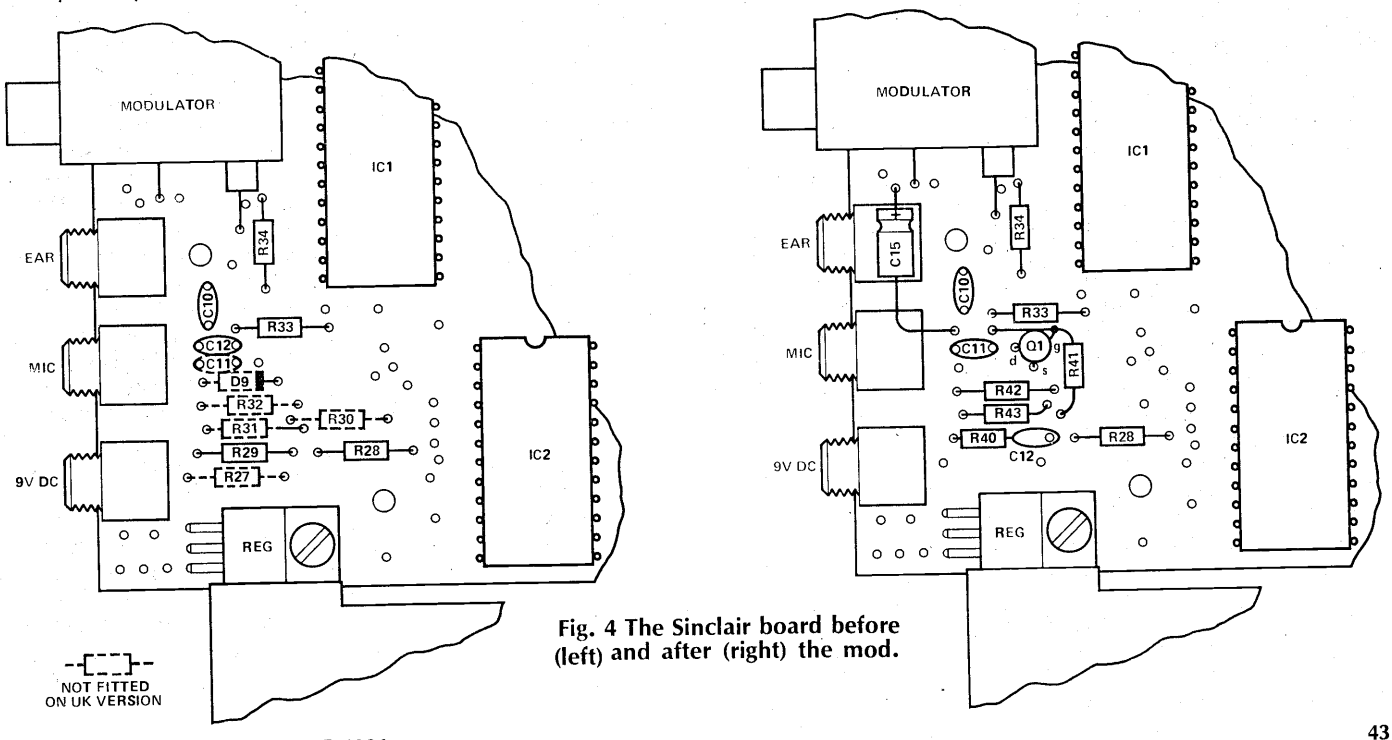



**Fig. 4 The Sinclair board before (left) and after (right) the mod.**

# ZX81 USER–DEFINED GRAPHICS

## For another example of the DIY guide to making your ZX81 a whole lot better, we present a project that gives 16K RAM pack owners the facility for user-defined graphics. Design by G.N. Hill, MA Hons.

The restrictions imposed by the Sinclair defined character set can considerably reduce impact of many programs and the attractions of having user-definable graphics are apparent. While several manufacturers supply modules to expand the character set, the cost of £20 plus must be a rather daunting prospect to the impecunious '81 owner.

It is possible, however, to obtain user-definable graphics on the 16K ZX81 for a total expenditure of less than £1 and some clever work with a soldering iron! It must be said, however, that this is not a project for the fainthearted, involving as it does a certain amount of modification and soldering within the computer; it also renders the computer unusable without the RAM pack.

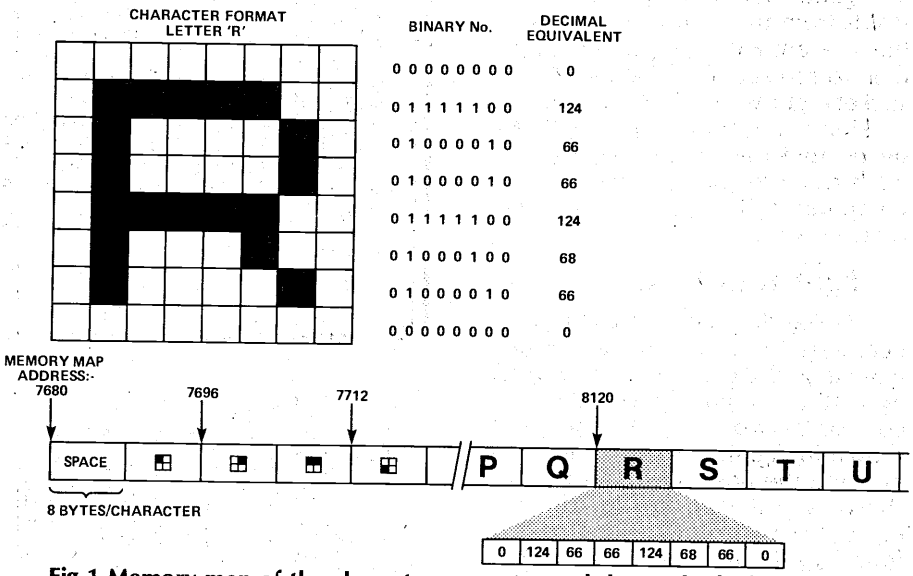



**Fig 1 Memory map of the character generator and the method of character generation, illustrated by the letter 'R'.**

### Principle of Operation

In normal operation the pattern of each character is defined by eight successive bytes in the ROM. Each byte represents the display for one of the eight lines of the character, and each binary bit one dot of that line: Fig. 1 clarifies the way that this works. As there are 64 characters, a total of 512 (64 × 8) bytes are required, and these are located at the top of the ROM from addresses 7680 to 8191 (1E00 to 1FFF in hexadecimal). An examination of these addresses reveals that the character generator is being addressed exclusively when lines A9 to A12 are high.

The additional circuit operates by detecting when these four lines are simultaneously high with the ROM chip select line, and instead of allowing the ROM to be switched on, the internal 1K RAM is activated (this is not normally used when the 16K RAM pack is attached). This can then be filled with characters of the programmer's choice. The circuit diagram is given in Fig. 2.

### Construction

The circuit is constructed on a PCB as shown in Fig. 3. Use fairly thin (insulated) wire, preferably with colour coding, for the leads between the PCB and computer. The length of the leads will need to be adjusted carefully during connection to the computer PCB in order to keep the modification as neat as possible.

### Installation And Computer Modifications

To open the computer remove all leads and accessories and then peel off the rubber pads on the underside of the computer, taking care to ensure that the adhesive film is removed with the pads. Now remove the five Phillips screws visible on the underside of the computer; note that these are of different lengths and *must not* be mixed on re-assembly. The base of the computer can now be lifted away to reveal the computer printed circuit board held in place by a further two Phillips screws, which must also be removed. The board is still attached to the case through the ribbon connector and should be carefully folded over the keyboard when you want to turn it over, since the ribbon connector is not easy to disconnect or reconnect.

The positioning of the switch does need to be thought out fairly carefully in order to ensure that it does not foul anything. Providing the switch is not too big, it should fit through the top of the case. Just in front of the RAM pack, towards the centre of the computer offers the most room, but do make sure before drilling the hole that it will all go back together (including the RAM pack connection) without fouling. If you find the prospect of drilling holes in your beloved

computer a little daunting, the leads can be brought out through one of the existing holes (eg round the aerial socket) to a floating switch.

The addressing of the RAM and ROM chips is performed in a slightly different way by the computer and it is therefore necessary to disconnect the 10 address lines to the RAM chip(s), reconnecting nine of them to the (A0' to A8') ROM chip address lines and the tenth to either an earth or a logic high. This cannot be done easily from the underside of the board, as the address tracks servicing the 1K RAM also service the 16K RAM pack. The most straightforward (if somewhat inelegant) solution is to remove the RAMs from their sockets, to bend the relevant legs through 90° and then to replace the RAMs in their sockets, but with 10 of the pins sticking out at right angles. It is now necessary to solder the address leads to these legs and into the computer PCB — Figs. 4a, b, c and d give details.

## Spot Your RAM

Before doing this, it will be necessary to decide whether your particular ZX81 is of the 2114 RAM or 4118 RAM type. The former consists of two separate chips as shown in Fig. 4a; the latter has only one and the position of this is shown as a dotted outline. Identify each of the address pin numbers from Fig. 4d and run a lead from this to the relevant computer PCB hole as shown in Fig. 4c. In the case of the 2114 type it is also necessary to run leads between the two ICs. If you find difficulty in soldering directly to the IC pins, it may help to have a very thin strip of Veroboard pushed over the pins and solder to this. The leads to the switch can now be soldered into place.

Figure 5 shows the nine connections to be made to the other side of the computer board; the letters for each connection correspond with those of Fig. 3. Resistor 'R28' is removed from the computer PCB and the ROM chip select input and output connections are made through the resulting holes as shown. Figure 5 also identifies the RAMCS' track, which must be (carefully!) cut through. The PCB fits under the keyboard by the side of the heatsink.

## Testing And Reassembly

It is worth quickly testing the computer at this stage. First,

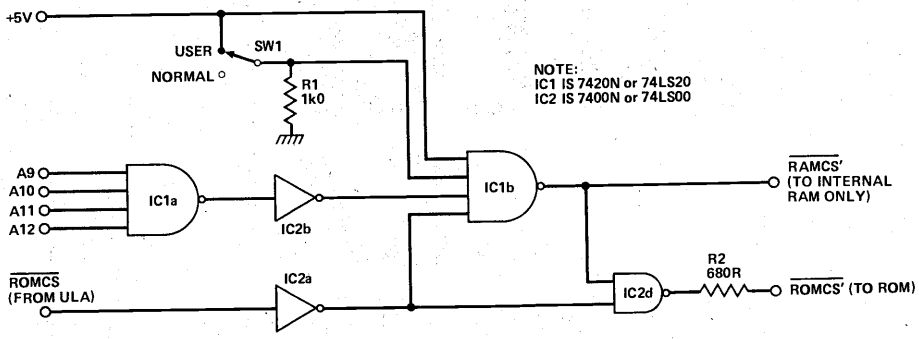



**Fig. 2 Circuit diagram of the user-graphic modification.**

## HOW IT WORKS

The inputs of the quad NAND gate IC1a are connected to the four address lines A9 to A12. The output of this gate is inverted (using the NAND gate IC2b with its two inputs connected together) and used as one of the four inputs for IC1b. A second input is produced by inverting the ROMCS (ROM chip select) signal from the ULA using IC2a. The third input is provided by the switch, which in 'user' mode supplies a high and in 'normal' mode a low. The fourth input is held permanently high.

Thus, when the switch is in 'user' mode and the ROM and the character generator are being addressed, all the inputs to IC1b will be high and the output will be low: in any other circumstances the output will be high. This, therefore, satisfies the condition required by the RAMCS' line to the internal 'user' RAM.

The output of IC2d provides the new ROM chip select line (ROMCS') by performing a logical NAND between the new RAMCS' output and the inverse of the ROMCS input from the ULA. Thus, when the user RAM is being selected, the output will always be high, and when the RAM is not being selected the output will copy the ROMCS input.

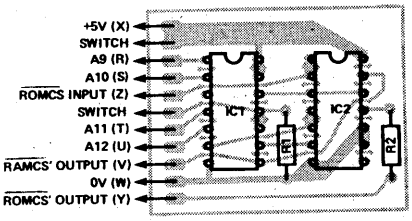



**Fig. 3 Component overlay for the off-board components. The letters correspond to Fig. 5.**

## PARTS LIST

| | |
|---|---|
| Resistors (all ¼W, 5%) | |
| R1 | 1k0 |
| R2 | 680R |
| Semiconductors | |
| IC1 | 74LS20 |
| IC2 | 74LS00 |
| Miscellaneous | |
| SW1 | SPST miniature toggle or slide switch |
| PCB (see Buylines). | |

however, check, check and check again that all the joints and connections are correct and that there are no solder bridges between tracks.

Place the RAM pack in position and connect the TV lead and power supply lead. With the switch in 'normal' mode the computer should operate quite normally. In 'user' mode a regular pattern should cover the screen (each space now calling up the same random character). The program given below can be entered in 'normal' mode and run in 'user' mode to ensure that the modification is functioning correctly:

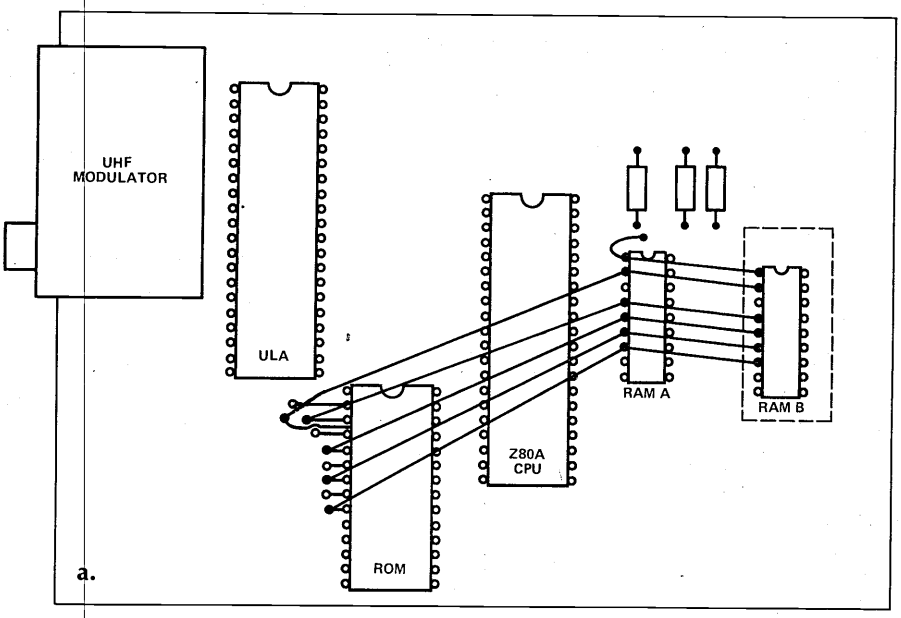
```
10 SLOW
20 FOR N=1 TO 64
30 PRINT CHR$ N;
40 NEXT N
50 FOR N=7680 TO 8191
60 POKE N, 255
70 NEXT N
```

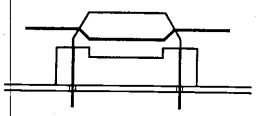
When run in 'user' mode the screen should suddenly black out apart from the top two lines, which should then progressively fill one character position at a time from left to right, each individual character filling from top to bottom.

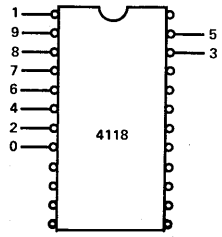
Reassembly of the computer is simply a reversal of the disassembly procedure. Ensure that no leads are trapped between the pillars and the board and that the correct screws are used; short ones at the front, long at the back.

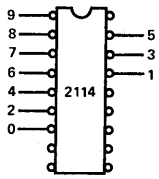
## Operation Of User Graphics

It is unlikely that you will want to redefine all 64 of the available characters and the most likely requirement will be to have four or five special characters amongst the existing ones. As it is not possible to display both the Sinclair graphics



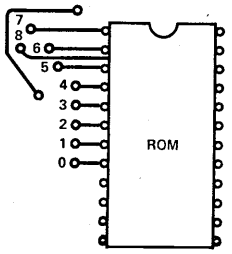




**Fig. 4 General view of the component side of the computer PCB, with 2114 RAM type (4118 type shown dotted). Only six of the 10 new connections are shown (a). The RAM address pins are bent outwards (b). Connections into the ZX81 PCB from the RAM chip(s) are shown in (c), while (d) gives the address pinouts for the RAM.**

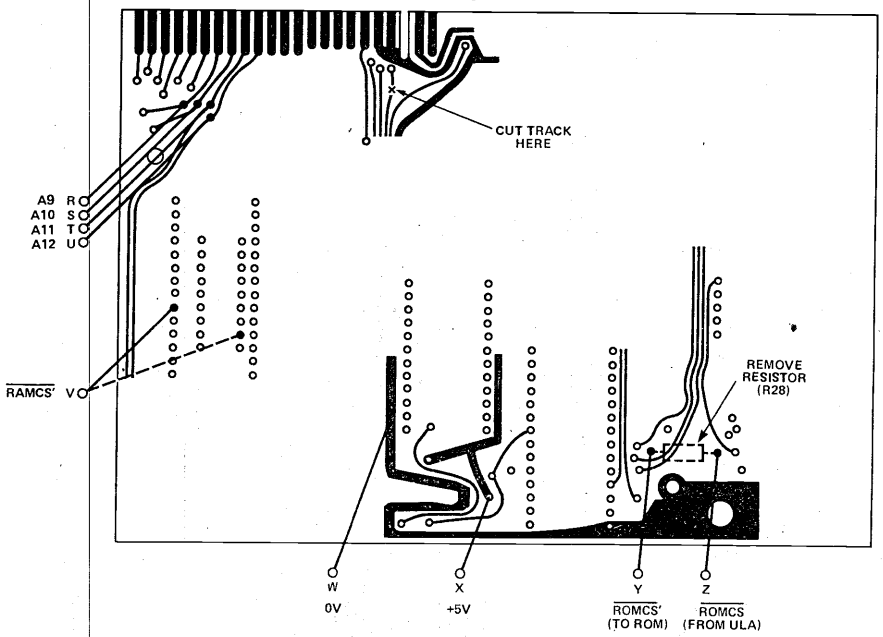



**Fig. 5 Connections to the underside of the ZX81 PCB. $\overline{\text{RAMCS}}$' connections are shown for both the 2114 type (dotted) and the 4118 type (solid) RAM ICs.**

and the user graphics at the same time, the simplest solution is to copy the Sinclair graphic set into the user RAM, and then to select some little-used primary characters (eg <, > , £, ?) and redefine them.

The exact way that this is done can vary, but my usual approach is to copy the Sinclair graphics into the user RAM at switch on. Listing 1 gives a short program (using machine code) to achieve this. When entering the REM statement (line 1) it must be entered exactly as listed. The outlines of the graphic characters have been added to clarify which symbols to use, and don't miss the three spaces in line 1 which require keying in. The keyword in line 1 is entered by first typing THEN (shifted 3), followed by GOSUB and deleting THEN using the Edit functions, but leaving the GOSUB intact. As a quick check that the line has been entered correctly, PEEK 16526 should return 118. The program borrows some of the unused computer RAM to temporarily store the 512 bytes read from the ROM and should not, therefore, be used as part of a larger program, otherwise this may be overwritten and corrupted. If this is a problem, the program can be written entirely in BASIC and listing 2 gives such a program.

Having thus copied the Sinclair graphics into the user RAM, each program can then amend the characters as required and preferably restore the original ones at termination.

```
    1 REM ■ "5 2) 2 GOSUB ▣TAN
    2 POKE 16552,80
    3 LET U=USR 16514
    4 PRINT "SWITCH TO ""USER"" A
ND PRESS ""CONT"""
    5 STOP
    6 POKE 16522,30
    7 POKE 16519,80
    8 LET U=USR 16514
    9 POKE 16519,30
```

**Listing 1 Program to copy Sinclair character set into user RAM, using a machine code routine.**

```
   10 FAST
   20 DIM C(512)
   30 FOR A=7680 TO 8191
   40 LET C(A-7679)=PEEK A
   50 NEXT A
   60 PRINT "SWITCH TO ""USER"" A
ND PRESS ""CONT"""
   70 STOP
   80 FOR A=7680 TO 8191
   90 POKE A,C(A-7679)
  100 NEXT A
```

**Listing 2 BASIC program to copy Sinclair character set into user RAM.**

## BUYLINES

Absolutely nothing out of the ordinary in this project, and any of the mail order component suppliers advertising in the electronics press will be happy to supply all the parts.

```
    1 REM 000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000
   5 LET AD=16580
  10 DIM A(8,8)
  20 FOR Y=1 TO 8
  30 PRINT "XXXXXXXX";Y
  40 NEXT Y
  50 PRINT "ABCDEFGH"
  60 PRINT AT 15,0;"INPUT PIXEL
COORD EG. A1 . HIT  NEWLINE TO F
INISH"
  70 INPUT A$
  80 IF A$="" THEN GOTO 160
  90 IF CODE A$(1)<CODE A$(2) TH
EN GOTO 70
 100 LET Y=CODE A$(2)-28
 110 LET X=CODE A$(1)-37
 120 IF X>8 OR X<1 OR Y>8 OR Y<1
 THEN GOTO 70
 130 LET A(X,Y)=ABS (A(X,Y)-1)
 140 PRINT AT Y-1,X-1;CHR$ (61+A
(X,Y)*57)
 150 GOTO 70
 160 CLS
 165 PRINT AT 15,0;"WHICH CHARAC
TER IS TO CHANGE?"
 170 INPUT A$
 180 LET CA=7680+8*CODE A$
 190 LET HB=INT (CA/256)
 200 LET LB=CA-256*HB
 210 POKE AD,LB
 220 POKE AD+1,HB
 230 LET AD=AD+2
 240 FOR Y=1 TO 8
 250 LET D=0
 260 FOR X=1 TO 8
 270 LET D=D+A(X,Y)*2**(8-X)
 280 NEXT X
 290 POKE AD,D
 300 LET AD=AD+1
 310 NEXT Y
 320 IF AD<16671 THEN GOTO 350
 330 PRINT AT 15,0;"YOU HAVE DEF
INED THE MAXIMUM OF 10 CHARACTER
S. IF YOU WISH TO DEFINE MORE CH
ARACTERS PLEASE SEE TEXT OF ARTI
CLE. OTHERWISE TYPE PRINT USR 15
360"
 340 STOP
 350 PRINT AT 15,0;"*****PRESS
*****TYPE Y OR N "
 360 INPUT A$
 370 IF A$(1)="Y" THEN PRINT USR
 16560
 380 CLS
 390 GOTO 10
```

**Listing 3 Program for defining new characters and storing the information in the REM statement for later use (left): the values above are to be POKEd into the REM statement of line 1 using listing 4.**

```
ADDRESS
16514     62     10     33    195    64
16519     17    186     64    229    205
16524     203    1     10     0     237
16529     176    46    186    64     209
16534     209    19    195    1      0
16539     0     237    176    209    225
16544     56     35     1     8      0
16549     237    176    225    1      10
16549     237    176    225    1      10
16554     0      9     64     200    24
16559     215    33    41     65     195
16564     3      4
```

**Listing 4 Program for POKEing machine code values into the REM statment of listing 3.**

```
 500 PRINT "WHAT IS THE START AD
DRESS?      (NORMALLY 16514)"
 510 INPUT A
 515 CLS
 520 PRINT AT 0,0;"INPUT NEXT NO
. TO BE POKED. TYPE 999 TO TERMI
NATE"
 530 INPUT D
 540 IF D=999 THEN STOP
 550 IF D<0 OR D>255 THEN GOTO 5
30
 560 POKE A,D
 570 SCROLL
 580 PRINT AT 16,0;A,D
 590 LET A=A+1
 600 GOTO 520
```

Listing 3 gives a program enabling up to 10 new characters to be defined with the aid of a visual display and the information stored in the REM statement of line 1. Only the primary characters (with their inverse) can be redefined and these are the first 64 characters of the list on page 181 of the Sinclair manual. When all the characters have been defined and "FINISHED?" is answered with "Y", the program deletes itself, excepting the REM statement. This can then be used as the first line of the main program and a call for USR 16514 will exchange the Sinclair characters with the new characters required; a further call of USR 16514 will exchange them back.

When entering the REM statement of listing 3, it is important to use noughts and to have exactly the right number (166). This can be checked by PEEKING 16680, which should return 118.

This program (ie listing 3) requires the machine code values given alongside it to be POKED into the REM statement. Listing 4 gives a suitable program for doing this and this should be entered and run immediately after entering the REM statement. All of this program (except the REM statement) may then be deleted and the rest of listing 3 entered.

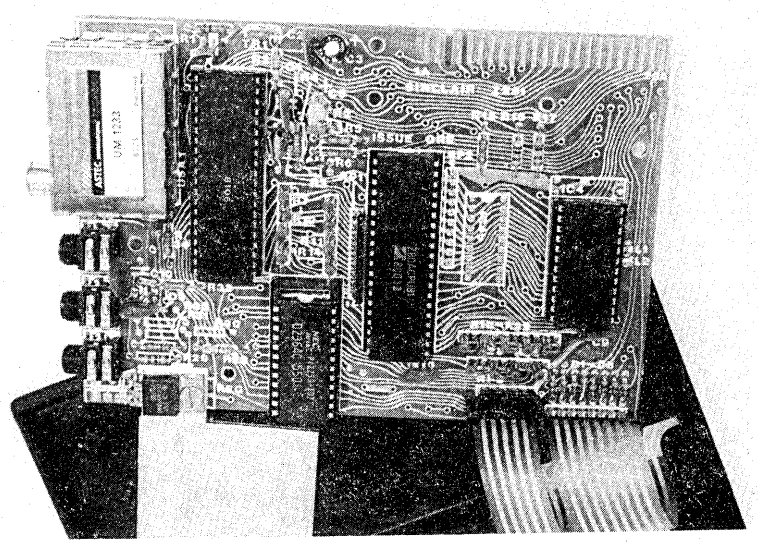
If more than 10 characters need to be defined, it will be necessary to add 10 further noughts to the REM statement for each extra character required; the final address (currently 16671) in line 320 will have to be increased by 10 for each extra character, as will the value held by the two bytes in addresses 16561 and 16562 (currently 16681). This latter value is held in the usual Z80 manner, ie. low byte first and high byte second. Having added on the required number of tens to 16681, divide by 256 to give the high byte (being the whole number part of the result) and multiply the remainder by 256 to give the low byte. For example, 16691/256 = 65.19922. Now 0.19922 × 256 = 51, so we have to POKE 16561, 51 and POKE 16562, 65.

Because the graphics are completely unaffected by RUN, NEW, LOAD, and so on, it is also possible to enhance many of your existing games and programs by redefining specific characters used in the game before loading the program. Thus chess games using I, K, Q etc can have proper chess symbols, missiles can magically turn from As into Cruise look-alikes, and so on.

## Further Hardware Modifications

While this project was conceived as a very low-cost entry into user-definable graphics, several modifications could be made to bring a greater degree of sophistication to their operation. The manual switching could be replaced by software switching, for example: if you have an input/output port, this can be done by using one of the output port lines as the switch input for IC1b.

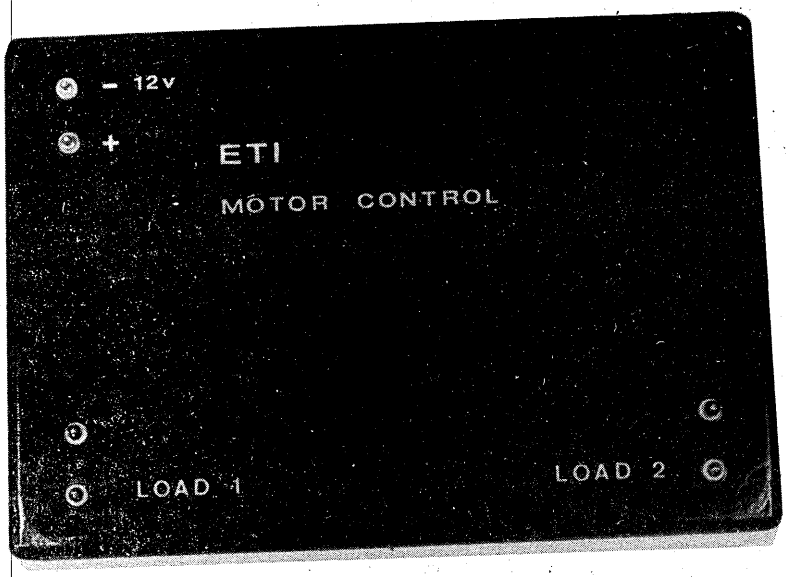
Only half of the 1K memory is used and the A10 address line from the RAM chip could be switched between high and low instead of being fixed to a high, thus allowing two pages of user graphics to be available. Again, this could be a software switch controlled by an input/output port.



**The PCB before modification; this is a 4118 type (compare with Fig. 4a).**

# ROBOT MOTOR CONTROLLER

## This heavy-duty piece of hardware is the first part of a series which could lead up to an intelligent programmable mobile. Design and development by Rory Holmes.




In this article we are offering a design for a high powered DC motor controller based on a switching amplifier capable of independently driving two motors at currents of up to 10 A.

The first part describes the design and construction of the power switching circuitry; in the following part we shall be offering both digital and analogue versions of pulse width controllers for driving the power stage.

Motor speed control amplifiers are very useful devices, finding many applications in such things as motorised wheelchairs, mobile computer peripherals, robots and many industrial process control functions. We are using ours as the main traction control in a small vehicle intended as the basis for a mobile computerised robot.

The ETI motor control is a four quadrant servo type; it can provide motor speed and torque in negative and positive directions and is designed to use 20 kHz constant frequency pulse width modulation. It features two CMOS logic inputs for forward/reverse control and duty cycle control and will suit a variety of motors up to 12 V — or even more if reduced speed is acceptable.

With suitably designed positional feedback it may also be used as the power amplifier in a positional servo system. Each control channel can handle 10 A of motor current and, due to a symmetrically designed PCB, it can easily be built as a single motor speed controller if desired.

DC motors are unruly beasts as far as electronic control is concerned and there is more to the design of suitable amplifiers than meets the eye.

Speed control can be achieved using a linear amplifier where a regulated voltage applied to the motor is varied; however, the torque available falls off at low speeds and the power dissipation in the output transistors becomes excessive at large currents due to the voltage dropped.

This power dissipated also represents a waste of valuable battery power. Hence the decision to use a 'pulse width modulated' system, where the amplifier switches the supply voltage on and off at a fixed frequency with variable mark-space periods so that an adjustable average current through the inductive motor load is produced.

The first problem encountered is in selecting the switching frequency. Most motors produce a disturbing noise if this lies in the audio band, but on the other hand power transistors don't switch off very quickly, which ruins high frequency performance. Fortunately this problem can be minimised using higher frequency transistors and low values of transistor "base turn-off" resistors.

Another problem encountered is when the transistors switch off; the motor current doesn't, due to motor inductance. This disaster can be averted by using 'flywheel' diodes to provide an alternative current path in a closed circuit round the motor. These diodes also protect the output transistors from switching transients caused by the back EMF of the motor. A single supply rail suitable for a 12 V car battery was chosen, and since a reversing function was desired, an 'H' type bridge output stage must be used. This can provide reversal of the motor voltage polarity within a few microseconds and is thus suitable for positional servos.

## Construction

We have arranged the PCB and the general construction to make it easy for those who only require one motor control, rather than the dual version we are presenting. The PCB is symmetrical and can simply be sawn in half and wired up in a smaller box in the same manner described for the double system. We strongly recommend using the diecast box we have specified, since this provides RFI screening and also acts as a firm heatsink for bolting the power transistors to.

Our aim was to bolt all the transistors on to the metal case with their leads directly soldered to the PCB and to get all the components on the inside of the box! Hence the unusual method of soldering the transistors from the copper track side of the board (see the internal photos). Bridge

rectifiers come in all shapes and sizes and at the specified current rating usually have a hole for heatsink mounting and large connecting tags. For this reason we decided to mount the bridge off the PCB, using stiff wire links for connecting to the board.

Solder the components to the PCB as normal, checking component polarity and remembering to solder the Veropins at the input points marked on the overlay; do not connect the power transistors yet. These power transistors, Q1-4, should now have their leads bent as shown in Fig. 3. They are fitted in to their appropriate holes from the underneath of the board and soldered in place — allowing a gap of 1 cm between the component side of the PCB and the metal bottom of the transistor.

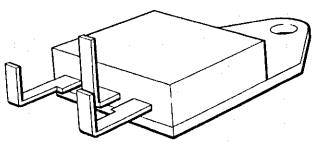
Lengths of thick tinned copper wire (about 3") can now be wired to the PCB at the points marked for connection to the bridge rectifiers. They should be left protruding from the top of the board. Testing of the assembled board can be done at this stage, providing a resistive load is used as described below, and it's probably worth doing before everything is tightly bolted up. After testing, the holes for transistor mounting must be drilled in the bottom of the diecast box.

The board is bolted by the transistors, component side up, in the bottom of the box and the bridge rectifiers are bolted on either side. Remember, it is essential to use insulated mounting kits for the transistors. The leads to the motor and battery supply can be let out of the box through small holes drilled near the bridge rectifiers. We actually used leadthrough insulating terminals mounted on the case and took the supply and motor leads from the bridge rectifier tags.

## Testing

The unit can be tested using a resistive load of about 100R, under these conditions it need not be connected to the bridge rectifier or the heatsink.

Connect a 12 V, 1 A power supply at the points shown on the overlay to one channel and a 100R resistor across the leadout wires where the motor



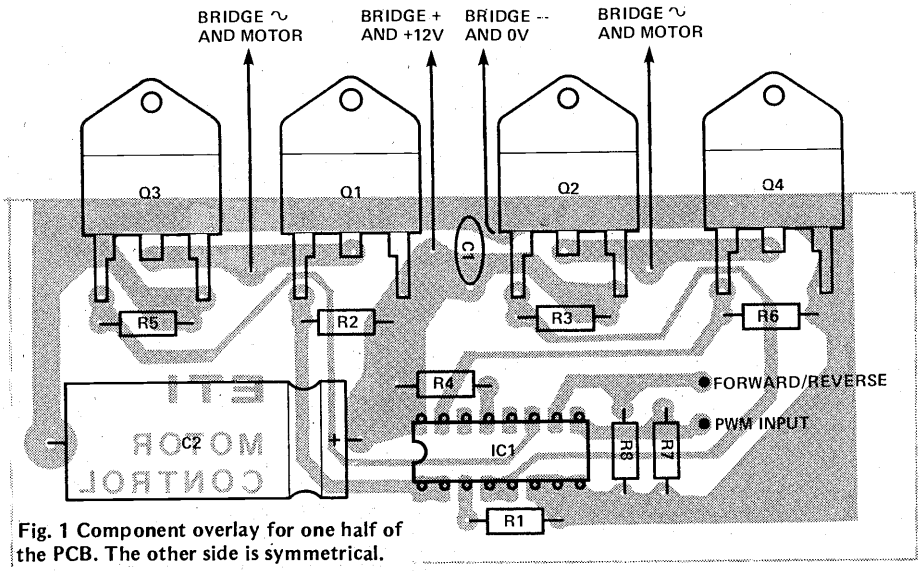
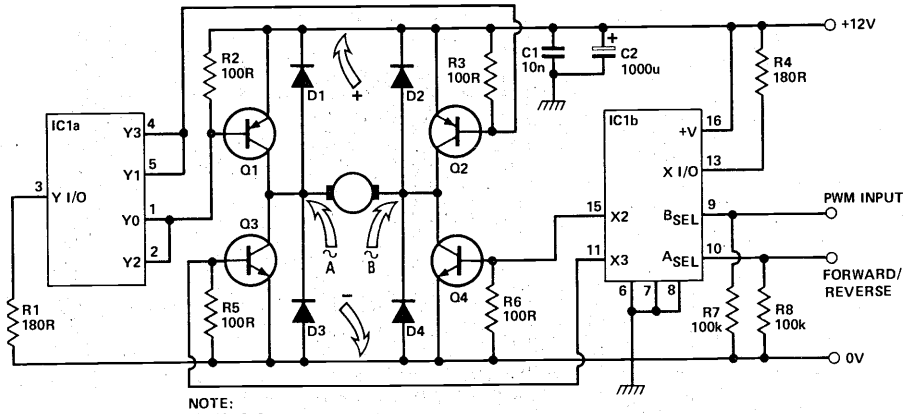



Fig. 1 Component overlay for one half of the PCB. The other side is symmetrical.




NOTE:
IC1 IS 4052B
Q1,2 ARE TIP146 OR TIP147
Q3,4 ARE TIP141
D1-4 IS 10A, 100V BRIDGE RECTIFIER

Fig. 2 Circuit diagram for one channel of the controller. Components for the other channel are designated R101, R102 etc.

## PARTS LIST

**Power Switching Section (for dual controller)**

**Resistors (all ¼W, 5%)**

| | |
|---|---|
| R2,3,5,6,102, 103,105,106 | 100R |
| R1,4,101,104 | 180R |
| R7,8,107,108 | 100k |

**Capacitors**

| | |
|---|---|
| C1,107 | 10n ceramic |
| C2,102 | 1000u 25 V axial electrolytic |

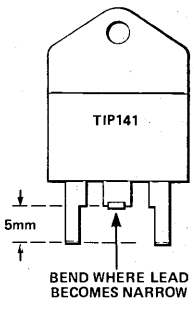
**Semiconductors**

| | |
|---|---|
| IC1,101 | 4052B |
| Q1,2,101,102 | TIP147 or 146 |
| Q3,4,103,104 | TIP141 |
| D1,2,3,4,101, 102,103,104 | Bridge rectifier (10 A, 100 V; heatsink mounting type) |

**Miscellaneous**

Transistor mounting kits; diecast box (172 x 120 x 55 mm); PCB (see Buylines); robot chassis and DC motors; 12 V battery.

Fig. 3 Bending details for the power transistor leads.




## BUYLINES

The semiconductors used in this project are somewhat exotic; however, the Darlington power transistors (a gain of 1000 at 10 A) are available from Rapid Electronics and Technomatic, while the bridge rectifiers should also be available from these firms and other semiconductor suppliers. The robot chassis that we used is available from Remcon Electronics, 1 Church Road, Bexleyheath, Kent DA7 4DD, at a price to be announced.

should be. With the two logic inputs unconnected, points A and B on the circuit diagram should both be at nearly 12 V. When a 20 kHz square wave of 12 V peak is now applied to the modulation input, the switching waveforms should now be obtained. If a variable pulse-width square wave is available it should be possible to control the average current through the resistor by altering the pulse width.

Once the unit is fully assembled and bolted up, the above procedure can be repeated using a DC motor as the load. Speed control should now be achieved by varying the duty cycle of a logic square wave applied to the modulation input.

Fast rise time power switching circuits are notorious generators of RFI, hence our screened diecast enclosure and decoupling capacitors C1 and C101. Ceramic bypass capacitors of 1nF should also be wired directly across the motor terminals (although many motors already have such suppression fitted). For the truly ambitious interference suppressor, very high permeability ferrite beads can be included round the supply and motor leads to absorb stray RFI.

## HOW IT WORKS

**Transistors Q1-4 form an 'H' type bridge output stage. This has the advantage of being able to apply the full supply voltage across the motor in either polarity direction using only a single supply rail, enabling 'solid state reversing' without the use of clumsy relay switching. The transistors used are high power Darlingtons which have a gain of 1000 at 10 A collector current (quite impressive!) and thus require a base drive of about 10 mA for the full rated output. This is a switching amplifier and so the transistors are switched either hard on or fully off; this not only reduces the transistor dissipation for a given power regulation but also the current consumption, so valuable in battery portable applications.**

**The bridge works in the following manner; when Q1 and Q4 are on, Q2 and Q3 are arranged to be off. Point A is now close to +12 V and point B will be near ground, applying nearly 12 V across the motor in one direction. Q4 is now switched on and off by a 'duty cycle' or 'pulse width' modulated square wave. The supply voltage is thus switched on and off across the motor in bursts of varying size; with the help of the motor inductance this produces a controlled average current in the motor.**

**Applying the full supply voltage in short bursts across the motor provides a much higher motor torque than would a constant but lower voltage, which assists greatly in the regulation of low speeds.**

**To reverse the motor, Q1 and Q4 are switched off and Q2 and Q3 now turn on, operating in exactly the same way as described above. The switching frequency we shall be using is around 20 kHz, since anything less produces an infuriating whine from the entire motor assembly.**

**The transistors switch off sharply, removing the source of current from the motor, but the motor current can't change suddenly due to the motor inductance. Thus the circuit wouldn't work if it wasn't for D1 and D2, which act as 'flywheel' diodes. When Q4 switches off, the motor current that was flowing through Q4 to ground now flows in a circle through D2, Q1 and the motor. This is the reason for keeping either Q1 or Q2 continously switched on while Q3 or Q4 provide the switching modulation.**

**The bridge rectifier (D1 to D4) also protects the output transistors from the back EMF switching transients caused by the inductive load. They would be destroyed without this protection.**

**The base resistors R2, R3, R5 and R6 enable the Darlingtons to switch off faster. The base drive and logic selection of the transistors is all provided by the unconventional use of IC1, a dual one-of-four analogue switch (the CMOS 4052B). These switches connect the bases of the appropriate transistors via resistors R1 and R4 to the corresponding supply polarity. If the 'A' and 'B' select lines of IC1 are both low, pin 3 will be connected by the internal 120R resistance to pin 1, effectively taking the base of Q1 to ground through a 300R resistor and turning it hard on. If the 'B' select is now taken high, pin 3 will switch to pin 2 keeping Q1 turned on, and pin 15 will connect to pin 13, taking the base of Q4 to +12 V. Q4 now switches on applying the voltage across the load.**

# PART 2

In this second part we shall describe the design of an analogue pulse width modulator for controlling the motor driver stage featured above. We shall also take a brief look at some of the modules being offered later in the series which can be added in stages to enhance the motorised vehicle. The intention is to build up to a complete computerised mobile.

A lot of flexibility has been allowed for in the actual use and configuration of the modules, as we are well aware that constructors interested in this type of project have firm ideas of their own on the final form and capabilities of their mobile. Construction and interconnection details for all the modules we are presenting will be given along with guidelines to a range of applications.

It is hoped that the designs will also prove useful as stand-alone modules for individual use in other applications. Optical proximity detectors, for example, have numerous applications in batch counting, limit sensing, detection, alarms and so on.

## Optical Proximity Detectors

These have been designed as small independent units with as much in-built versatility as possible. The circuitry is housed in a short length of aluminium tube axially aligned in the detector direction, with three external connecting points; ground, positive supply, and an open collector digital output. A number of detectors can thus be easily mounted in strategic locations. All circuit operating parameters are independent of the supply voltage, which can be anywhere between 5 and 35 V at a current of 20 mA.

The proximity switch works on the principle of transmitting and detecting a modulated infra-red beam. The infra-red transmitter receives 1 A peak current pulses, of 10 uS duration, with a modulation frequency of 1 kHz. The 100:1 duty-factor thus achieved allows high currents to be used to increase the detection range, while reducing the average supply current to only 10 mA.

The sensor can be set by a preset pot, accessible through a small hole, to detect an object at any distance in the range 1 cm to 35 cm.

A small amount of hysteresis is introduced into this switching distance to ensure clean switching thresholds and stability of the output signal. The use of tuned detector amplifiers provides excellent infra-red interference rejection.

## Analogue Speed Control

The analogue speed control has

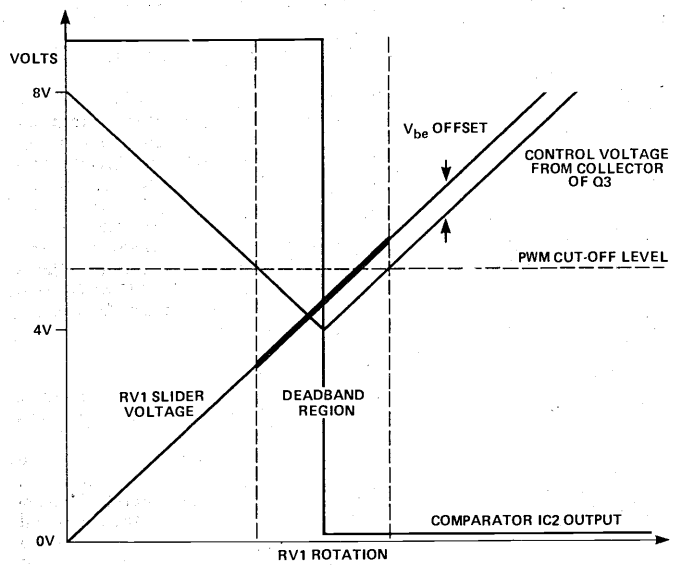



Fig. 1 Various voltages associated with the circuitry around Q3. The control voltage is measured at point A in Fig. 5.

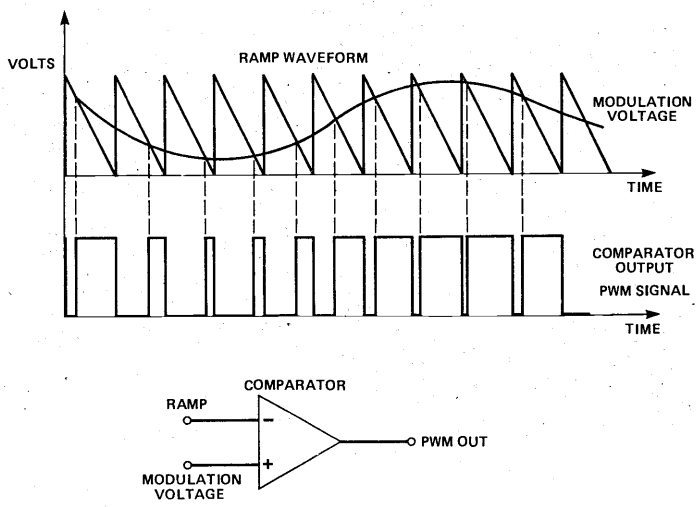



Fig. 3 How PWM waveforms may be generated using a comparator.

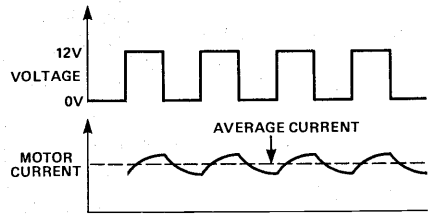



Fig. 2 PWM motor driving waveforms for last month's circuit.

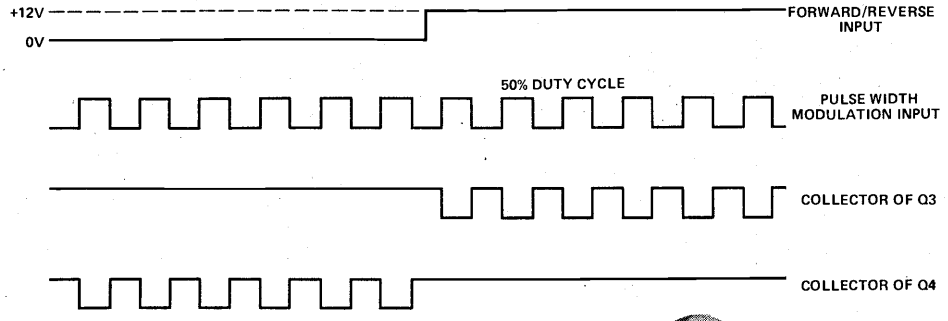


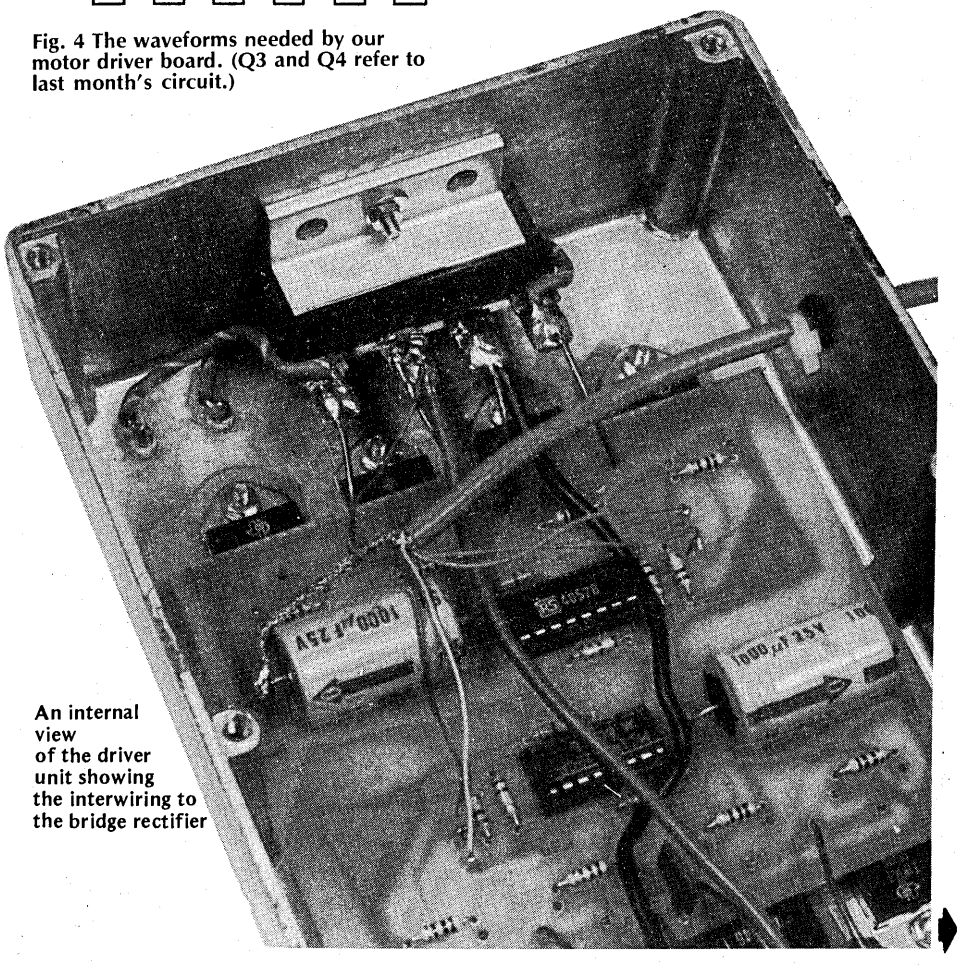

Fig. 4 The waveforms needed by our motor driver board. (Q3 and Q4 refer to last month's circuit.)

been devised for manual control of the main traction motors; it provides two pulse width modulated signals suitable for the motor driver amplifier.

The circuit is designed to provide a linear control-voltage-to-pulse-width relationship for greater flexibility in application, and to simplify the addition of speed feedback velocity control.

The modulator can be built either single or dual, and the manual control section, if not required, is easily omitted. Speed control is achieved via two remote potentiometers, allowing speed to be set in either forward or reverse directions independently for each traction drive.

Since both motors are controlled via switching amplifiers from the same battery supply, it is important to reduce the peak currents that are drawn. This can be achieved by offsetting the phase of the switching waveforms relative to each other, such that at 50% duty cycle modulation, power



An internal view of the driver unit showing the interwiring to the bridge rectifier

## BUYLINES

No problems here with any of the components specified — most mail order companies who advertise in ETI will be able to supply everything. We can supply the PCB.

The circuit for the dual analogue pulse width modulator is shown in Fig. 5; it will be seen that each channel is identical with the exception of the circuitry around the CMOS gates IC1 and IC4. As described earlier the two switching waveforms must be the same frequency and synchronized 180° out of phase, to distribute the motor current peaks more evenly through the cycle. This is achieved by synchronizing both pulse generators to a master clock based around IC1a and b. A 20 kHz square wave is generated by this conventional astable arrangement and its frequency, set by R1 and C1, is fairly independent of supply variations.

The output of IC1d at pin 6 provides a buffered square wave in the same phase as the output on pin 10 of IC1b. C2 and R3 differentiate the positive-going edge of the square wave to produce a very short logic low pulse at the output of Schmitt inverter gate IC1c. In similar fashion C9 and R16 produce a logic high pulse coinciding with the negative-going square wave edge. IC4b further inverts this signal to a logic low pulse. Two separate trains of 500 nS negative-going pulses are thus provided in the correct phase relationship for resetting the charging cycle of two sawtooth oscillators as described below.

The pulse width modulators are identical from here on and we shall refer to the topmost circuit for description. Voltage controlled pulse width modulation is, in principle, very simple; a ramp waveform (sawtooth) is applied to one input of a comparator and the modulation voltage to be encoded is applied to the other, producing the required PWM squarewave at the comparator output. Figure 3 illustrates this operation.

Due to the design requirement of a linear relationship between control voltage and pulse width, a constant current source formed from Q2 is used to generate the linear ramp waveform. LED1 and the base-emitter junction of Q2 are forward biased by R6 and together define a temperature-compensated voltage across R7 which in turn defines a constant emitter and collector current of about 1 mA. C3 is charged up negatively from this current, until the negative-going reset pulse arrives from inverter IC1c. This pulse turns Q1 hard on for a very short period (500 nS), during which C3 is completely discharged, taking the ramp voltage back to +8 V. This process repeats at the clock frequency of 20 kHz, providing a negative-going sawtooth of about 3 V peak-to-peak referenced to the +8 V rail.

IC3b, the comparator used to perform the modulation, is an LF353 dual op-amp, chosen for its large bandwidth and high slew-rate. The inverting terminal on pin 2 is fed from the ramp waveform, while the non-inverting terminal is fed from op-amp IC3a, an inverting amplifier configured to sum control voltage inputs relative to a 4 V reference.

The potential divider R11 and R12 provides the 4 V reference to the non-inverting terminal of IC3a, and the control voltage applied to R13 at point A is summed relative to the 4 V. An offset voltage set by PR1 is also summed at the inverting terminal of IC3a, and is used to bring the control voltage into the correct operating range and for setting a deadband region on the manual control pot RV1.

The output of op-amp IC3b (and indeed most others) will not swing to the full supply rail voltages, so the inverter gate IC1e is used to buffer the square wave to full CMOS logic levels.

The manual control system included in this circuit enables a single potentiometer to control the speed in both forward and reverse directions. When the pot is at centre travel, and for a certain deadband around this point, the motor must be stopped and no switching pulses should occur (ie the PWM signal is continuously low). As the pot is turned in either direction from its midpoint, the pulse width should in-

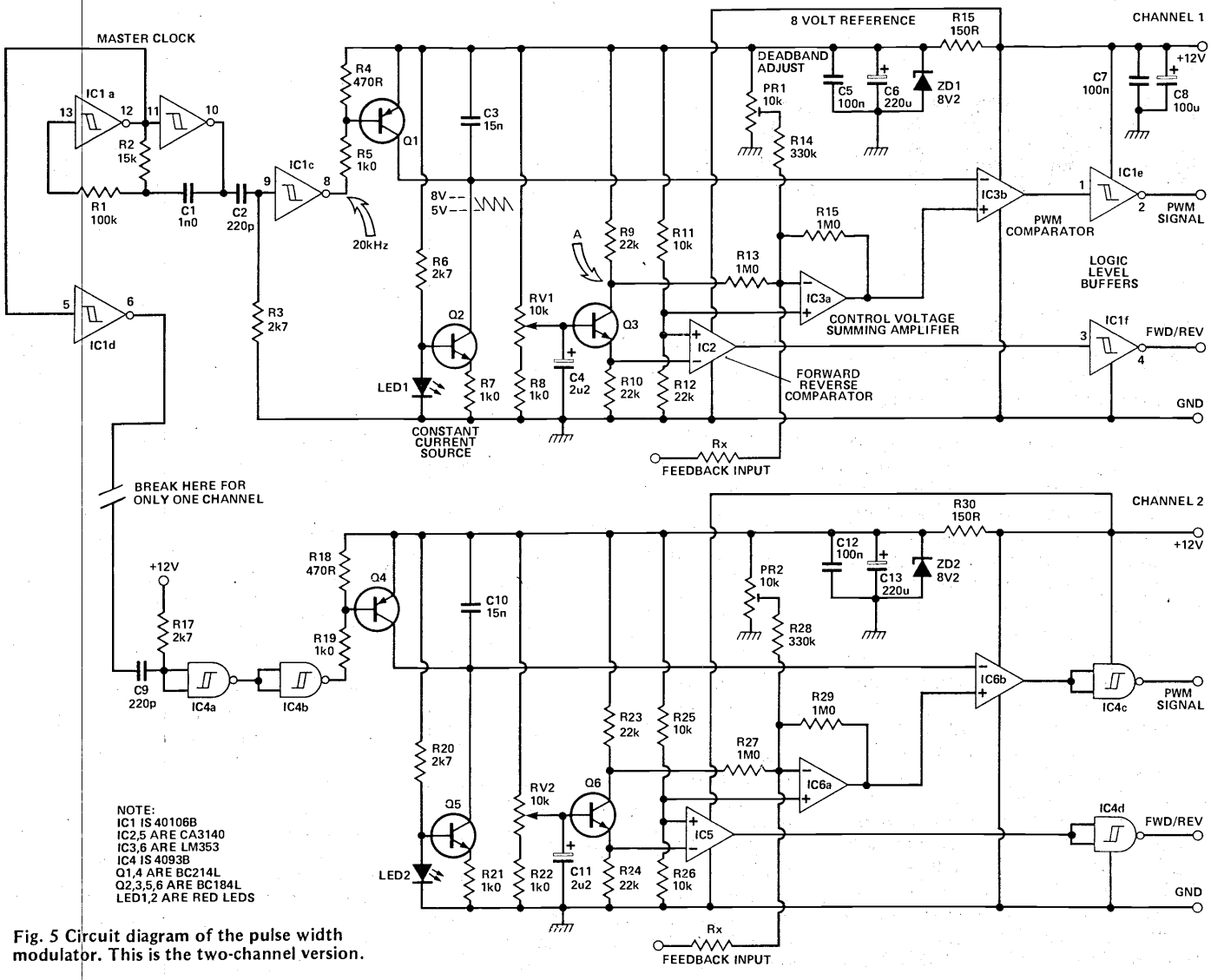



Fig. 5 Circuit diagram of the pulse width modulator. This is the two-channel version.

crease and this requires a positive-going input voltage to the summing amplifier IC3a. The forward/reverse logic level should also change state as the pot moves through its midpoint. Q3 provides the necessary voltage transfer function from the pot RV1 to the control voltage summing amplifier, as explained graphically in Fig. 1.

The emitter and collector resistors of Q3 are both equal and the base voltage is taken directly from the slider of the manual control pot RV1. The output voltage is taken from the collector of Q3 to feed the summing amplifier, and will be held at +8 V via R9 when Q3 is switched off. As the slider of RV1 moves toward the centre of travel, the base voltage rises, slowly turning on Q3 and lowering the collector voltage.

When Q3 is turned hard on as RV1 reaches its mid-point, R9 and 10 will form a potential divider giving 4 V as the minimum control voltage. Further increase of base voltage can now only increase the emitter and collector voltages back up to the positive rail, reaching a maximum at one $V_{be}$ drop from the +8 V rail.

During the above process the voltage on the emitter of Q3 rises from zero to the same maximum voltage, and is fed to the inverting terminal of IC2, a CA3140 used as a comparator. The other comparator input receives 4 V derived from the potential divider R11 and R12. This provides the required forward/reverse signal that corresponds to each half of the control pot. Inverter gate IC1f buffers the output of IC2.

C7 and C8 provide supply decoupling for both channels, while C5 and C6 provide further smoothing for the 8 V zener regulator formed by R16 and ZD1. This 8 V reference rail is used for two reasons; firstly to allow for fluctuation in the 12 V battery power supply that would otherwise affect the output pulse width, and secondly to ensure that the op-amp supply voltage is well above the maximum input voltage.

The resistor marked as Rx in the circuit shows where a speed feedback voltage will be added to the controller to close the velocity control loop. An infra-red tachometer module to directly sense the traction speed will be described later in the series.

If the manual control input is not required, the components associated with this can be simply omitted (ie RV1, R8, R9, R10, C4, Q3, IC2 and their equivalents in the other channel). Control voltages may now be fed to the unconnected end of R13, where a variation of 3 V, set by PR1 to be anywhere in the range 0 V to 8 V, will provide 100% control of the output pulse width. Forward/reverse switching must also be applied to the input of IC1f on pin 3.

## BUYLINES

You shouldn't have any supply problems with the components for this project — everything is absolutely standard. The case we used is a Vero type 65-2514F and should be available from any Vero stockist. We can supply the PCB.

will be switched alternately to each motor. This spreads the current peaks more evenly over the switching cycle.

# PART 3

The overlay diagram for the analogue pulse width modulator described last month is shown in Fig. 2. Assemble the PCB following the component orientations indicated and soldering in sockets for the ICs. If only one channel is required, the board can be cut in half along the dividing line, omitting the two links and assembling the circuit for channel 1. Overlay pictures for the transistors are correct for the specified 'L' versions and pin outs should be identified carefully if other types are used.

Veropins should be soldered in at all the points marked for terminations (18 in all). Spare positive and ground terminals have been included near the input and output points for flexibility. The resistor shown on the circuit diagram in last month's issue as the speed feedback input and marked as Rx on the overlay, should be included on the board; a value of 100k is required. A 1M0 shorting resistor should also be connected between the terminal pins marked for speed feedback input (GND and INPUT). This resistor prevents stray interference in the basic unit and can be easily removed for later addition of closed loop velocity control.

After mounting the two presets they should be set at about half-travel. (Ceramic base presets can crack quite easily, so take care when inserting these into the PCB.) These presets are used to apply a DC offset to the voltage summing amplifier, so shifting the control voltage range to the required level.

## Installation

Without plugging in the ICs, a 12 V power source can be connected to the board as shown. The power rails should be checked at the relevant pins on the IC sockets; also check that the 8 V reference is present at the positive end of C13 and C6. If all is well, the power can be disconnected and the ICs plugged in. At this stage the board can now be mounted using brackets or insulating pillars directly above the power switching PCB in the diecast box. The 12 V power source is taken from the existing bridge-rectifier tags, and the two output signals, PWM and FOR/REV, should be wired to the corresponding pins on each channel of the power switching PCB. Have a look at the internal photographs with the analogue board.

The two manual control potentiometers, RV1 and 2 can now be wired up to the control voltage input. In the circuit diagram the positive ends of RV1 and 2 are shown connected to the +8 V reference; a four core cable would thus be needed for the remote attachment of these potentiometers with their limiting resistors R8 and R22. Essentially all that's required is a voltage variable from 0 to 8 V applied to the base of Q3, so to allow the use of cheaper twin core screened cable we mounted our pots in a small hand-size box with their own 9 V PP3 battery power source. A five pin DIN plug and socket mounted at one end connects the control voltages from the pot sliders, via the cable, to the modulator PCB. The cores are wired to the input terminals at the bases of Q3 and Q6,

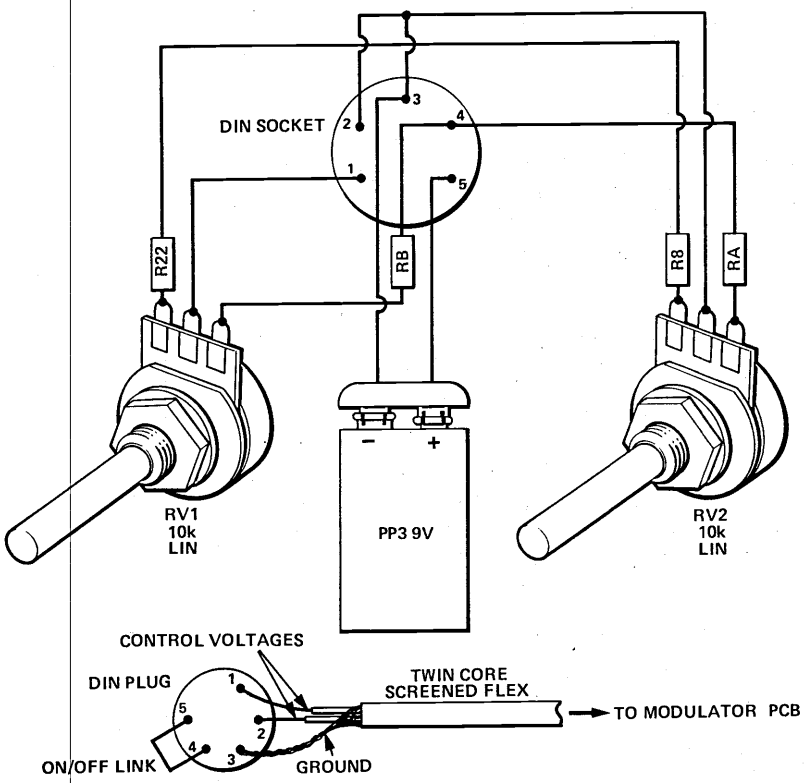
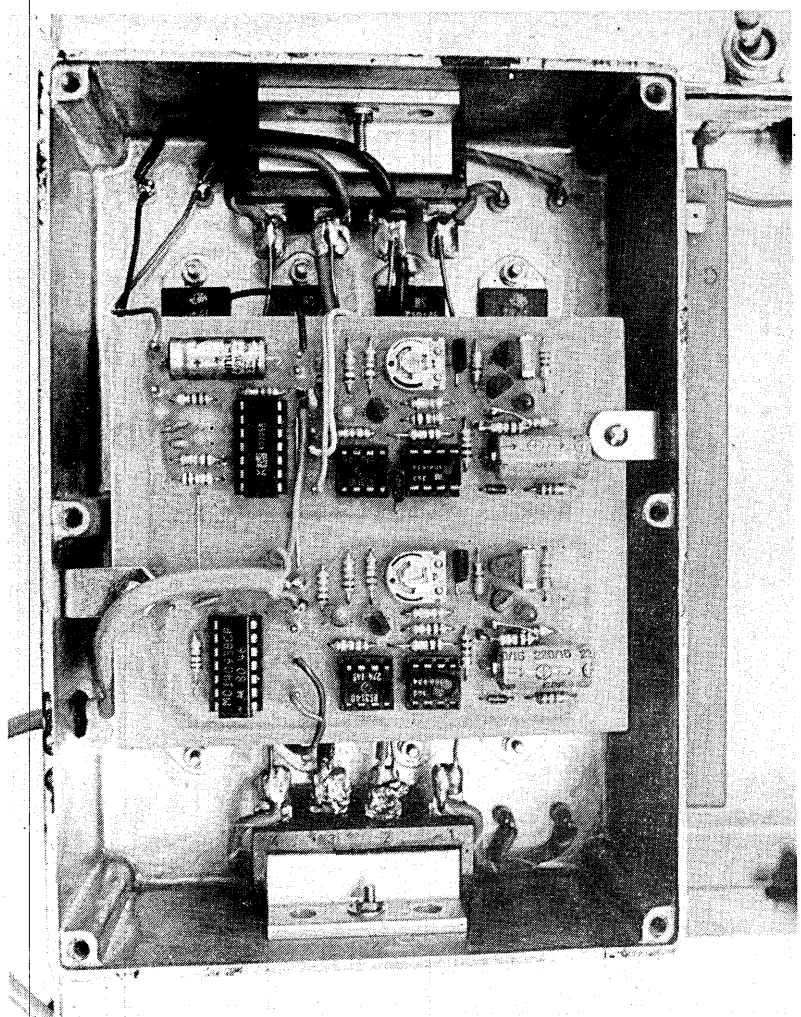



Fig. 1 Wiring diagram for the manual control unit.



The analogue PWM board installed above the dual motor driver.

## PARTS LIST

MANUAL CONTROL BOX

Resistors RA and RB; 1k0 (¼W, 5%) (see text)
Case (see Buylines)
5 pin DIN plug and socket
PP3 9 V battery
Twin core screened cable (length to suit remote operation)
Two control knobs

with the screen taken to the adjacent ground pin. Figure 1 illustrates the manual control wiring, the accompanying photo the internal appearance.

R8 and R22 are shown on the circuit diagram as 1k0; these may be altered as required to limit the maximum reverse speed. RA and RB however are optional resistors for limiting the maximum slider voltage to 8 V; we used 1k0 resistors to suit the 9 V battery.

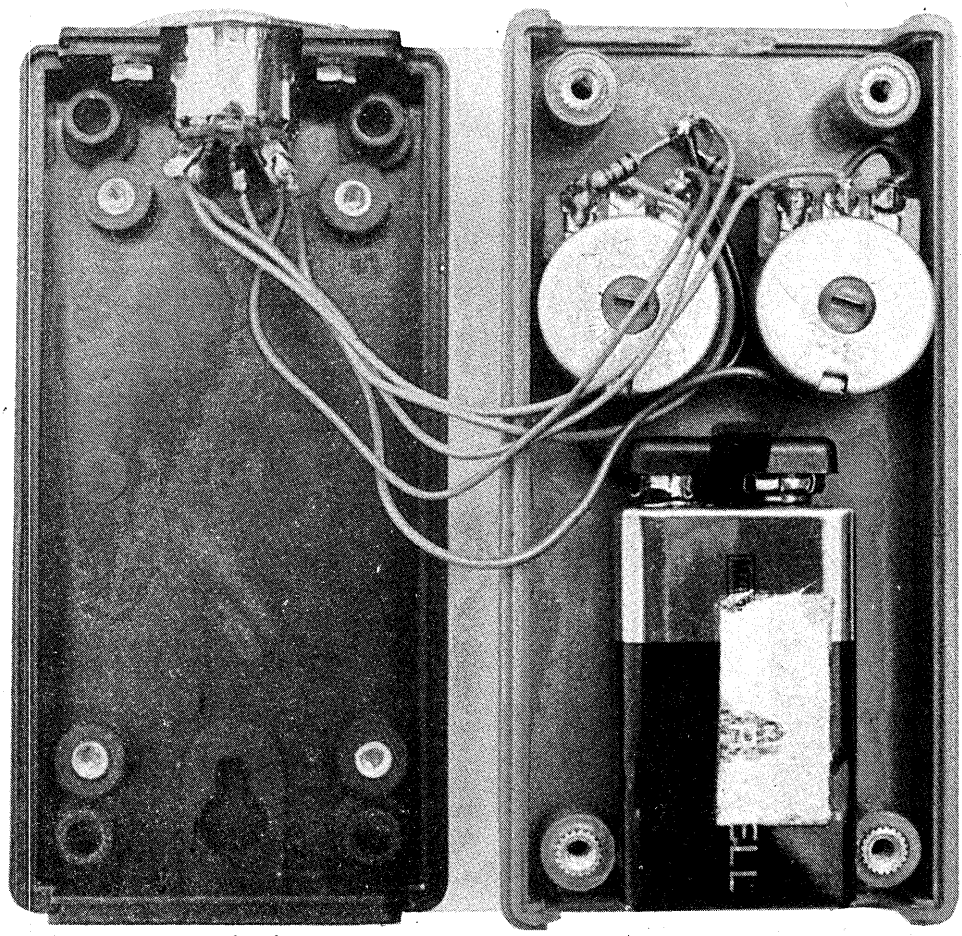
If this type of manual forward/reverse control is not required the associated components can be omitted during assembly (C4, C11, Q3, Q6, IC2, IC5, R9, R23, R10, R24). Pin 6 of sockets IC2 and 5 now becomes the FWD/REV logic control, the unused collector pad of Q3 and 6 becomes the control voltage input.

The ratio of R15 to R13 determines the gain of this input, and with the specified 1M0 values is set at unity. Thus an input variation over a 3 V range will provide 0% to 100% duty-cycle control. The input should be referenced to the 0 V ground, and PR1 and 2 can provide full offset adjustment of the control ranges. To increase the gain, the value of R13 should be decreased.

### Testing

Once the manual potentiometers have been wired up in a suitable fashion the completed controller may be set up for proper operation. Temporarily disconnect the PWM signal wires and solder them to the adjacent ground terminals; this will prevent the power stages being damaged if there are any errors.

Connect a 12 V supply to the main controller; two glowing LEDs are the first signs of success! A voltmeter or scope set for 12 V FSD should now be hooked up to the FWD/REV output; depending on the pot position a 0 V or 12 V level will be present, and should sharply change state as the pot is turned through its centre travel. A ramp waveform of 3 V peak should be observed if a scope is put on pin 6 of IC3 or IC6. The PWM output can now be measured for each channel with the meter, indicating a voltage



The manual control unit.

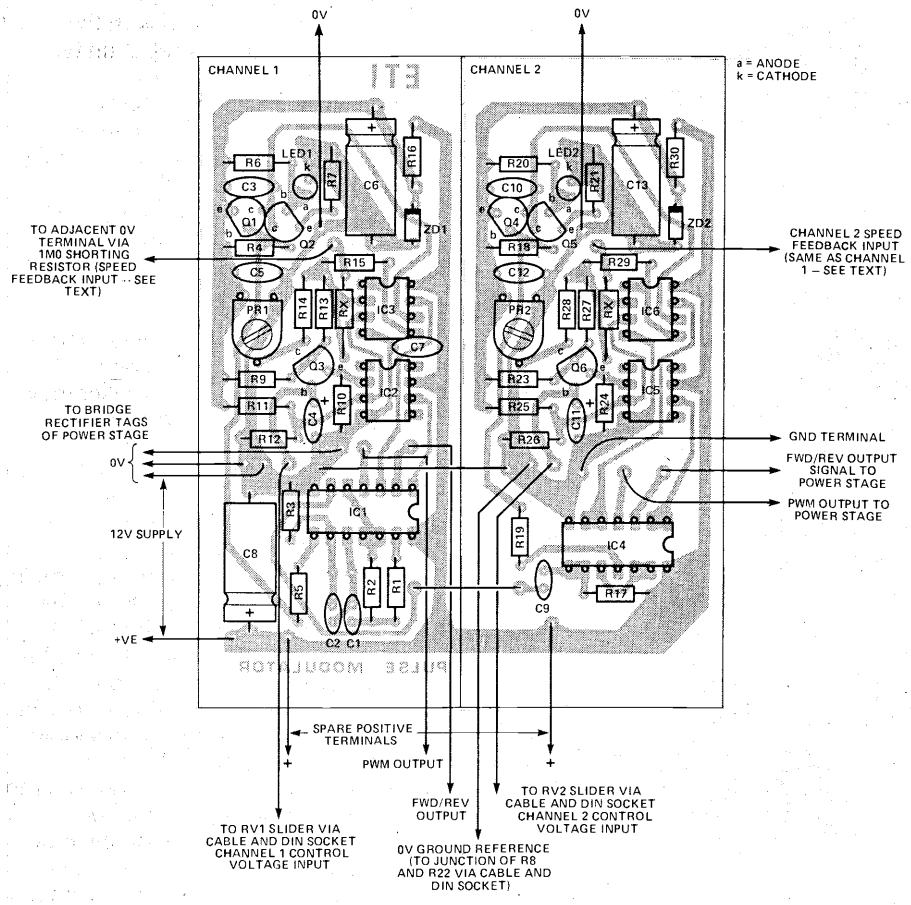



Fig. 2 Component overlay for the analogue PWM unit.

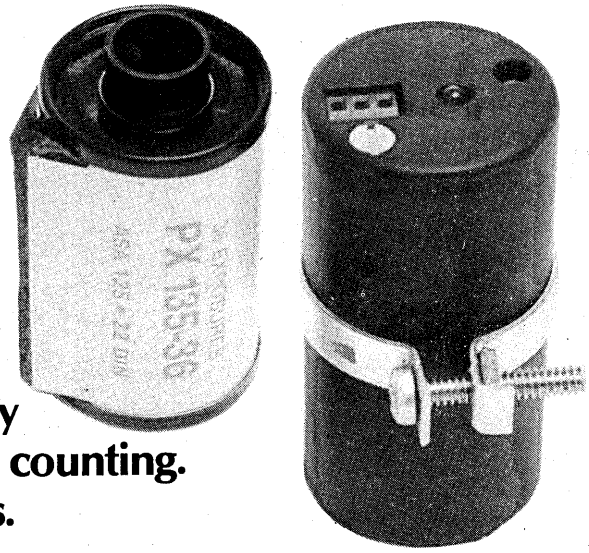
proportional to duty-cycle, or a scope to show the pulse waveform. For a given position of the control pot, clockwise adjustment of the corresponding preset will increase the duty cycle. Leaving the preset and now turning the manual pot towards centre travel will decrease the duty-cycle linearly until the pulse width vanishes, giving a dead band of 0 V. Turning the presets again, anticlockwise this time, will increase the deadband. A small proportion of the pot rotation should be left as deadband, so enabling the motor stop position to be easily located. Adjustment is more difficult with a meter than a scope, but with patience the desired setting can soon be achieved.

The PWM signal wires can now be reconnected to the outputs and the chosen motors wired across the power amplifier load terminals. A 20 A toggle switch wired in series with the positive battery supply is strongly recommended at this stage. (WARNING: If the motors used are rated at less than 6 V, the maximum obtainable duty-cycle should be limited accordingly by increasing the deadband; good motors are usually expensive.) A 12 V bulb will provide a good motor substitute for the purposes of testing. Each manual pot will now independently control the average power into any load; bidirectionally!

## PARTS LIST

| | |
|---|---|
| **Resistors (all ¼W, 5%)** | |
| R1 | 100k |
| R2 | 15k |
| R3,6,17,20 | 2k7 |
| R4,18 | 470R |
| R5,7,8,19, 21,22 | 1k0 |
| R9,10,23,24 | 22k |
| R11,12,25,26 | 10k |
| R13,15,27,29 | 1M0 |
| R14,28 | 330k |
| R16,29 | 150R |
| **Potentiometers** | |
| RV1,2 | 10k linear |
| PR1,2 | 10k linear miniature horizontal preset |
| **Capacitors** | |
| C1 | 1n0 ceramic |
| C2,9 | 220p ceramic |
| C3, 10 | 15n polycarbonate |
| C4, 11 | 2u2 35 V tantalum |
| C5, 7, 12 | 100n ceramic |
| C6, 13 | 220u 16 V axial electrolytic |
| C8 | 100u 25 V axial electrolytic |
| **Semiconductors** | |
| IC1 | 40106B |
| IC2,5 | CA3140 |
| IC3,6 | LF353 |
| IC4 | 4093B |
| Q1,4 | BC214L |
| Q2,3,5,6 | BC184L |
| LED1,2 | red LED |
| ZD1,2 | 8V2 400 mW zener diode |
| **Miscellanous** | |
| PCB (see Buylines) | |

# PROXIMITY DETECTOR



**This project will endow your man of steel with infra-red vision; and it's not much bigger than a human eyeball. Alternatively you can use in applications such as batch counting. Design and development by Rory Holmes.**

This project provides a very useful means of detecting the presence of *anything* by the reflection of infra-red light, and provides a direct digital output of object detection.

The transmitter and receiver of the infra-red beam are both mounted on the same miniaturised PCB, which is housed in a short length of aluminium tube for screening and protection. By the use of modulation and high power bursts of infra-red at a very low duty cycle, a detection range of over a foot is achieved. The receiving photo-amplifier is tuned to the same modulating frequency of 1 kHz, and thus provides good rejection of stray infra-red interference. Bright lights will not affect the operation of the module.

The module features a wide supply voltage range, with an LED to indicate correct operation. A preset adjustment pot at the rear of the sensor allows the detection range to be preset at any distance between 1 and 35 cm.

## Construction

Although the PCB layout (Fig. 2) is quite dense, with several vertical mounting resistors, the assembly should be straightforward. The only component of note is PR1, a ¾" 20 turn rectangular cermet preset. These are readily available, though, and should fit the board exactly. The power transistor Q3 is mounted flat, with the metal side face down; likewise, observe the orientation of the other transistors. Photodiode D1 has a chamfered edge on one side; this is mounted to face the infra-red emitter LED2, so allowing the sensitive surface to face outwards. The infra-red LED should be mounted with the flat side facing away from the photodiode (the flat identifies the cathode). After assembly of the board it is important to mount a small guard between the infra-red emitter and detector, to prevent infra-red light passing directly to the detector before it has been reflected. The guard should be a 7 mm square, cut from un-etched PCB or a piece of aluminium. It can be stuck between the two diodes and directly in front of C4 with a blob of superglue.

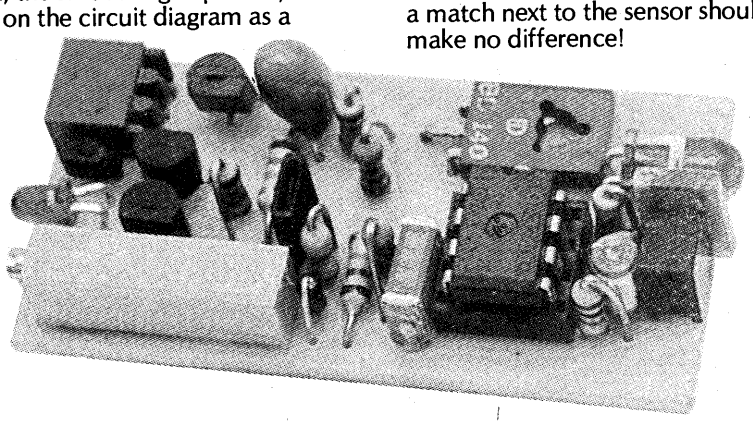
The board is mounted in a 55 mm length of aluminium tube, of internal diameter 27 mm or greater.

The diagram of Fig. 3 illustrates how a 6BA nut is soldered sideways onto the PCB track directly beneath the 3-way connector socket. Holes are drilled in one end of the tube to mate up with the indicator and preset adjustment screw. A rectangular hole also needs to be cut, allowing access to the connector socket. A 6BA bolt can now be used to tighten the board against the tube end. The sensing end of the tube may be covered with anything that is transparent to infra-red (red filter plastic polarising sheet, or just clear plastic). If openings are cut for the emitter and detector then an aluminium disc could also be used. The disc should be cut to fit the tube and mounted flush against the small guard plate with epoxy. The sensor tube may be mounted with a circular clamp that tightens round the tube; this can be seen on the photographs of our prototype.

C2, the smoothing capacitor, is shown on the circuit diagram as a 100uF 10 V tantalum electrolytic. This value was chosen to fit on the PCB and consequently limits the supply to 9 V maximum, although the circuitry is capable of operating up to 35 V. To allow higher supply voltages, change C2 to 22uF 35 V tantalum. An additional electrolytic of 100uF 40 V should be mounted underneath the board and soldered to the same pads as C2.

The sensor is now ready for testing, and the three way connector plug should be wired to a suitable power source capable of providing 100 mA (this is for the benefit of the bulb, if used; the circuit itself only takes 20 mA). A PP9 9 V battery is adequate. One of the test circuits illustrated in Fig. 4 should be adopted; if the LED arrangement is used, for example, the LED will be on when the sensor points into free space. Start with the preset fully anticlockwise; this gives minimum sensitivity and the sensors should not trigger at all.

Keeping the sensor pointed at free-space, the preset should be turned clockwise to increase the sensitivity until the LED just goes out. The preset should now be backed off until the LED just comes on again, thus setting the maximum range. Placing a hand about 12" in front of the sensor will now turn off the test LED, while striking a match next to the sensor should make no difference!



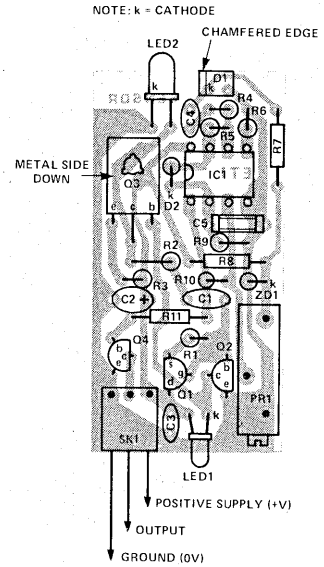



Fig. 1 Component overlay for the optical sensor. The photos overleaf show how small the unit is.

## HOW IT WORKS

The proximity sensor works on the principle of transmitting a beam of modulated infra-red light from the emitter diode LED2, and receiving reflections from objects passing in front of the beam with a photodiode detector D1. The circuit can be split into three distinct stages; the infra-red transmitter, the photodiode amplifier, and a variable threshold comparator.

The transmitter provides 1 A peak current pulses for 10 uS through the infra-red emitter diode, at a repetition rate of 1 kHz. Q1 is arranged as a constant current source to supply the base of Q2, and to charge up C1. As C1 charges up, the base voltage of Q2 rises until it reaches about 0V6 relative to ground. Q2 then turns on, so turning on another constant current source formed by Q3 and LED1. This current source sets a temperature compensated voltage of about 1V5 across R3, thus defining a current of 1 A through the infra-red emitter LED2. After Q3 turns on, a negative pulse through C1 turns off Q2 again, thus re-starting the oscillation cycle. The current pulse, determined by C1 and R2 is set at 10 uS. A 10 uS pulse every 1 mS is equivalent to a duty factor of 1:100, so that although 1 A peak pulses are generated, the average current required is only 10 mA. Capacitors C3 and C2 are there to provide power supply smoothing to decouple the fast current pulses.

The detector is built around IC1, a CA3240 dual op-amp. IC1a is configured as an inverting amplifier with a gain of −2. It amplifies the infra-red signal picked up by photo-diode D1. C4, which couples the diode signal to IC1a, acts as a high-pass filter in combination with the input impedance of the amplifier. Positive-going pulses of 10 uS duration are fed from the output, via rectifier D2, to a smoothing filter C5 and R9. This provides the signal voltage reference for the inverting input of comparator IC1b. A 2V7 reference, formed by R8 and ZD1, provides the biasing voltage for the photodiode through R7. It also provides the reference voltage for the non-inverting comparator input, set by potential divider PR1. R10 creates positive feedback round the comparator, to improve the switching, and introduce a small amount of hysteresis. Thus, if a reflected light signal received due to the presence of an object rises above the threshold set by PR1, the comparator output will go into negative saturation. The comparator output is used to turn Q4 on or off, thus providing an open collector output for digital interfacing to logic circuits.

NOTE:
IC1 IS CA3240
Q1 IS 2N3819
Q2,4 ARE BC184L
Q3 IS BD140

D1 IS PHOTODIODE
D2 IS 1N4148
ZD1 IS 2V7 400mW ZENER
LED1 IS 3mm RED LED
LED2 IS IS INFRA-RED LED

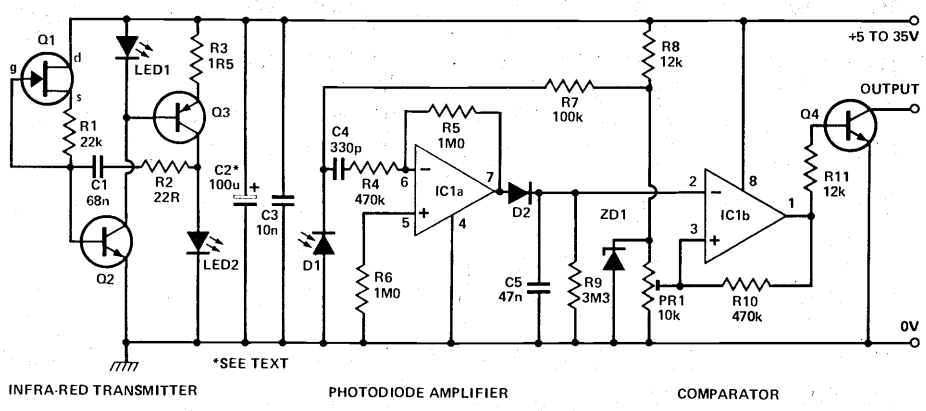



Fig. 2 Circuit diagram for the sensor.

## PARTS LIST

| | |
|---|---|
| Resistors (all ¼W, 5%) | |
| R1 | 22k |
| R2 | 22R |
| R3 | 1R5 |
| R4,10 | 470k |
| R5,6 | 1M0 |
| R7 | 100k |
| R8,11 | 12k |
| R9 | 3M3 |
| Potentiometers | |
| PR1 | 10k ¾" 20 turn cermet trimmer |
| Capacitors | |
| C1 | 68n ceramic |
| C2 | 100u 10 V tantalum |
| C3 | 10 n ceramic |
| C4 | 330p ceramic |
| C5 | 47n polycarbonate |
| Semiconductors | |
| IC1 | CA3240 |
| Q1 | 2N3819 |
| Q2,4 | BC184L |
| Q3 | BD140 |
| D1 | Photo-diode (TIL100 or similar) |
| D2 | 1N4148 |
| LED1 | 3 mm red LED |
| LED2 | infra-red LED (TIL38 or similar) |
| Miscellaneous | |
| PCB (see Buylines); three-way PCB plug and socket (see Buylines); aluminium tube (27 mm diameter, 55 mm long). | |

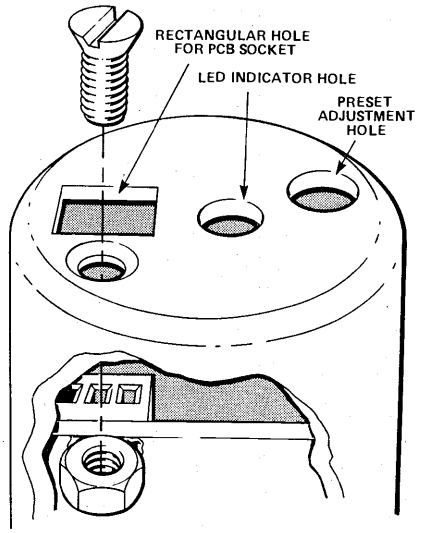



Fig. 3 This is how we mounted the PCB in the aluminium tube. The nut is soldered to the ground track under SK1.

## BUYLINES

The only out-of-the-ordinary thing used in this project is the inter-PCB plug and socket (also known as the KK system). This, of course, is not essential, but if you want to use it for neatness it can be obtained from Watford, or via your local supplier from RS Components. We are selling the PCB.

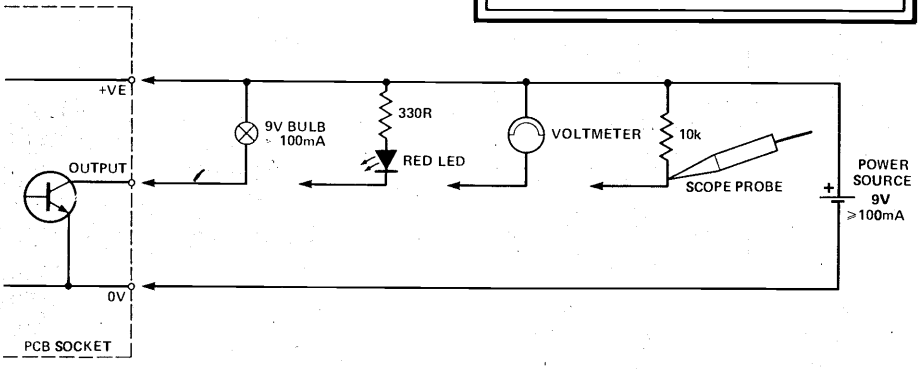



Fig. 4 Any of these test circuits may be used to check out the sensor.

# ROBOT MODULES

## Here is the digital PWM interface board for the motor controller so your computer can take your robot for a walk. Design and development by Rory Holmes.

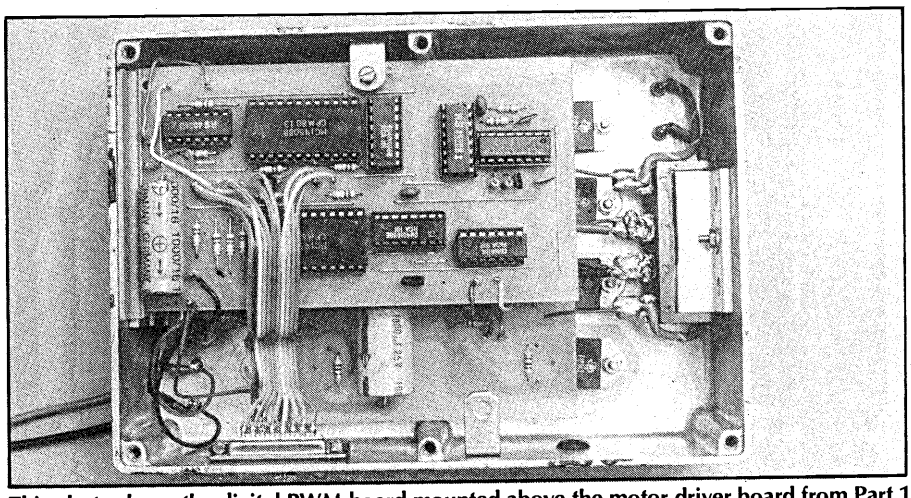
The digital pulse width modulator is constructed entirely from CMOS logic ICs to provide pulse control of two independent channels. Pulse width modulation can be used to control a variety of analogue functions; this design is offered to allow computer control of ETI's Dual Motor Switching Amplifier.

Binary counters are used to count both the ON and the OFF periods of the mark-space cycle; thus the duty cycle is independent of the timing circuitry, being an exact ratio of two numbers.

The duty cycle data for either channel is loaded through an eight bit port common to both, and is addressed to a particular channel by the use of two strobe lines. When a strobe line is taken low the input data is latched into the associated control channel, which generates a pulse width corresponding to the input value. The pulse width will remain at this value after the data has changed and until its strobe line is activated again. The minimum strobe pulse required is 50 uS; thus both motor speeds can be changed in about 100 uS.

To suit the standard eight bit data bus of most microprocessors we have allowed seven bits for controlling the pulse width, with a further bit for setting the forward/reverse motor direction. Pulse widths can thus be obtained with a resolution of one part in 128. With the modulation frequency set at 20 kHz (just above audibility), the pulse width can be increased from zero in 390 nS increments up the maximum of 50 uS; this represents 100% duty cycle modulation (50 uS = one cycle at 20 kHz).

Timing signals for pulse generation can, of course, be implemented directly with software from the controlling microprocessor; however, this creates large overheads on processor time. Furthermore, any application software must be designed around these timing routines which can soon become a nightmare of interactive real-time programming.



**This photo shows the digital PWM board mounted above the motor driver board from Part 1 of the series.**

Our design strategy has been to dedicate external hardware circuitry to the mundane and repetitive tasks, thus freeing the controlling processor for running the more sophisticated command programs.

Figure 2 shows the circuit diagrams of both channels with the accompanying description. The unit is assembled on a PCB which fits inside the existing motor driver amplifier case and runs off the same 12 V power source. A 15-way D-type cannon socket mounted on the side of the diecast box provides input for the programming data.

### Construction

The PCB should be assembled as illustrated in the overlay diagram. We recommend soldering in the 14 links first, followed by the resistors and IC sockets. Veropins should be inserted at all the input and output terminals, 17 in all, and finally the capacitors can be soldered in. Before plugging in the ICs, connect a 12 V power source at the supply points shown and check the ground and positive rail voltages at all the IC sockets. If all is well, a short length of 11-way ribbon cable can be wired up to the eight input bits, ground, and the two strobe lines.

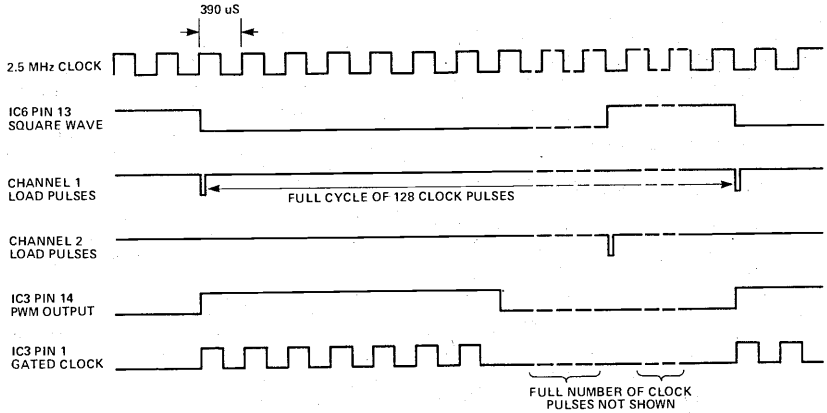



**Fig. 1 Timing diagram for the digital pulse width modulator.**

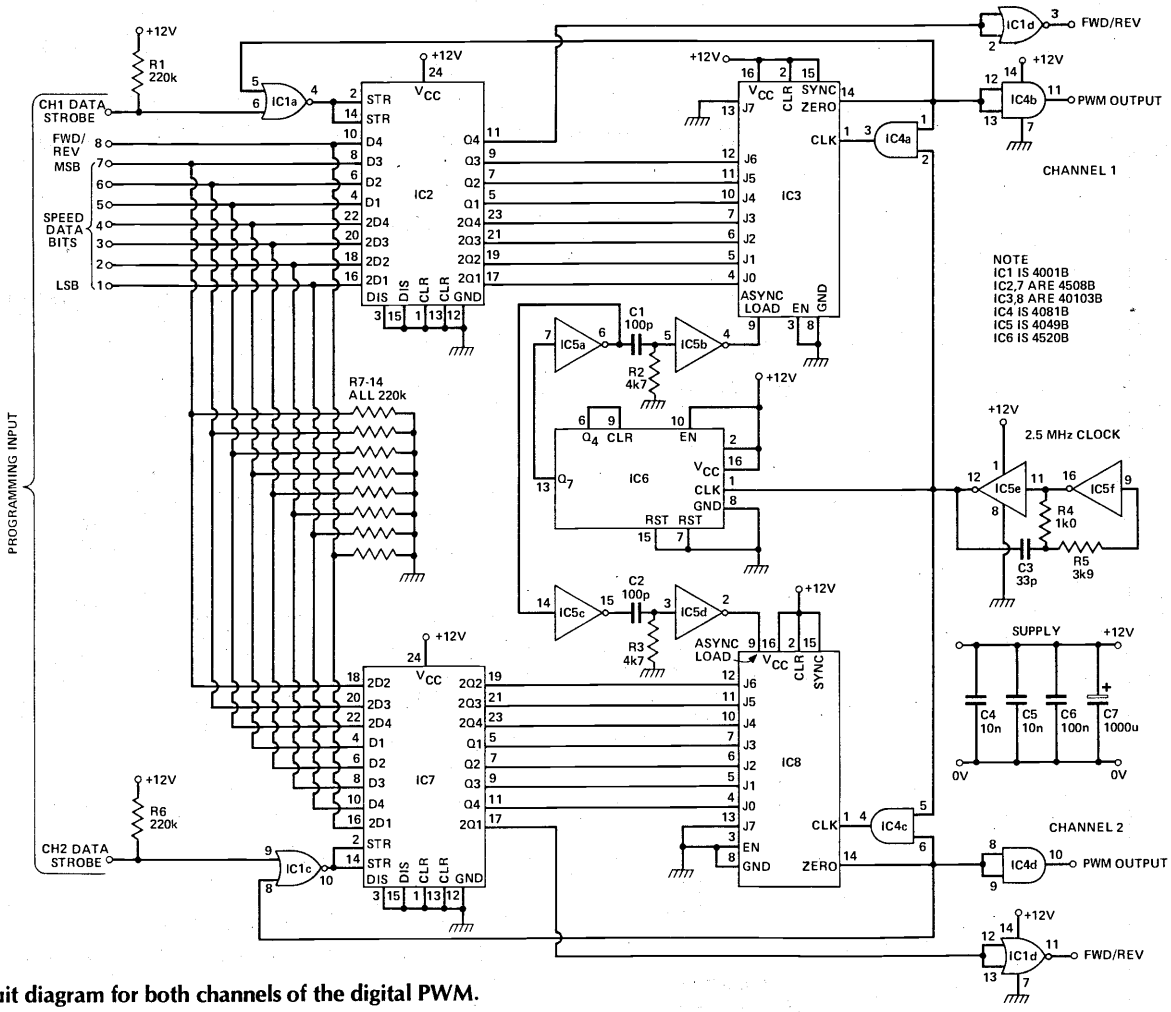



Fig. 2 Circuit diagram for both channels of the digital PWM.

## HOW IT WORKS

Each of the two control channels operates in an identical fashion with some circuitry common for both. We shall describe the basic circuit action referring to the upper half of the circuit diagram in Figure 2. IC3 is the heart of this circuit, being an eight bit binary down counter with presettable data inputs; it is used to generate the variable pulse widths. The eight data bits presented at the preset inputs J0 to J7 are loaded into the counter when the synchronous load input on pin 9 is pulsed to logic low. If a clock is now fed to pin 7 the counter will start counting down to zero from this preset value. When zero is reached the previously high zero detect output on pin 14 goes low. In our configuration this zero output is used to gate the clock input via AND gate IC4a. Thus, as soon as zero is reached the clock is disabled, leaving pin 1 low and so allowing the zero output on pin 14 to remain low also. The counter IC3 can only be restarted by another load pulse to pin 9, which presets the count start value and returns the zero output to logic high, thus enabling the clock for the down count. A look at the timing diagram of Fig. 1 should clarify this sequence.

The 'zero detect' output on pin 14 directly provides the required PWM signal and is buffered by AND gate IC4b before driving the power switching stage. The load pulses are produced by the edge detector implemented using inverter gates IC5a and b. These negative-going pulses are very narrow, with a width set by differentiator C1/R2 of about 200 nS; they are derived from the negative-going edges of a square wave produced by the counter IC6. The load pulses for the other channel are generated in a similar fashion, but from the positive-going square wave edges. This ensures the 180° relative phase offset of the two PWM signals, necessary for reducing the peak supply current.

This square wave determines the full period of each pulse width cycle and thus sets the constant frequency of the PWM output. The counter IC6 is a dual four-bit binary counter; the stages are wired in cascade for ripple counting, and the square wave is taken from the seventh output bit. A 2.5 MHz astable clock built around IC5e and f drives both IC6 and the down counter IC3. IC6 will thus divide the master clock by 128, providing load pulses at a repetition frequency of 20 kHz which is the required modulation frequency. For a maximum duty cycle modulation of 100% the down counter must have a maximum start count of 128, giving a resolution of one part in 128. This is achieved by using only the first seven input bits of IC3 and taking the eighth to logic low. The pulse width is thus variable from 0 to 50 uS in 400 nS increments. The eighth bit of the input port is used to set the foward/reverse direction of the motor.

The pulse width data for the preset inputs of counter IC3 must be continuously available, since it is loaded afresh for each cycle on the negative-going edge of the asynchronous load pulse. The eight bit data port from the 'outside world' is thus latched by IC2, a dual four bit hold-follow latch (the CMOS 4508). When the 'store' inputs on pins 2 and 14 are logic high the data appearing at the Q outputs will follow the data on the inputs; when 'store' is taken low the current data is held internally and remains on the Q outputs to control the down counter.

The corresponding data input bits of each control channel on IC2 and 7 are wired together to form a common input port; data is thus altered for a particular motor channel by taking its associated store input to logic high. Strobing in new speed data is achieved using IC1a, a NOR gate. One input, the strobe line, is held normally high by R1, and the other input is taken from the PWM output.

When the strobe line is now taken low the store input will only go high as the PWM output signal goes low. This arrangement ensures that any new data will be stable when it is loaded into the down counter; it also prevents the forward/reverse control from changing state while the motor driver transistors are turned on. Once the required data has been set up on the inputs, the data strobe line must be taken low for a minimum period of 50 uS to ensure the data has been latched. NOR gate IC1b is used to buffer the forward/reverse control line before it leaves the PCB. The 12 V power requirement is taken from the motor driver stage described in Part 1, with C7 providing smoothing and C4, 5, and 6 decoupling the fast switching pulses.

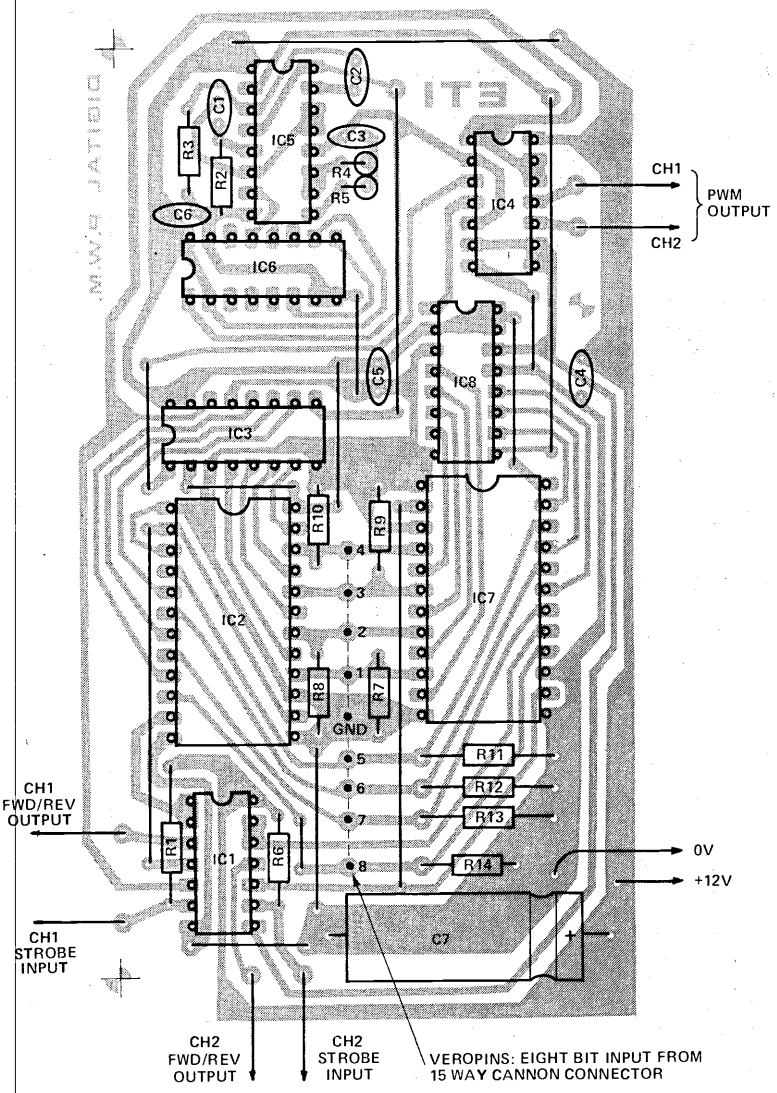



**Fig. 3 Component overlay for the PWM board.**

## PARTS LIST

**DIGITAL PWM**

**Resistors (all ¼ W, 5%)**

| | |
|---|---|
| R1,6-14 | 220k |
| R2,3 | 4k7 |
| R4 | 1k0 |
| R5 | 3k9 |

**Capacitors**

| | |
|---|---|
| C1,2 | 100p ceramic |
| C3 | 33p ceramic |
| C4,5 | 10n ceramic |
| C6 | 100n ceramic |
| C7 | 1000u 16 V axial electrolytic |

**Semiconductors**

| | |
|---|---|
| IC1 | 4001B |
| IC2,7 | 4508B |
| IC3,8 | 40103B |
| IC4 | 4081B |
| IC5 | 4049B |
| IC6 | 4520B |

**Miscellaneous**

PCB (see Buylines); Veropins.

## BUYLINES

**No problems at all here; most mail order companies advertising in this issue should be able to supply the components.**

## Testing Times

With nothing connected to the ribbon cable inputs, the two pulse width modulation outputs should be continuously low and the two FWD/REV outputs should be high (you may need to briefly touch the strobe input wires to a ground terminal). These outputs may be observed with either a 12 V FSD meter or a scope. If a scope is used, the 2.5 MHz clock can also be checked; a square wave should be observed on pins 2 and 5 of IC4.

Pulse width control for each channel may now be verified. Wire bit seven of the ribbon cable to the positive terminal. When either strobe is now taken briefly to ground, the corresponding PWM output line will change to a square wave. The duty cycle is now 50%, this ratio being 64 (bit 7) divided by 128 (full cycle), and the meter will read 6 V. If a scope is being used the square wave pulse trains from each channel can be observed; they should be 180° out of phase with each other (in the case of square waves this looks like inversion).

This process can be repeated, taking only bit 1 positive, to give the minimum pulse width of 1/128th of the full 50 uS cycle (about 390 nS). This very small pulse cannot be detected by the meter but can be seen on an oscilloscope. Further bit combinations can be strobed in, to provide varying pulse widths in increments of 390 nS.

## Modifications

The eighth bit of the data input port is used to set the forward/reverse signal line. For applications that do not require a separate direction signal, ie a computer-controlled pulse generator, this bit may be utilised for higher resolution pulse width control; the full eight bits give a resolution of one part in 256. To implement the modification the following changes can be made.

The tracks to pins 1 and 2 and pins 12 and 13 of IC1 should be cut; this provides two unused NOR gates, whose input pins should be connected to the nearest ground track via insulated links. The tracks to pins 13 of both IC3 and IC8 should be cut; pin 13 of IC3 is now linked to pin 11 of IC2 and pin 13 of IC8 to pin 17 of IC7. Pin 7 of IC5 should be cut from its track and linked to pin 14 of IC6. The forward/reverse input line, bit 8, now becomes the most significant bit of the pulse width control.

The pulse width modulator now works in exactly the same way as before but with an output frequency of 10 kHz, exactly half the original value.

Once the board has been tested it can be mounted using brackets or adhesive PCB slots directly above the motor driver board in the diecast box. The two supply pins are wired to the existing 12 V input terminals, while the FWD/REV and PWM outputs are connected to the corresponding inputs of each motor driver channel. We used a 15-way D-type Cannon socket bolted onto the case side for the eight bit data input. The two strobe lines and ground wire should also be wired to this connector.

# THREE MINI-MICRO PROJECTS

**Confused by the multitudes of microcomputers on the market, or just dissatisfied by what they offer? Readers who are often brew their own, so here's a collection of circuits to help enhance your system. Designs by Martin Postranecky.**

## Computer System Reset Generator

In custom (ie home) built computer systems and peripherals it is advantageous to generate a SYSTEM RESET pulse which does not lose or corrupt data already stored or in 'transit'. This circuit accepts a variety of reset inputs (from simple push-to-earth switches, any number of which can be connected to point A using the 1N4148 diodes) and generates its own 'power-on' reset pulse.

When any reset pulse is received, via the D1, D2 . . Dn 'OR' gate, both halves of the IC3 timer are triggered. If the SYSTEM BUSY is *not* asserted (ie the 2Clear input is held 'low') the $\overline{2Q}$ output is 'high' and the reset output 1Q passes via IC2b to the SYSTEM RESET output.

If, on the other hand, the SYSTEM BUSY is asserted when a reset input is received, the 1Q output will be blocked by the $\overline{2Q}$ line until either the busy signal ends (to prevent any data in I/O transit being lost) — or, if the busy line remains 'high' due to a fault, until timeout pulse $\overline{2Q}$ ends. Set the timeout components (C2, R5) so that the interval exceeds safely any expected busy interval. Set the RESET interval (C3, R4) to give the required length of pulse *longer* than the timeout period.

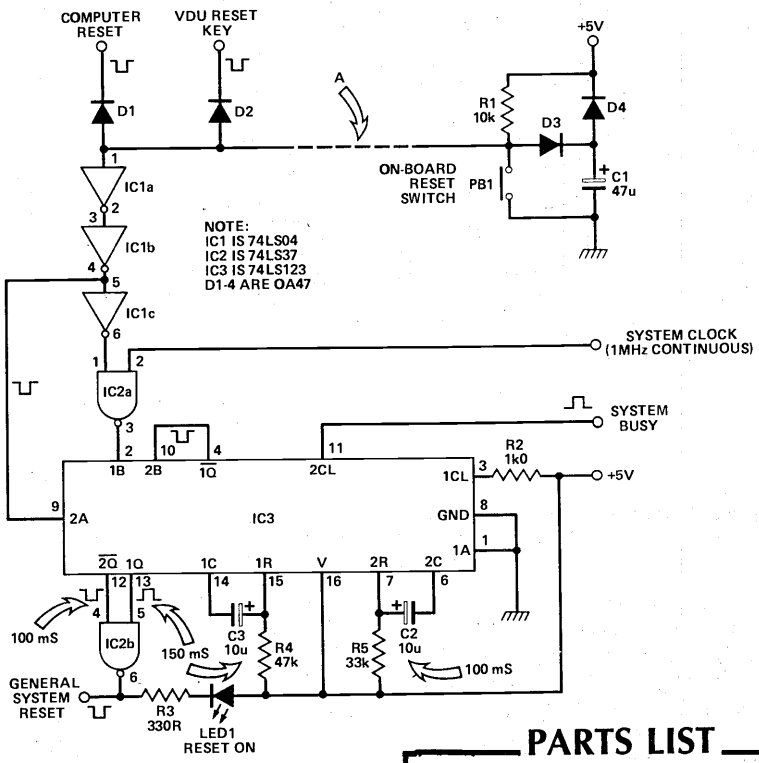



**Fig. 1 Circuit diagram for the computer reset generator.**

The CLOCK line (any continuous system clock will do) and IC2a provide a permanent RESET if any reset swtich is held down LED1 gives a visual indication of the RESET function.

**PARTS LIST**

| | |
|---|---|
| **Resistors (all ¼W, 5%)** | |
| R1 | 10k |
| R2 | 1k0 |
| R3 | 330R |
| R4 | 47k |
| R5 | 33k |
| **Capacitors** | |
| C1 | 47u 16 V Electrolytic |
| C2,3 | 10u 16 V Electrolytic |
| **Semiconductors** | |
| IC1 | 74LS04 |
| IC2 | 74LS37 |
| IC3 | 74LS123 |
| D1-4 | OA47 |
| LED1 | any red LED |

## Time-out Generator And System Failure Alarm

This circuit provides an audible warning of a master clock and/or derived clock failure. It also generates its own time-out reset pulse (which can be applied to the previous circuit at the point A). The two clock lines tested (I suggest the continuous MASTER CLOCK and some derived clock pulse train far down the circuit chain — eg the RAM clock) are biased by R1, R2 and R3, R4 so that pulses via C1 and C2 continously re-trigger the second timer in IC4. If any interruption occurs in either clock line longer than that set by C4, R6 (preset this safely longer than any normally expected period between the clock pulse trains), the $\overline{2Q}$ line will return high causing IC5a to change its outputs and trigger the alarm.

The SYSTEM BUSY line going low will trigger the other half of the timer. If after a preset interval (C3, R5) the busy line is still asserted low the output $\overline{1Q}$ of IC4 will clock IC5a and change the 2Q and $\overline{2Q}$ outputs. In addition to triggering the alarm a TIME-OUT RESET is generated via IC1b. (This could be applied to the RESET GENERATOR circuit making a virtually self-correcting system). IC1b prevents the self-preset system operating in the case of the clock failure as this will usually indicate a more serious hardware fault, ie a short).

The alarm used is a two tone 555-based sound generator requiring 15 V for adequate volume. This has been used to distinguish between the clock and/or timeout failure (high tone) and the high temperature warning described in the fourth circuit (low tone). The output from IC2d enables the 555, while IC2e controls the tone by switching Q1.

### PARTS LIST

| | |
|---|---|
| **Resistors (all ¼W, 5%)** | |
| R1,2 | 1k5 |
| R3-6,9,10 | 1k0 |
| R7,14 | 33k |
| R8,15,16 | 10k |
| R11,13 | 15k |
| R12 | 56k |
| **Capacitors** | |
| C1,2 | 1n0 ceramic |
| C3 | 47u 16 V electrolytic |
| C4 | 22u 16 V electrolytic |
| C5 | 33n ceramic |
| **Semiconductors** | |
| IC1 | 74LS32 |
| IC2 | 74LS06 |
| IC3 | 74LS08 |
| IC4 | 74LS123 |
| IC5 | 74LS74 |
| IC6 | 555 |
| **Miscellaneous** | |
| TX1 | PB-2720 |

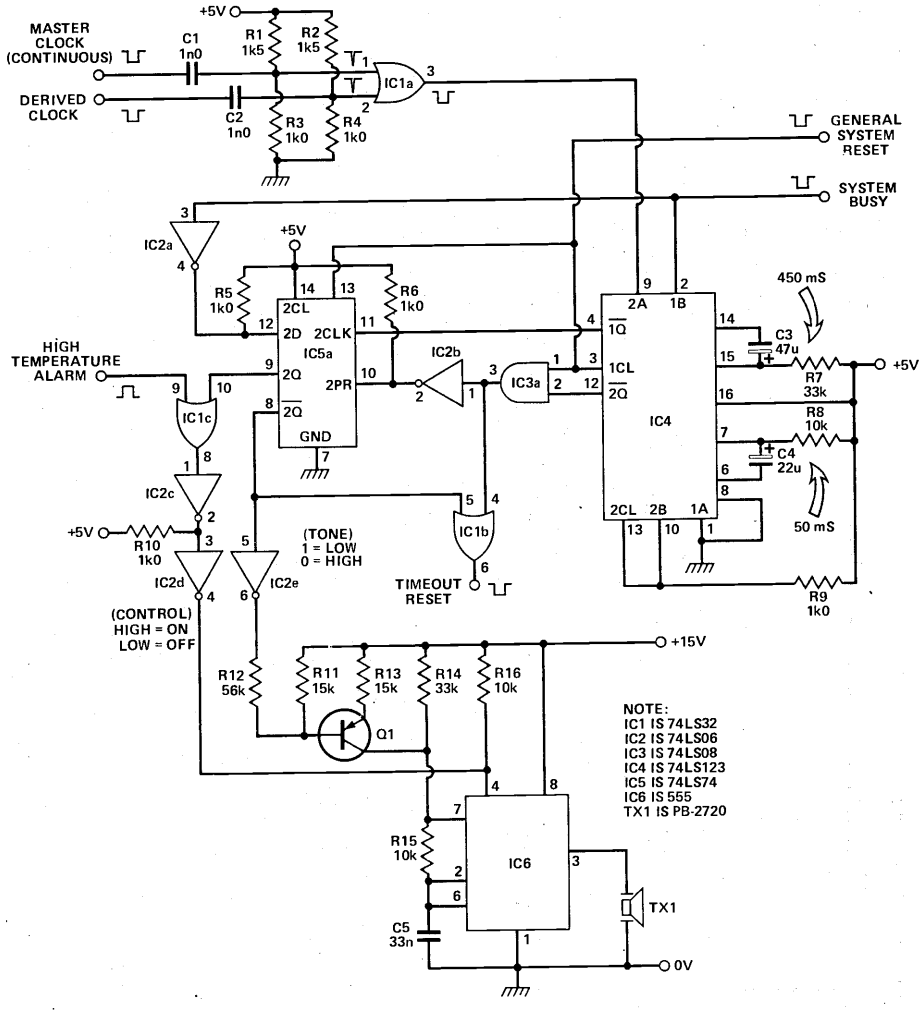



**Fig. 2 The circuitry for the time-out generator and system failure alarm.**

## Supply Voltage Check Scan and DVM Display

This simple scanning circuit is used to check the status of the multitude of DC supply lines in a typical computer/peripherals system. A continous clock input is divided by IC1 (more 7493s can be used if only fast clocks are available) and decoded by IC2.)

The various supply voltages are divided down by the resistor networks R2-R9 and R10-R17 to provide safe levels for the analogue multiplexer IC3, a CMOS 4051B, which passes the selected supply line voltage to the standard DVM circuit. In this case the Intersil ICL 7107 single chip voltmeter is used together with four 7-segment LEDs to provide an accurate visual check on the DC supply lines.

The last input is used to display (in °C) the temperature reading taken by the sensor described in the next section.

### PARTS LIST

**Resistors (all ¼W, 5%)**

| | |
|---|---|
| R1-9 | 1k0 |
| R10-17 | 5k6 |
| R18 | 270R |
| R19 | 1k5 |
| R20 | 470k |
| R21 | 180k |
| R22 | 100k |

**Potentiometers**

| | |
|---|---|
| PR1 | 220k miniature horizontal preset |
| PR2 | 2k2 miniature horizontal preset |
| PR3 | 470k miniature horizontal preset |

**Capacitors**

| | |
|---|---|
| C1,9,11 | 47u 16 V electrolytic |
| C2,4,8,10 | 100n polycarbonate |
| C3 | 100p ceramic |
| C5 | 10n ceramic |
| C6 | 47n ceramic |
| C7 | 220n polycarbonate |

**Semiconductors**

| | |
|---|---|
| IC1,2 | 74LS90 |
| IC3 | 4051B |
| IC4 | 7107 |
| DISP1-4 | common anode seven-segment displays, one half and three full |

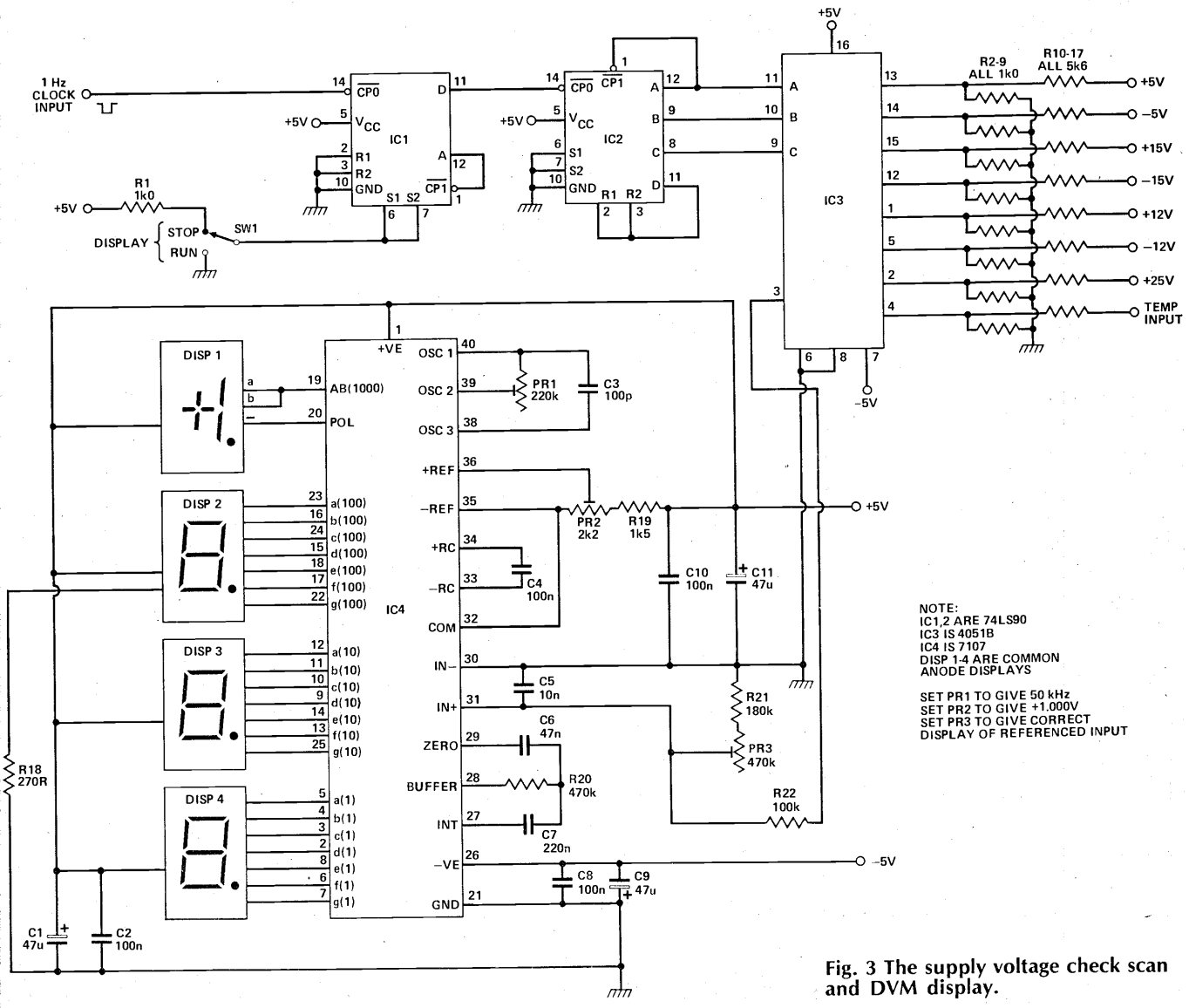



**Fig. 3 The supply voltage check scan and DVM display.**

# REAL TIME CLOCK/CALENDAR

**It seems strange that many microcomputers cannot tell the time of day or the date when such a facility can be so useful to the programmer. Never fear, ETI is here, with a simple peripheral for 6502-based machines. Design by M.D. Bedford**

Programmers who are familiar with mainframe or minicomputers will probably be aware that it is generally possible to access the actual time and date from within a program. Such a facility is known as a real time clock and is often not available on the more modest microcomputers. It is not difficult to see that a real time clock would enhance any system — applications range from control programs, to the determination of the elapsed time between occurrences, to giving listings that professional touch by using the time and date in the header.

Two approaches are possible for the implementation of a real time clock — software or hardware. Traditionally, a software solution has been used in which a hardware interrupt is generated at regular intervals, probably every 20 milliseconds, these being counted by the interrupt handling routine which then calculates the time and date. Such a system obviously requires initialising and would prompt the user for the time and date each time the computer was switched on *(our own word processor uses this system — Ed)*. Quite apart from the possible inconvenience, this method is probably unsuitable for most microcomputer users as it would require modification of the monitor program in ROM to prompt for the time and date. On the other hand, it is possible to devise a hardware alternative with battery back-up which is transparent to the system when not being accessed and doesn't lose the time and date on power-down of the main system.

For these reasons a hardware approach is presented here. The design is primarily intended for the Tangerine Microtan system, the PCB given here being of such a size that it will plug directly into the system rack. From an electronic point of view, however, there is no reason why the board may not be used with any 6502-based computer.

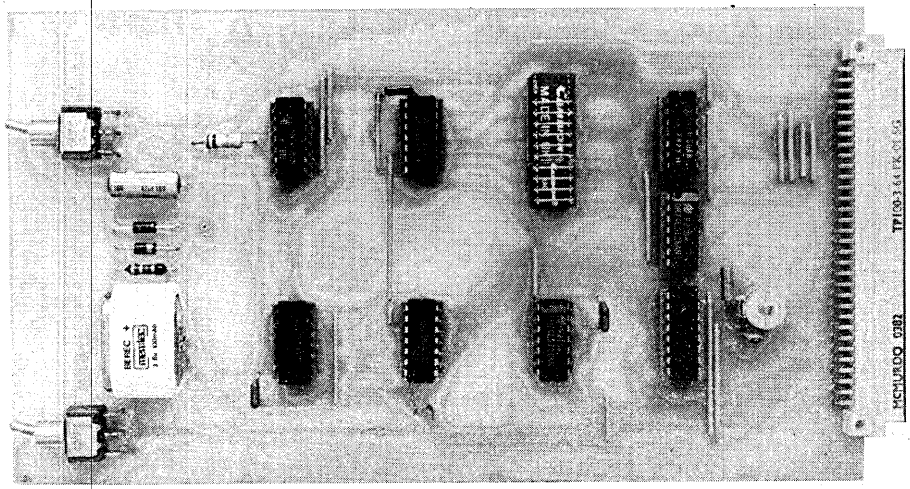
## Functional Description

The real time clock, which may be configured to occupy any 16-byte block within the Tangerine I/O area, has 16 registers as specified in Table 1. It will be noticed that 12 of the registers are used to store the time and date, two registers being used to store (in BCD format) any number which may take a value greater than nine. For example, the value of the minutes is calculated as (10*REGISTER 5) + REGISTER 4. Of these 12 registers, registers 1-3 are read-only, these 'seconds' registers being automatically set to zero on starting the clock.

Each time the clock is updated, ie every tenth of a second, a flip-flop is set, writing a value of 15 to all the readable registers to indicate that an update has taken place since the last read. Reading a register under these conditions resets the flip-flop so that a further read will produce a valid result.

This board may also be used to generate interrupts at regular intervals, this function being controlled by register 15 as described in Table 2. Switch SW2 may be used to disable interrupts, a facility which is especially useful in view of the fact that this board does not reset at switch-on.

The remaining registers are write-only and have various control functions. Register 0 should have a value of 0 written to it to select non-test mode for normal operation. A value of 1, 2, 4 or 8 should be written into register 13 to indicate leap year, leap year + 1, leap year + 2, or leap year + 3 respectively. A value of 1 written to register 14 will start the clock, whereas a value of 0 will stop it. Switch SW1 gives the board write-proteciton, hence obviating the accidental overwriting of the time and date once initialised. This facility does not affect register 15 so that interrupts may still be selected when the



**A bird's eye view of the completed project.**

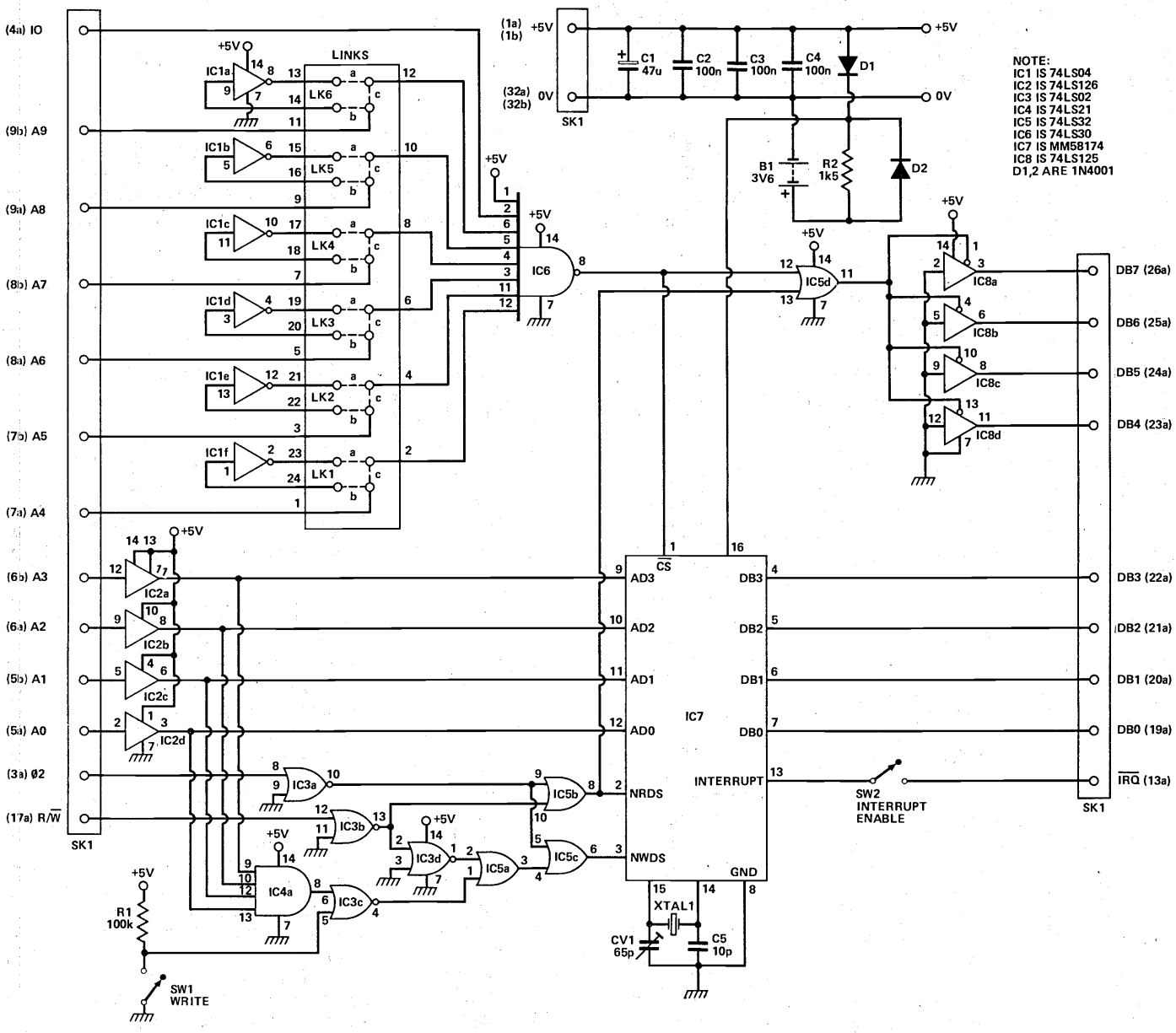



**Fig. 1 Circuit diagram of the real time clock/calendar. Non-Microtan owners will find a circuit to generate the IO signal in last month's ETI.**

board is write-protected. Both switches are mounted so that 'down' selects the enabling of the appropriate function.

The battery back-up facility allows data to be retained when the computer is not switched on, hence avoiding the need to initialise the clock at power-on. The time and date will be retained for about three months with a fully charged battery and a minimum of one hours use every nine days will ensure that the battery remains in a state of full charge.

## Construction

If the printed circuit board layout presented here is adhered to, construction should present no difficulties. Since the board is of a single-sided design, a number of wire links need to be fitted as shown on the component layout diagram. Sockets should be used throughout for the integrated circuits. It should be noted that the MM58174 IC is fabricated in CMOS and accordingly the usual precautions of not touching the pins of the IC and not soldering the board while the IC is in its socket should be adhered to.

We suggest that DIL headers plugged into DIL sockets should be used for the wiring of the selectable address links. A 16-pin and an 8-pin socket should be used to make up the 24-pin by 0.3'' socket used for these links. The required start address should be set up as follows: the start of the board is 16* (the binary number represented by links 1-6) from the start of the Tangerine I/O area, where link 1 is the least significant bit. Making links a and b gives a 0, making link c gives a 1. So, for example, the following links will set up the board to start at 48 bytes from the start of the I/O area: link 6 ab, link 5 ab, link 4 ab, link 3 ab, link 2 c, link 1 c. If the board is to be constructed to a different layout to suit non-Tangerine systems, the only points to be borne in mind are that C2, C3 and C4 should be well distributed around the board and that XTAL1, CV1 and C5 should be mounted close to IC7.

## HOW IT WORKS

The heart of the circuit, IC7, is the MM58174 real time clock which reads and writes four bits of data onto DB0-DB3. ALthough not absolutely necessary (since the top four bits could be masked out by programming), a neat hardware solution is provided by the use of IC8 to zero DB4-DB7 during read operations. The circuitry comprising IC1, IC6 and the DIL links gives a chip select for IC7 and IC8 when an address in the range selected by the links is accessed.

Since the MM58174 is specifically intended to interface with microprocessors such as the 8080 or Z80, the circuitry comprising IC3 and most of IC5 is required to generate the NRDS and NWDS signals from the 6502 R/$\overline{W}$ and $\phi 2$. Hence write protection may be provided by blocking NWDS when SW1 is in the closed position. IC4 is used to detect when register 15 is being addressed (A0-A3 all high) and under these circumstances overrides the write protection.

IC2 is to buffer A0-A3 — in fact, the whole circuit is designed to present no more than one TTL load to any bussed signal.

D1 is used to pass the + 5 V supply to IC7 when it is present, the battery being trickle-charged through R2 under these conditions. When the + 5 V supply is not present, D1 prevents the battery from discharging through the power supply and IC7 is supplied with sufficient voltage to operate in standby mode via D2.

## PARTS LIST

| | |
|---|---|
| Resistors (all ¼W, 5%) | |
| R1 | 100k |
| R2 | 1k5 |
| Capacitors | |
| C1 | 47u 16 V axial electrolytic |
| C2-4 | 100n disc ceramic |
| C5 | 10p ceramic plate |
| CV1 | 5-65p trimmer |
| Semiconductors | |
| IC1 | 74LS04 |
| IC2 | 74LS126 |
| IC3 | 74LS02 |
| IC4 | 74LS21 |
| IC5 | 74LS32 |
| IC6 | 74LS30 |
| IC7 | MM58174 (see Buylines) |
| IC8 | 74LS125 |
| D1,2 | 1N4001 |
| Miscellaneous | |
| SW1,2 | SPCO PCB-mounting toggle switches (see Buylines) |
| B1 | 3V6 PCB-mounting Nicad battery (see Buylines) |
| SK1 | 2 x 32 way A + B DIN Euro connector (male, angled pins) — see Buylines |

PCB (see Buylines); DIL sockets to suit.

## TABLE 1

List of Real Time Clock Registers

| Reg No | Function | Access Mode |
|---|---|---|
| 0 | test | write |
| 1 | tenths of seconds | read |
| 2 | units of seconds | read |
| 3 | tens of seconds | read |
| 4 | units of minutes | read/write |
| 5 | tens of minutes | read/write |
| 6 | units of hours | read/write |
| 7 | tens of hours | read/write |
| 8 | units of days | read/write |
| 9 | tens of days | read/write |
| 10 | day of week | read/write |
| 11 | units of months | read/write |
| 12 | tens of months | read/write |
| 13 | year status | write |
| 14 | start/stop | write |
| 15 | interrupt | read/write |

## TABLE 2

DESCRIPTION OF INTERRUPT MODES

| Function | Value in Register 15 |
|---|---|
| no interrupts | 0 or 8 |
| single interrupt after 60 seconds | 4 |
| repeated interrupts at 60 second intervals | 12 |
| single interrupt after 5 seconds | 2 |
| repeated interrupts at 5 second intervals | 10 |
| single interrupt after 0.5 seconds | 1 |
| repeated interrupts at 0.5 second intervals | 9 |

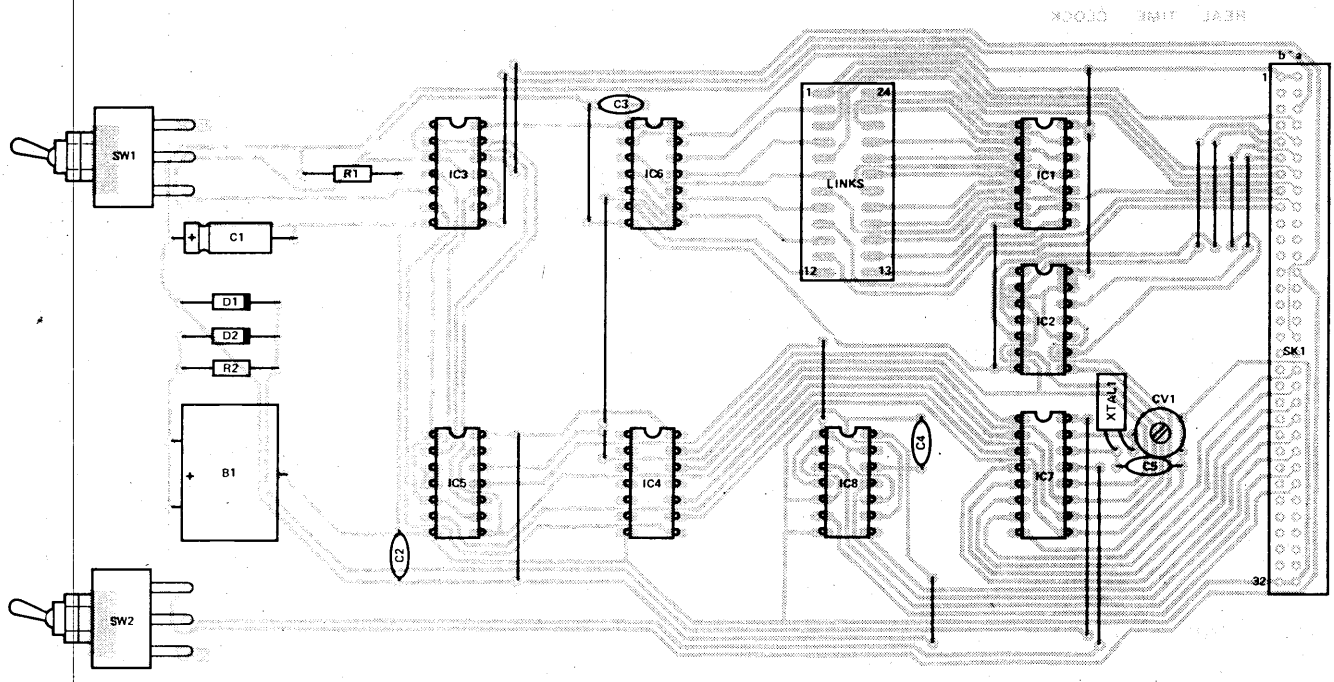
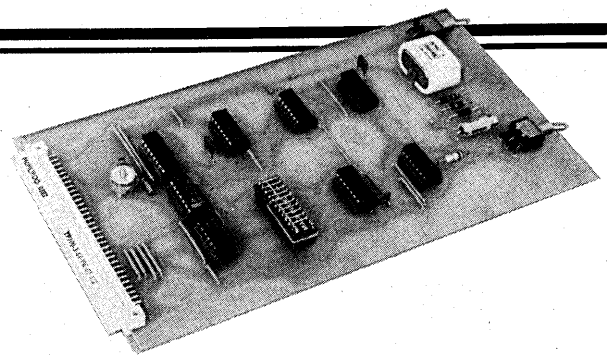



Fig. 2 Component overlay for the real time clock.



## Programming

The following BASIC program is used for initialising the real time clock/calendar. The board should be write-enabled before running the program — however, if this is not done the user will be instructed to do so by the program. The program will fully validate the information given before writing it to the clock, to reduce the likelihood of human errors. We suggest that a time and date a few minutes ahead of the actual time is entered, the RETURN following the day of the week request being pressed exactly as this time arrives.

```
10 REM . . .MM58174 REAL TIME
     CLOCK INITIALISATION
     PROGRAM
20 DEF FNC(I) = VAL(MID$(TD$,
     I, 1))
30 DEF FNN(I) = 10 * FNC(I) +
     FNC(I+1)
40 DIM DM(12)
50 DATA 31, 28, 31, 30, 31, 30,
     31, 31, 30, 31, 30, 31
60 FOR I=1 TO 12: READ
     DM(I):NEXT I
70 PRINT "MM58174
     INITIALISATION"
80 INPUT "ENTER START
     ADDRESS OF BOARD"; AD
90 POKE AD, 0:REM . . . NON
     TEST MODE
100 POKE AD+15, 0:REM . . .
     DISABLE INTERRUPTS
110 POKE AD+14, 0:REM . . .
     STOP CLOCK
120 I=PEEK(AD+4):I=PEEK(AD+4)
130 J=15 AND (I+1):POKE AD+4,
     J
140 I=PEEK(AD+4)
150 IF I=J THEN 180
160 PRINT "WRITE ENABLE REAL
     TIME CLOCK — RETURN
     WHEN DONE"; :GET A$
170 GOTO 120
180 INPUT "ENTER TIME AND
     DATE IN THE FORM HH MM
     DD/MM/YY"; TD$
190 HH = FNN(1)
200 IF HH<0 OR HH>23 THEN
     180
210 POKE AD+7, FNC(1): REM . . .
     HOURS * 10
220 POKE AD+6, FNC(2): REM
     . . .HOURS
230 MM = FNN(4)
240 IF MM<0 OR MM>59 THEN
     180
250 POKE AD+5, FNC(4):REM . . .
     MINUTES * 10
260 POKE AD+4, FNC(5):REM
     . . .MINUTES
270 YY = FNN(13)
280 IF YY<0 OR YY>99 THEN 180
290 YR =
     2↑(3-(YY-4*INT(YY/4)))
300 IF YR=8 THEN
     DM(2)=29:GOTO 320
310 DM(2)=28
320 POKE AD+13, YR:REM . . .
     YEAR STATUS
330 MM = FNN(10)
340 IF MM<1 OR MM>12 THEN
     180
350 POKE AD+12, FNC(10):REM
     . . . MONTH * 10
360 POKE AD+11, FNC(11):REM
     . . . MONTH
370 DD = FNN(7)
380 IF DD<1 OR DD>DM(MM)
     THEN 180
390 POKE AD+9, FNC(7):REM . . .
     DAY * 10
400 POKE AD+8, FNC(8):REM . . .
     DAY
410 INPUT "ENTER DAY OF WEEK
     (1-7, 1=MONDAY)"; DW
420 IF DW<1 OR DW>7 THEN
     410
430 POKE AD+10, DW:REM . . .
     DAY OF WEEK
440 POKE AD+14, 15:REM . . .
     START CLOCK
450 PRINT "WRITE DISABLE REAL
     TIME CLOCK"
460 STOP
470 END
```

To access the time and date from within a program, the following BASIC subroutine may be used: a few words of explanation are probably appropriate. Line 1040 clears the update flip-flop by reading the clock once. The following two lines then loop until a value of 15 is read, indicating that an update has just taken place and that a tenth of a second is available to read the registers before the next update. It is the requirement to read 11 registers in this 100 milliseconds time slot (in order to avoid the possiblity of an update occurring between the reading of two registers) which accounts for the strange appearance of much of the rest of the subroutine. The inherent slowness of BASIC on an eight-bit microcomputer dictated the avoidance of FOR-NEXT loops, subscripted variables and numerical constants in the time-critical portion. The routine returns with numeric values of seconds, minutes . . . months in R2-R7 respectively, an ASCII representation of the time in TM$ and an ASCII version of the date in DT$.

```
1000 REM . . .MM58174 READING
     ROUTINE
1010 R2=AD+2:R3=AD+3:R4=
     AD+4:R5=AD+5
1020 R6=AD+6:R7=AD+7:R8=
     AD+8:R9=AD+9
1030 RA=AD+10:RB=AD+11:
     RC=AD+12
1040 Z = PEEK(AD+2)
1050 Z = PEEK(AD+2)
1060 IF Z <>15 THEN 1050
1070 R2=PEEK(R2):R3=PEEK(R3):
     R4=PEEK(R4)
1080 R5=PEEK(R5):R6=PEEK(R6):
     R7=PEEK(R7)
1090 R8=PEEK(R8):R9=PEEK(R9):
     RA=PEEK(RA)
1100 RB=PEEK(RB):RC=PEEK(RC)
1110 TM$=CHR$(48+R7)+CHR$
     (48+R6)+":"+CHR$(48+R5)
     +CHR$(48+R4)
1120 TM$=TM$+":"+CHR$(48+-
     R3)+CHR$(48+R2)
1130 DT$=CHR$(48+R9)+CHR$
     (48+R8)+" "+MM$(RB+
     10*RC)
1140 R2 = R2 + 10*R3
1150 R3 = R4 + 10*R5
1160 R4 = R6 + 10*R7
1170 R5 = R8 + 10*R9
1180 R6 = RA
1190 R7 = RB + 10*RC
1200 RETURN
```

Prior to calling the above subroutine, the following portion of program should be executed to store the names of the months in the array MM$:

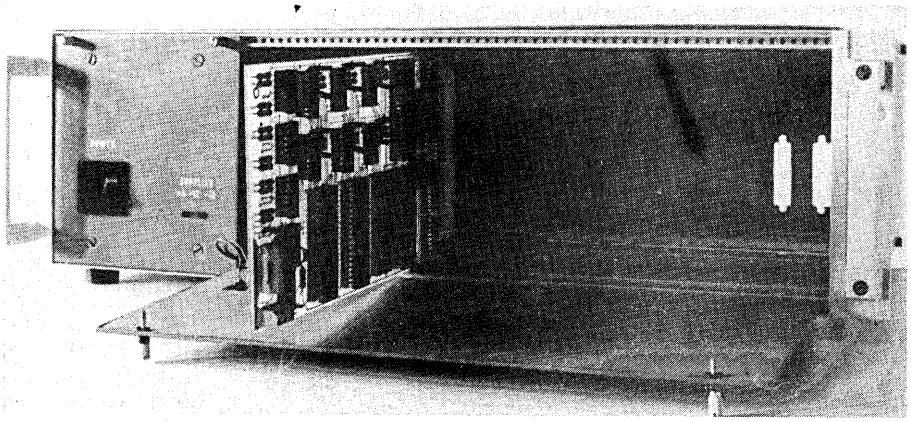
```
10 DIM MM$(12)
20 DATA "JANUARY",
     "FEBRUARY", "MARCH",
     "APRIL", "MAY", "JUNE"
30 DATA "JULY", "AUGUST",
     "SEPTEMBER", "OCTOBER",
     "NOVEMBER", "DECEMBER"
40 FOR N=1 TO 12: READ
     MM$(N):NEXT N
```

## BUYLINES

**The MM58174 real time clock/calendar IC is available from Cricklewood Electronics, Technomatic or Watford Electronics. The PCB-mounting switches and Nicad battery might be a bit tricky to find unless you have industrial contacts, but non-PCB types could be used and wires taken to the PCB pads; there's enough room on the PCB, which is available from our PCB Service as usual. The Euro connector is stocked by Watford Electronics.**

# AUDIO BOARD

**Here's a powerful and versatile peripheral for people requiring a digital-to-analogue or sound-generating capability for their computer. Although the PCB is designed for Tangerine users, the circuit may be readily interfaced to any 6502-based system. Design and development by M. D. Bedford.**



**The completed Sound/DAC board in a Tangerine Microtan rack-mounted system. The board may be used with other 6502-based systems.**

When control applications on a microcomputer are proposed, it soon becomes evident that DACs and ADCs are required to interface to the real world. For more light-hearted uses and for games, the addition of sound effects can do much to enhance a program. Although multiplexed ADCs are available, giving eight or 16 channels and requiring only one eight-bit port plus control lines to support them (see the ZX ADC in the January '83 issue), DACs require eight bits per channel and would quickly use up the available I/O ports on most systems. For this reason, the circuit which is presented here was designed so that it wouldn't use system I/O ports. The board presented here is specifically intended for the Tangerine Microtan system and as such will plug directly into any expansion slot on the system motherboard: DIL switches or links are provided on the board in order to configure it to start at any 16-byte boundary within the 1K I/O area.

For users with other 6502-based systems it should not prove difficult to interface the circuit. The only non-standard signal is the one designated IO which is used in the Microtan system to indicate that an address within the I/O area (ie within the address range BC00 to BFFF) is being accessed. On any other system, address lines A10-A15 should be decoded to generate such a signal and Fig. 3 shows a simple circuit which may be used.

## The Circuit

The DAC0800 is a low-cost high-speed multiplying DAC with an accuracy of ± 1 LSB which, when used in conjunction with an op-amp, is capable of giving a low impedance voltage output. Six of these devices have been used with 747 dual op-amps to give outputs in the range 0-10 V.

The General Instruments AY-3-8910 programmable sound generator IC forms the basis of the sound effect feature of this board, the LM380 being provided in order that a loudspeaker may be driven with no external circuitry. 6520 PIOs are used as a simple and inexpensive means of interfacing the DAC0800 and AY-3-8910, but 6821s may also be used. Buffering of some signals limits the load presented to the bus signals.

The circuitry comprising IC16, IC17 and the bank of DIL switches allows the positioning of the board in any 16-byte block within the I/O area. The start address is calculated as the binary number given by the six switches multiplied by 16, where the switches on the circuit diagram are shown in the '1' position and SW1 is the least significant. For example, if switches SW1-6 are in positions 0,0,0,0,0,1 then the first address on the board will be 16 bytes from the start of the I/O area.

## Construction

The layout of a printed circuit board is presented here and, due to the fairly high packing density it is suggested that, for those intending to incorporate the circuit into a Microtan system, this layout be adhered to. It should be noted that the board is of the double-sided, pinned-through type with the result that every hole not intended for the mounting of components should be fitted with a pin and soldered both on the top and the bottom of the board. Scrutiny of the overlay will reveal that this must be carried out prior to the fitting of any DIL sockets. If, however, it is not intended to fit the circuit board into a card frame along with other Tangerine boards, then a larger board may be used and some form of breadboarding technique employed for construction. If this option is taken care should be

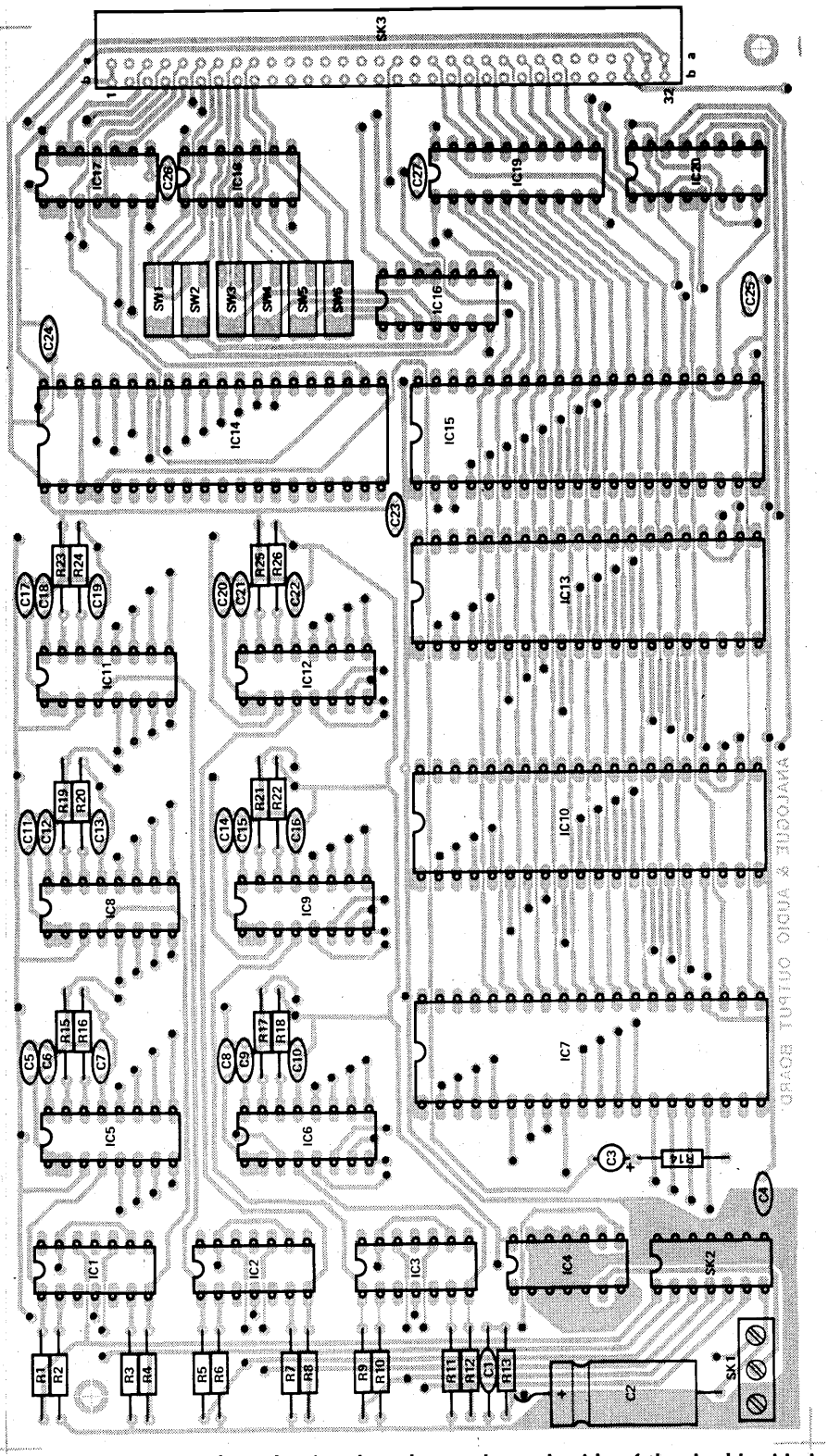



**Fig. 1 Component overlay. The tinted tracks are the underside of the double-sided board: through-board pins soldered on both sides are required wherever a dot appears.**

exercised in the positioning of certain capacitors. C4 and C23-27 should, as far as possible, be well distributed around the board. C7,10,13,16,19 and 22 decouple the —12 V rail, and these components should be positioned so that one capacitor from each of the two sets is close to each of the DAC0800s.

One final point applies irrespective of the method of construction: since DIL switches are relatively expensive and in such an application will most probably be set up once and rarely changed, it is suggested that, as an alternative, DIL headers with appropriate soldered links are used and plugged into DIL sockets.

## PARTS LIST

| | |
|---|---|
| Resistors (all ¼W, 5% except where stated) | |
| R1-12 | 10k 2% |
| R13 | 2R7 |
| R14 | 20k |
| R15-26 | 12k 2% |
| Potentiometer | |
| RV1 | 22k logarithmic |
| Capacitors | |
| C1, 5, 7, 8, 10, 11, 13, 14, 16, 17, 18, 20, 21 | 100n ceramic |
| C2 | 470u 16 V axial electrolytic |
| C3 | 2u2 10 V tantalum |
| C4, 6, 9, 12, 18, 19, 22-27 | 10n ceramic |
| Semiconductors | |
| IC1-3 | LM747 |
| IC4 | LM380 |
| IC5, 6, 8, 9, 11, 12 | DAC0800 or DAC0801 |
| IC7, 10, 13, 15 | 6520, 6820 or 6821 |
| IC14 | AY-3-8910 |
| IC16 | 74LS30 |
| IC17 | 74LS08 |
| IC18 | 74LS04 |
| IC19 | 74LS245 |
| IC20 | 74LS138 |
| Miscellaneous | |
| SK1 | 3-way, 5mm pitch, PCB terminal block |
| SK2 | 14-pin DIL socket |
| SK3 | 2 × 32 way A + B DIN Euro connector (male, angled pins) |
| SW1-6 | hex DIL changeover switch (see text) |
| PCB (see Buylines); DIL sockets to suit | |

## BUYLINES

**No problems with the semiconductors for this project — it's all standard stuff and people like Technomatic, Watford and Cricklewood should have no trouble supplying you. The only difficulty may lie in finding a supplier for the DAC0800 which seems to be a bit elusive: Maplin supply the 0801 which is an acceptable substitute. The Euro socket required for Tangerine rack owners is available from Watford, while the PCB can be obtained from out PCB Service as usual.**

## Programming

While it is beyond the scope of this article to give a detailed functional description of the 6520 and AY-3-8910 ICs, it is expected that the BASIC routines presented here will enable the board to be used without difficulty. In order to make full use of the sound generator, however, it is suggested that an AY-3-8910 data sheet is consulted (the company you buy the chip from should be able to help). In the routines given, the

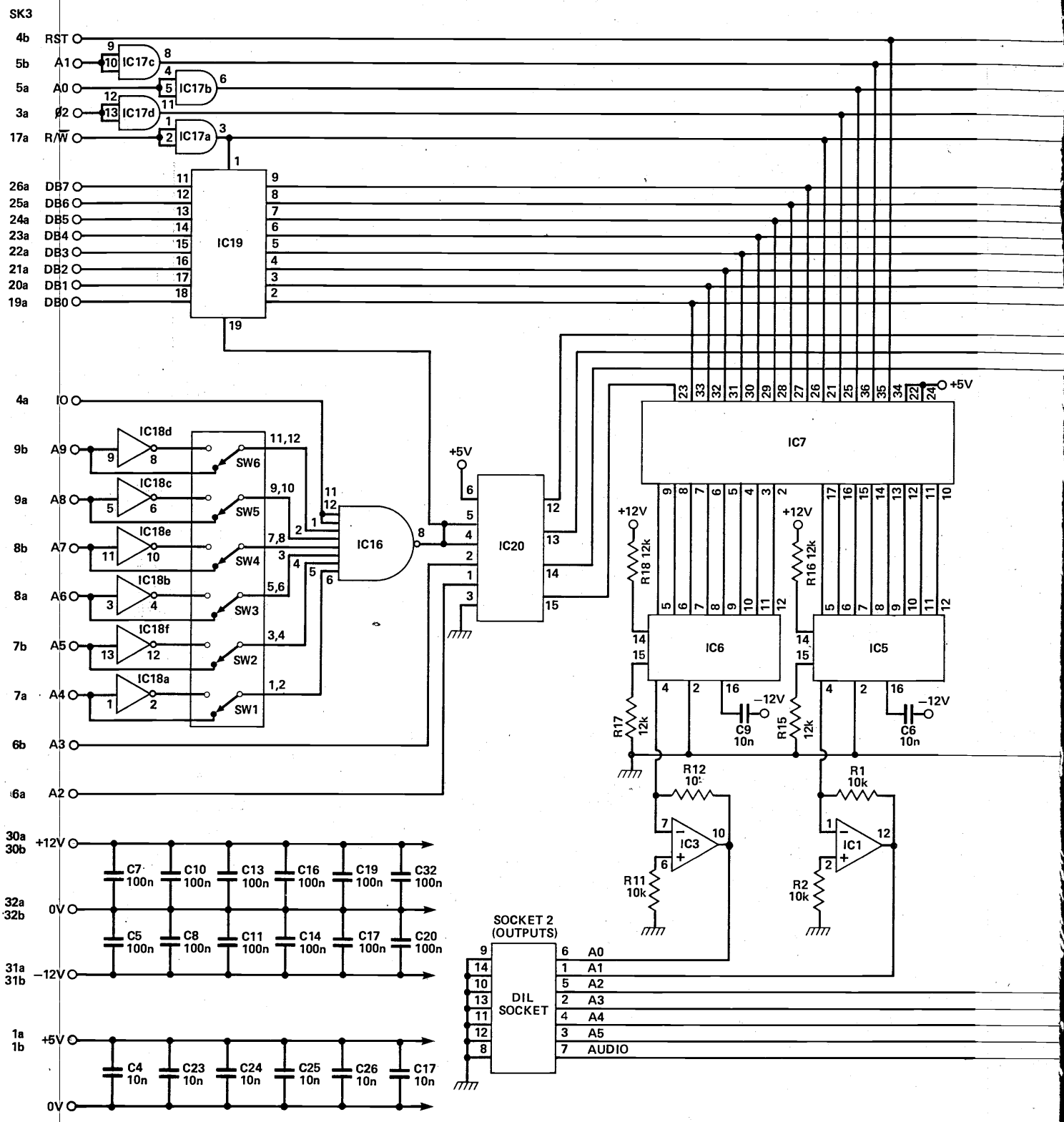



Fig. 2 Circuit diagram of the sound/DAC card.

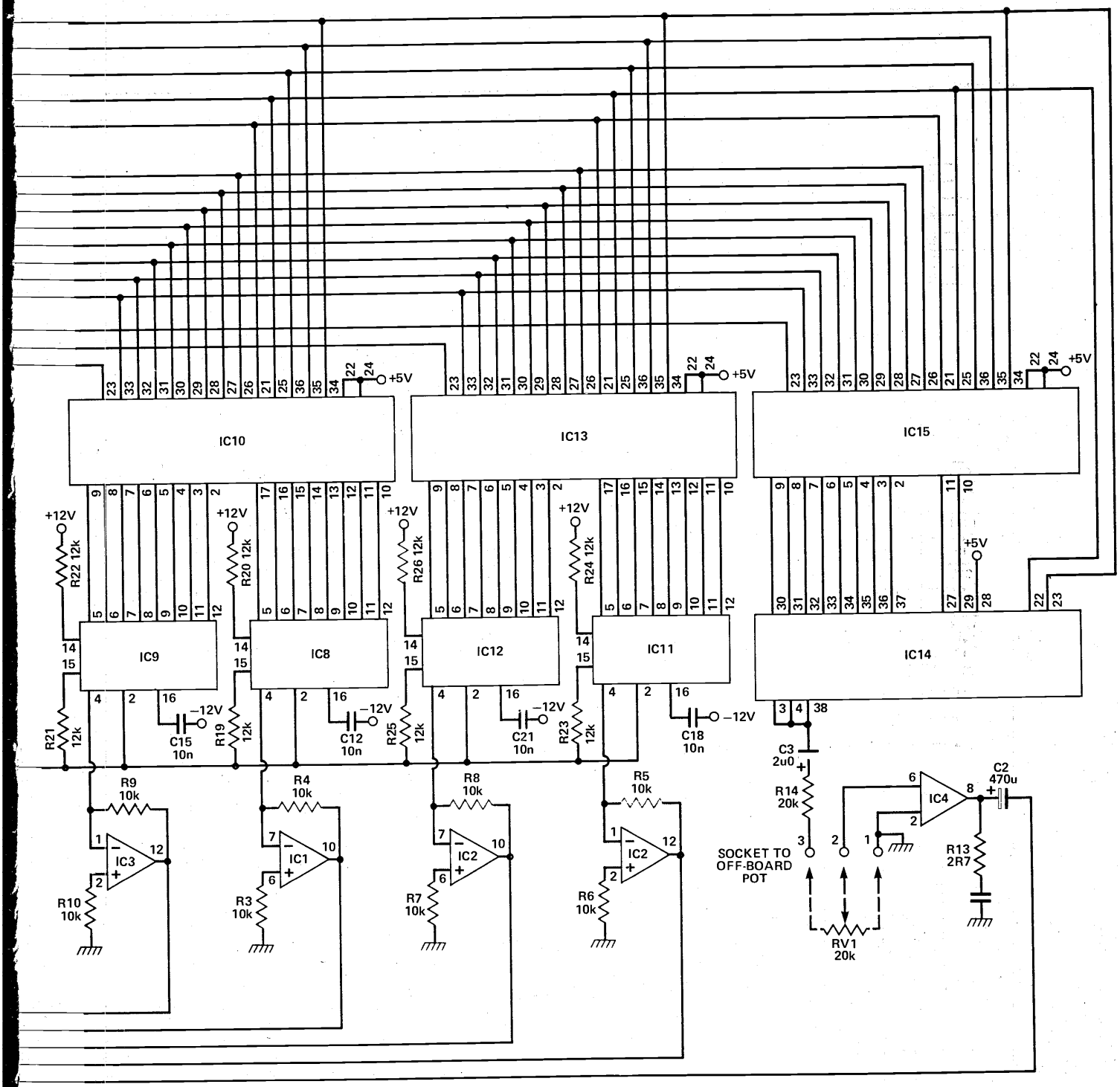
## HOW IT

The circuitry consists of six DACs, ICs 5-12, and one sound generator chip, IC14, which has three separate sound channels and a noise generator.

The circuitry for each of the DAC ICs is identical and we will only discuss IC6. The eight bits of data are held on pins 5-12, while the complementary current outputs are on pins 2 and 4. The DAC is configured for positive low impedance output operation, with $\overline{I_{OUT}}$ on pin 2 connected to ground and the other output fed into the buffer op-amp IC3a.

The data input to IC6 is latched by port A of the peripheral interface adapter IC7 (port B of IC7 latches the data for IC5). Similarly ICs 10 and 13 latch the data for ICs 8,9 and 11,12 respectively.

The fourth PIA, IC15, utilises both ports to control the sound generator chip, IC14. The three audio outputs of




## WORKS

this chip are fed via C3, R14 and the volume control RV1 into IC audio power amp IC4. C1 and R13 form the Zobel network and C2 the DC-blocking capacitor for the amp output. The audio signal and analogue voltages are fed off-board via SK2, a DIN socket with a header plug.

Address decoding for the board is performed by ICs 16, 18 and 20. SW1-6 select which 16-byte block of memory the board occupies. For people using systems other than the Microtan, which generates the IO signal required, the additional decoding circuit for lines A10-15 shown in Fig. 3 will be required. The data bus is buffered by IC19, with the R/W signal on pin 1 selecting the direction and the output of IC16 enabling the tri-state buffers via pin 19. Bus signals which are required by several chips in the circuit are buffered by IC17.

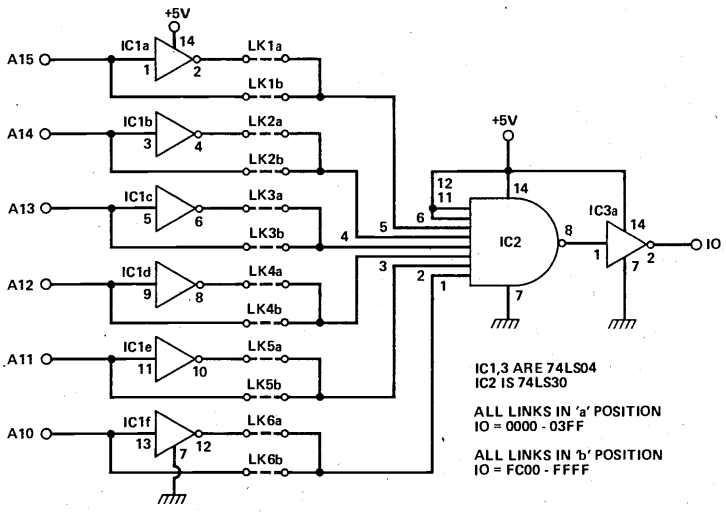



**Fig. 3 Non-Microtan users who wish to build this project will require this circuit to produce the IO signal.**

| | IC1 | IC2 | IC3 | IC4 | IC5 | IC6 | IC7 | IC8 | IC9 | IC10 | IC11 | IC12 | IC13 | IC14 | IC15 | IC16 | IC17 | IC18 | IC19 | IC20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0V | | | | 3,4,5 7,10 11,12 | | | 1 | | | 1 | | | 1 | 1 | 1 | 7 | 7 | 7 | 10 | 8 |
| +5V | | | | | | | 20 | | | 20 | | | 20 | 40 | 20 | 14 | 14 | 14 | 20 | 16 |
| +12V | 9,13 | 9,13 | 9,13 | 14 | 13 | 13 | | 13 | 13 | | 13 | 13 | | | | | | | | |
| −12V | 4 | 4 | 4 | | 3 | 3 | | 3 | 3 | | 3 | 3 | | | | | | | | |

**Table 1 This is a list of power supply connections to the various ICs for people doing their own board layout.**

variable BA should be set to the base address of the board.

**DAC Handling Routine.** In this routine, which should be executed once at the start of the program, N should be set to the number of DAC channels to be initialised. After execution of the routine, the statement:

POKE X, BA+2*(N−1)

will write the value X to the Nth DAC channel.

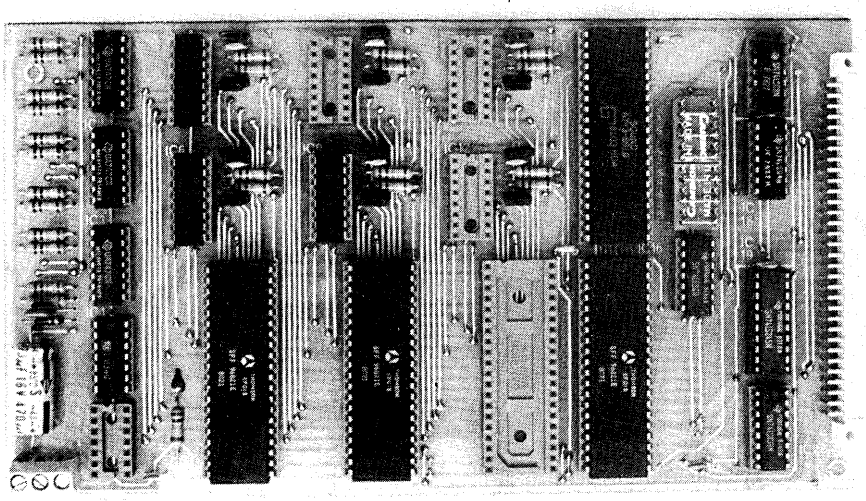
```
10 REM ....DAC INITIALISATION
20 FOR AD = BA TO BA + 2*
(N−1) STEP2
30 POKE AD+1,0: REM....ALLOW
ACCESS TO DDR
40 POKE AD,255: REM....SET DDR
TO OUTPUTS
POKE AD+1,4: REM....ALLOW
ACCESS TO OUTPUT REGISTER
60 NEXT AD
```

**Sound Effect Routines.** The initialisation routine should be executed once at the start of the program: after this subroutines 1000 and 2000 may be called for writes and reads respectively to the AY-3-8910 registers. In these two routines, REG should be set to the register number before entry: for a write, DAT should be set to the value of the data to be written and when reading, DAT will contain the data read after return from the subroutine.

```
10 REM....AY-3-8910
INITIALISATION
20 POKE BA+15,0: REM....6520
PORT B TO WRITE
30 POKE BA+14,255
40 POKE BA+15,4

1000 REM....AY-3-8910 WRITE
ROUTINE
1010 GOSUB 3000: REM....LATCH
ADDRESS
1020 POKE BA+12,DAT:
REM....WRITE DATA
1030 POKE BA+14,2
1040 POKE BA+14,0
1050 RETURN

2000 REM....AY-3-8910 READ
ROUTINE
2010 GOSUB 3000: REM....LATCH
ADDRESS
2020 POKE BA+13,0: REM....6520
PORT A TO READ
2030 POKE BA+12,0
2040 POKE BA+13,4
2050 POKE BA+14,1: REM....READ
DATA
2060 DAT=PEEK(BA+12)
2070 POKE BA+14,0
2080 RETURN

3000 REM .. LATCH ADDRESS
ROUTINE
3010 POKE BA+13,0: REM....6520
PORT A TO WRITE
3020 POKE BA+12,255
3030 POKE BA+13,4
3040 POKE BA+12,REG:
REM....LATCH ADDRESS
3050 POKE BA+14,3
3060 POKE BA+14,0
3070 RETURN
```

Where it is only intended to write to the AY-3-8910 registers, a saving in execution time and program size may be made by incorporating lines 3010-3030 of the latch address routine into the initialisation routine and using the following routine for writing. In this case subroutines 2000 and 3000 are not required.

```
1000 REM....AY-3-8910 WRITE
ROUTINE FOR WRITE-ONLY
APPLICATIONS
1010 POKE BA+12,REG:
REM....LATCH ADDRESS
1020 POKE BA+14,3
1030 POKE BA+14,0
1040 POKE BA+12,DAT:
REM....WRITE DATA
1050 POKE BA+14,2
1060 POKE BA+14,0
1070 RETURN
```



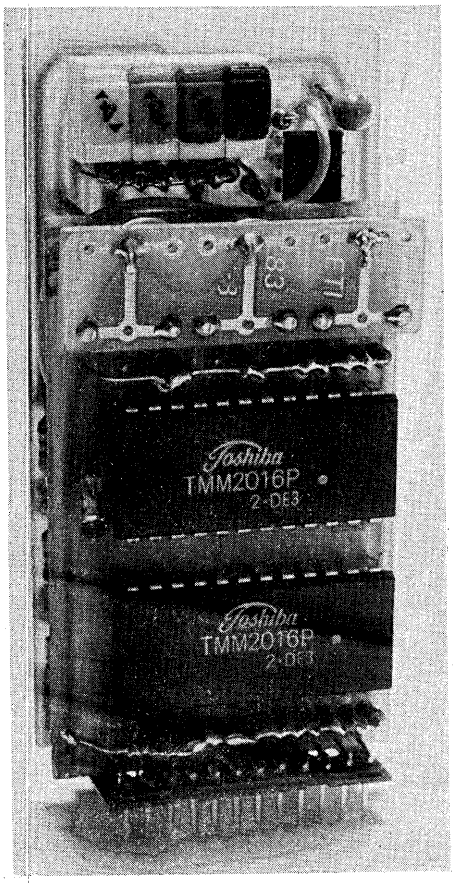
**A completed board, sporting the sound chip and three of the possible six DAC channels.**

# PSEUDOROM

## EPROMs never forget but they're a damned nuisance to reprogram. RAMs are easy to overwrite but they lose their contents on power-down. ETI, naturally, has combined the best of both worlds. Design by Phil Walker.

The ETI PseudoROM now offers the home constructor a device having the capacity of the larger ROM and EPROM chips, but with the programmability of RAM, which can be inserted into the existing ROM socket of your microcomputer. This makes it possible to develop large and complex operating software in one module which only occupies the same physical board area as the eventual ROM or EPROM (if used).

Another advantage is that the access time of the module is only 50 nS or so slower than the memories it contains (if you're using both the 74HC32 and the 74HC138). This will be somewhat less than the normal 450 nS spec of many EPROM devices advertised, and could enable you to run your system at a higher clock frequency (processor permitting), so increasing throughput.




Here's a fine, upstanding project . . .

### The Circuit

The circuit used in this project is essentially very straightforward and consists of four 2K by 8 bit CMOS RAM chips, connected to an address decoder to make an effective 8K by 8 bit memory. By means of a small battery and some extra circuitry this sizeable chunk of memory will retain its data even when unplugged from its socket. To make it even more useful the unit is constructed such that it can be used as four areas of 2K by 8 read/write memory or alternatively four sections of 2K, two sections of 4K or a single section of 8K by 8 read-only memory. All these modes are selectable by the integral switches, except that 8K by 8 is only available when inserted into a 28 pin socket.

The point of this versatility is that the unit can be programmed in an existing RAM socket (or one can be provided) at any time and then transferred to an EPROM or ROM socket without losing data (as long as the write protect switch is used). This means that quite large sections of operating systems or special software can be modified and tested without the delays of erasing and reprogramming EPROMs or losing data in system crashes.

### 2-4-8K

The only difference between the 4K and 8K versions of this unit lie in the type of DIL plug used, the size of socket into which it is plugged and the numbering of the input/output pins on the circuit diagram. For memory simulation up to 4K only 24 pins are needed, while for 8K all 28 pins are required. The easiest way of inserting the unit correctly is to identify the 0 V pin and make sure it goes into the 0 V pin position in the socket used. For 24 pin sockets this is pin 12, for 28 pin sockets it is pin 14. Note that for 28 pin plugs and sockets the equivalent functions will occur on pin numbers having a value of 2 greater than those of 24 pin sockets, with the exception of +5 V on pin 28 and A12 on pin 2 (this latter address line goes to the connection marked (2*) on the circuit diagram).

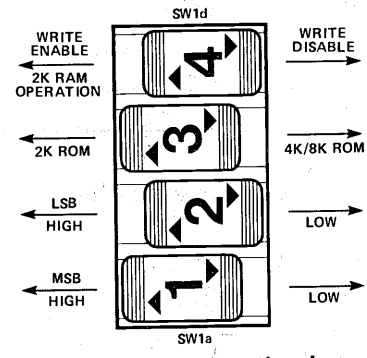
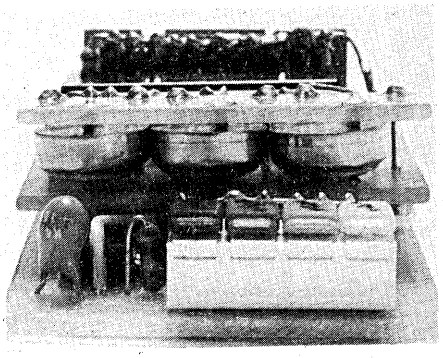



**Fig. 1 Switch operations for the PseudoROM.**



View from the top.

### BUYLINES

**The 74HC series logic ICs which are required for this project are fairly new to the hobbyist scene and appear so far only to be stocked by Ambit International, 200 North Service Road, Brentwood, Essex CM14 4SG. The DIL switch shouldn't cause any problems but if you do have difficulty Maplin stock a suitable type. The mercury cells should be obtainable from most high street chemists and photographic shops, while the PCBs are available using our PCB Service order form.**

## HOW IT WORKS

**The essential part of the circuit consists of IC1-4. These are 6116 CMOS 2K x 8 RAMs, which have the property of consuming very little power when not in use. However, for this to happen pin 18 ($\overline{CS}$) of each device must be at a logic high level and preferably within a few per cent of the supply voltage. Also, the remaining device pins should be held close to either supply rail. R1-19 hold the address and data pins of the RAMs close to 0 V (via R29) when the unit is unplugged, or to the positive supply rail when in position.**

**ZD1, R28,30,31 and Q1 sense the presence of a suitable external power supply. When this is found the inputs to IC5a go low which causes its output to go low also. This then enables IC5b, IC5c and IC6 to operate. IC5c acts as a buffer for the $\overline{OE}$ control (pin 20), while IC5b buffers the $\overline{WE}$ function (normally pin 21). IC6 is used both to buffer the $\overline{CS}$ input and to use the two most significant address lines to select which RAM device is to be activated. SW1a and SW1b with their associated resistors provide default addresses to IC6 when direct inputs are not available. SW1d selects either ROM or RAM type operation whilse SW1c selects 2K or 4K/8K ROM simulation.**

**D1 isolates the memory power supply from that of the host machine when the main power is off and is a germanium device for minimum voltage drop. Likewise, D2 isolates the backup battery from the main supply when not in backup mode. When the external supply fails, the control signals to the RAM ICs are all forced to logic HIGH and the battery will be able to maintain the data in the RAMs for a long time.**

## Construction

Examine the overlay diagram and photographs *very* carefully . . . The unit consists of three PCBs mounted one on top of another. These must be assembled correctly and put together in the right order as there may be no second chance. We suggest that you follow the procedure here so that it goes together correctly, but read it carefully before starting.

Start with the smallest PCB: this is the battery connector. Take three thin brass shims about 0.1" (2.5 mm) wide by 0.6" (15 mm) long and solder each to the central bar of each pattern on the PCB such that they overlap one edge by 0.4" (10 mm) or so. Bend the overlapping length round the edge of the PCB so that it lies near the non-track side. This will form the positive contact for each cell. Alternatively, any springy material to hand may be soldered into the centre hole to do the job. In fact we used contacts from a piece of edge connector.

Next solder six lengths of 20 swg copper (or paper clip) wire into the holes nearest the free end of the battery connector described above. These wires will connect to the next board down and complete the battery holder.

The middle-sized PCB should be assembled next: this will form the middle of the sandwich. Be very careful to get the right value resistors in the right holes. There are nineteen 47k resistors and one 1k0 resistor on the board, all of which mount vertically. Insert them into their holes and solder into position as close as possible to the board. DO NOT CUT ANY OF THE LEADS OFF.

Using two $2\frac{1}{2}$" (60 mm) lengths of 22 swg wire, connect the free ends of each line of resistors together and pass the remainder of the wire through the hole shown on the overlay; solder into position but

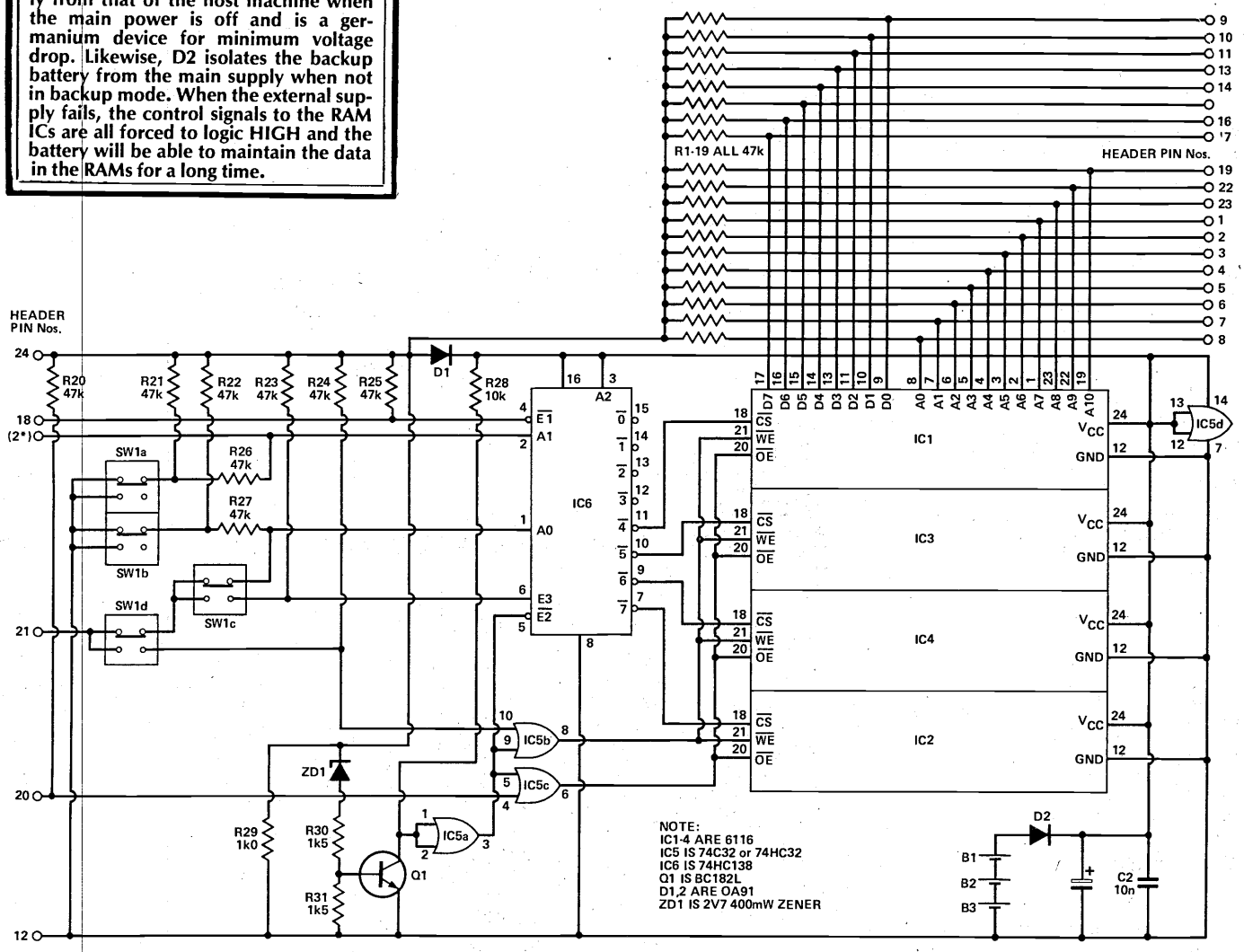



**Fig. 2 Complete circuit diagram for the PseudoROM. All the header pin numbers refer to the 24-pin version: add two to all the numbers for the 28-pin version except for the pin marked (2*). This is left unconnected on the 24-pin version.**

DO NOT CUT OFF. There are seven more links to be made to the lower board which should consist of 1" (25 mm) lengths of 22 swg wire. A useful tip — squeeze the end of the wire with pliers to help it stay in the hole when soldering. Now mount D2 on the PCB making sure that the polarity is correct.

The next step is to assemble the battery connector onto this board. First insert some 22 swg tinned copper wire into the three holes in the middle PCB which lie in the centre of the cell positions. Solder them into position and crop off close to the track side and to about ¼" (6 mm) on the component side. Bend this short wire towards the edge of the PCB and flatten it (. . . GENTLY . . .). These form the negative battery contacts. Now take the small PCB and insert the six wires from it into the corresponding holes in the medium-sized PCB. Put three of the specified mercury cells in position and adjust the two PCBs until the cells are held reasonably firmly in their correct positions. Solder the link wires into position and crop them close to the PCB; then remove the cells.

Solder 12 ½" (12 mm) lengths of 22 swg wire (14 lengths for a 28-pin plug) into the row of holes on the edge of the PCB (starting at the R29 end) such that the wire projects on the component side. Now carefully insert the two memory devices into the PCB and solder them into position. MAKE ABSOLUTELY SURE THEY ARE THE RIGHT WAY ROUND and TAKE FULL PRECAUTIONS AGAINST STATIC DAMAGE. Crop the IC leads close to the foil side.

The largest PCB can now be assembled. First insert all the resistors, diodes, capacitors and transistor. Note that six of the resistors (all 47k) are mounted vertically as before with their free ends connected together and to a hole in the PCB. This time, however, the leads can be cropped close to the PCB. Next fit the DIL switch SW1, the two 74 series CMOS devices, and the remaining two memories. REMEMBER THAT ALL THE ICs ARE SENSITIVE TO STATIC. Make sure that they are the correct way round before soldering them in. Finally on this PCB, use four lengths of THIN insulated wire to make the long links shown on the overlay (W-W, X-X, Y-Y, Z-Z).

Having reached this stage it would be advisable to recheck all the solder joints on all the boards with a magnifying glass for accidental blobs, splashes or other faults.

From now on it will be virtually impossible to rectify any constructional errors. So check again!

Take the smaller PCB assembly and crop the wires projecting on the foil side so that the three links nearest the battery holder are virtually full length but the longest is at the edge and the shortest is farthest from it. Next crop the two lines of resistor leads and links so that the longest are about ¾" (18 mm) and the shortest are about ½" (12 mm), graded evenly across the board width. This is done to make the next operation easier.

Now the tricky bit. With great care feed the wires from the smaller PCB assembly into the corresponding holes in the larger PCB. Make sure that the wires are straight when you do this and don't allow the long links to cross each other on top of the memories. It should be possible to get the two PCBs down to about ¼" (6 mm) apart. If you can't do this it may be that the transistor or capacitor is in the way. Rectify this before carrying on. The transistor must be very close to the board. If this is not the problem one of the links may be bent. This is a fault, as when properly assembled all the links between boards are straight.

When this stage has been reached successfully, solder all the inter-board links and crop off the excess wire.

The end is now in sight. Examine the connector end of the assembly you have made and offer up the 24 or 28 pin plug you wish to use. It may be necessary to file a little material off one of the PCBs so that the plug will lie square against the two PCBs when the pins are soldered in position. Do this now if necessary.

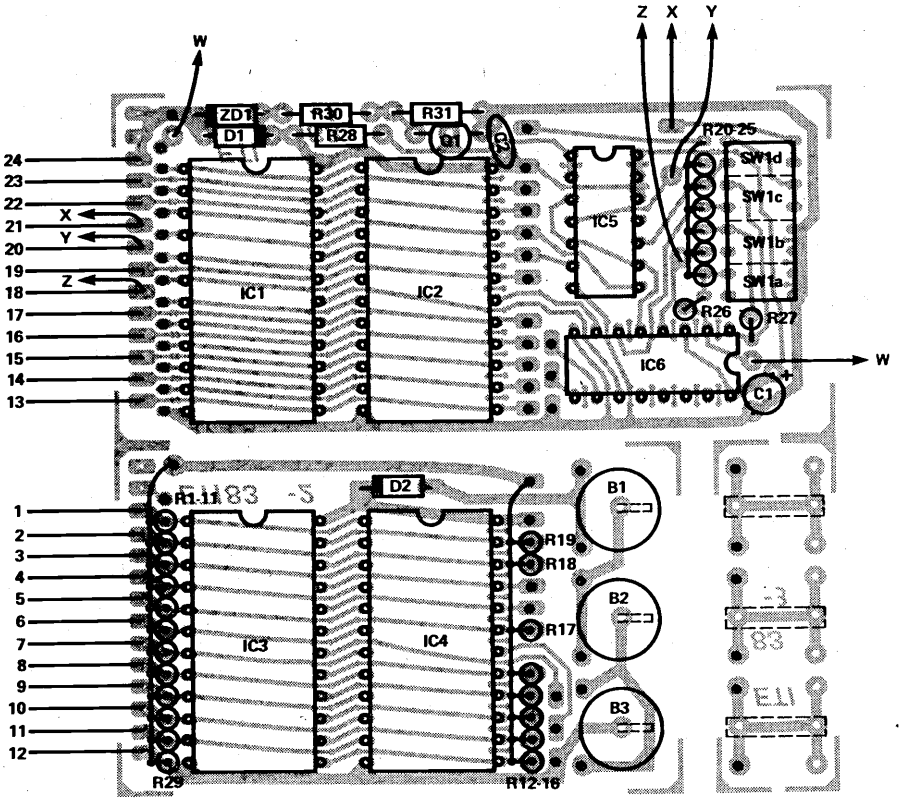



**Fig. 3 Component overlay for the PseudoROM. Remember to separate the three boards *before* soldering!**

## PARTS LIST

| | |
|---|---|
| **Resistors (all ¼W, 5%)** | |
| R1-27 | 47k |
| R28 | 10k |
| R29 | 1k0 |
| R30,31 | 1k5 |
| **Capacitors** | |
| C1 | 4u7 16 V tantalum |
| C2 | 10nF disc ceramic |
| **Semiconductors** | |
| IC1-4 | 6116 |
| IC5 | 74C32 or 74HC32 |
| IC6 | 74HC138 |
| Q1 | BC182L |
| D1,2 | OA91 |
| ZD1 | 2V7 400 mW zener |
| **Miscellaneous** | |
| SW1 | quad SPDT DIL switch |
| B1-3 | PX675 mercury cell |
| PCBs (see Buylines); 24 pin (or 28 pin) DIL header plug; wire, thin brass shim or other contact material. | |

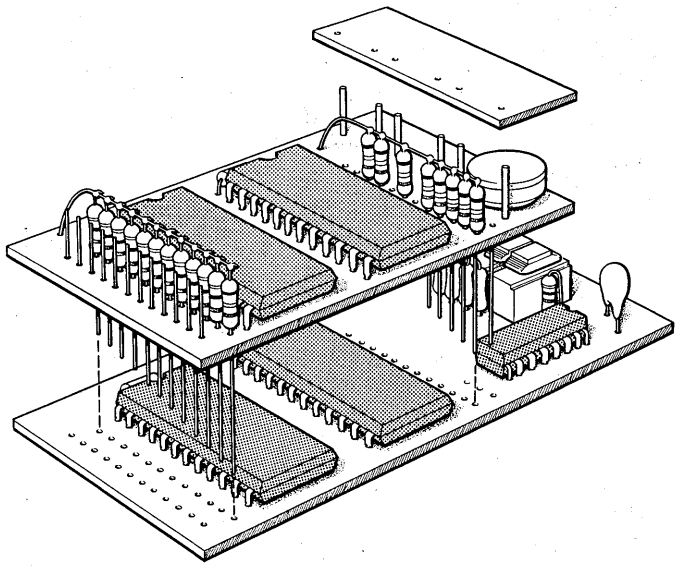
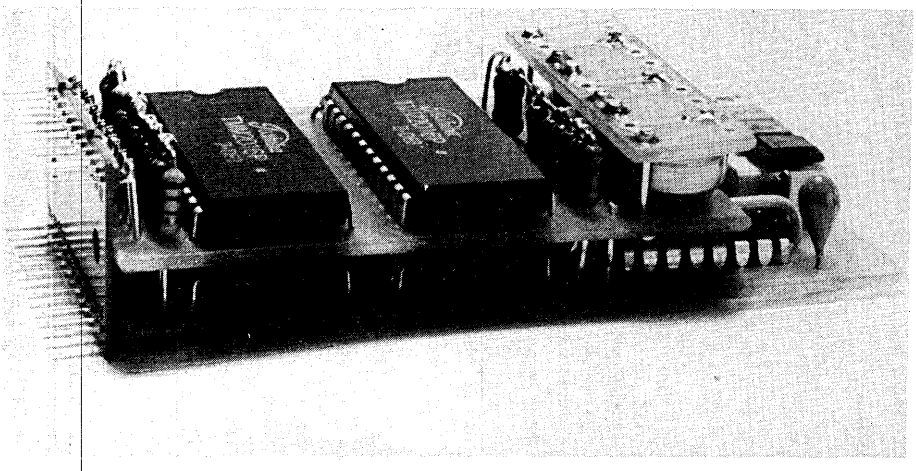


**Fig. 4 An exploded view to help you through the traumas of construction. Compare with the photo below.**

If there is a large blob of solder on any of the three pads where the links have been soldered already, remove some of it with a solder sucker or piece of fluxed braid. This is to allow the plug pins to seat down onto the PCB foil. Align the tags of the plug labelled with pins 13 to 24 (or 15 to 28 for the larger plug) against the pads along the edge of the largest PCB in the assembly. Solder them carefully to the pads such that pin 24 connects to the innermost of the three joined pads (or pin 28 joins to the outermost in a 28 pin plug). This should cause the remaining pins to be forced against the link wires coming from the middle PCB. With any luck they will be in the right place to be soldered but check first. Pin 12 (or 14 on 28-way plug) should lie next to R29 (1k0). If all is correct then solder them into position and crop off any excess wire.

The assembly should now be complete and after inserting the mercury cells is ready for use. These cells should be inserted with the flat or outer part of the case nearest to the small PCB. This is the POSITIVE terminal. The rounded stud is negative and goes to the middle PCB. Make sure the batteries are pushed well in so that they locate against the pairs of supporting struts and do not touch each other.



**The assembled project is neat and compact — at least it should be if you've done things correctly.**

Otherwise one battery will be shorted out.

## Using the PseudoROM

It is recommended that the unit is only inserted or removed from its socket when the equipment using it is switched off. To make use of the unit the data must first be written into it in 2K blocks. For this a 24 pin socket must be provided with a R/$\overline{W}$ signal on pin 21 and other signals as for a standard 2K EPROM. SW1d should be set to connect the pin 21 input through to IC5b and SW1a and SW1b set to select the 2K segment to be used. Data can now be entered into the unit. Change the SW1a and SW1b settings to load up the other segments as required, remembering that SW1a is the MSB and SW1b the LSB of the address.

When placed in a suitable 24 or 28 pin ROM socket, SW1d should be set so that pin 21 (23) is no longer connected to IC5b: this prevents accidental overwriting of the data. SW1c can now be used to select either 2K ROM simulation (connecting to IC6 pin 6) or 4K in a 24 pin socket and 8K in a 28 pin socket (connecting to IC6 pin 1). Whichever mode is selected, the segment select addresses selected by SW1a and SW1b will be overridden by the relevant external address lines. This saves much extra switching but anomalies may occur if the 2K ROM mode is selected in a 28 pin socket.

Note that this module is only suitable for use in EPROM sockets with single rail power supplies and the $\overline{CE}$ or $\overline{CS}$ signal on pin 18 (20 for a 28 pin socket) and the $\overline{OE}$ signal on pin 20 (22 for a 28 pin socket). Pin 21 (23 for a 28 pin socket) must be logic high level: A11 address or $\overline{WE}$ as appropriate for the mode used. Unfortunately this rules out some devices such as the three-rail TMS2716, the TMS2532, and under some circumstances the TMS2516. If in doubt, consult the data sheets for your particular device and circuit. **Warning:** DO NOT TRY TO PROGRAM THIS UNIT IN AN EPROM PROGRAMMER — the high voltage will damage it permanently. And take care when handling the device out of its socket — once assembled it's unlikely that any of the CMOS chips will be blown by static, but if it should happen, your guess is as good as ours how to replace them.

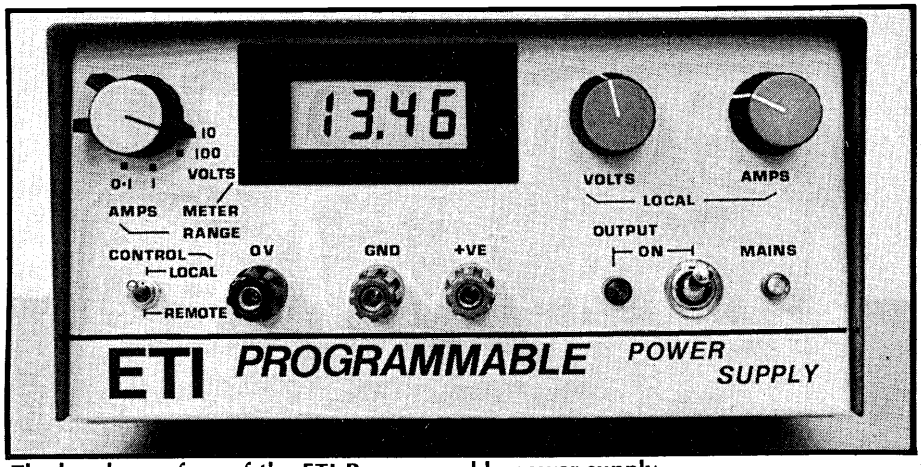
# PROGRAMMABLE POWER SUPPLY

**We hear a lot these days about computing power but here it is, literally. With this PSU plugged into a suitable port on your micro you can control your voltage and current bit by bit. There's manual control, too. Design by Phil Walker.**

This versatile piece of equipment is basically a programmable power supply. It allows you to set the output voltage and/or current to very close limits by means of your computer keyboard without actually touching the unit itself. Even more useful, in some cases, is that a sequence of voltage and/or current levels can be programmed in advance. The unit has a range of 0–25V5 in 100 mV steps and 25 mA to 1A6 in 25 mA steps.

The prime controlling element in the power supply section is an LM317 integrated voltage regulator device. In this project it is used as a high-gain self-protecting power transistor. In order to get the rated output from the device under all operating conditions, two power rails are used. The lower one, 17 V, is used while the required output is less than 12 V, the higher supply rail of 34 V is used when the required output is greater than 12 V. The purpose of this configuration is to keep the dissipation in the LM317 as low as possible. This is necessary because the LM317 will not allow the full 1A6 output to flow if there is more than about 17 V across it.

The reduced power dissipation in this circuit arrangement allows us to have a constant current output characteristic over the whole range instead of a 'foldback' limiting circuit.




**The handsome face of the ETI Programmable power supply.**

## Construction

The project was constructed in the specified Newrad case, but it's a fairly tight fit and it may well be easier to use the next size up (S2/38) to allow more elbow room. Construction of the PCB is straightforward so long as polarity of components is observed where relevant. We used PCB plugs and sockets for the digital input as this is most convenient. IC8, Q3, D4 and D5 are mounted on the heatsink and wired through a grommet in the rear panel. It makes things easier here if some PTFE insulators are used to hold the diodes and connecting wires. Use insulating mica washers with heatsink compound under IC8 and Q3 to conduct the heat away while preventing unwanted short circuits. Together with the heatsink, the mains switch, fuse, mains cable and control input socket are all mounted on the rear panel.

It will help considerably if the front and rear panel components are fitted and wired as far as possible before the chassis is finally fitted. Make sure, however, that you leave enough room to insert the chassis afterwards. Note that R4, 25, 27 to 31 are mounted on their associated front panel components, though we did find room on the PCB for R4 after the prototype was completed, and it is shown thus on the overlay.

The transformer, main capacitors, bridge rectifier and PCB together with D6 and R26 are mounted on the chassis supplied with the case. Some solder tags and tagstrip will be useful for mounting the smaller components and connecting to the capacitors. The large capacitors used in the prototype were mounted horizontally using two clips each.

### BUYLINES

**Not too much in the way of unusual components for this project: the ZN428 and D-range connector may be hard to track down but Technomatic can supply both parts. The case we used is available from Newrad Instrument Cases Ltd, Tiptoe Road, Wootton, New Milton, Hants BH25 5SJ, telephone New Milton 615774; the PCB can be obtained, as usual, from our PCB Service as advertised. The optional DVM module, the DPM05, is stocked by Lascar Electronics, Oakland House, Reeves Way, South Woodham Ferrers, Chelmsford, Essex CM3 5XQ, telephone 0245 329797.**

## Setting Up

Having constructed the project and checked for wiring errors very carefully, put RV1 and RV2 to minimum, close SW3 and set SW2 to local. Switch on and check that the voltages on C1, C2 and C7 are +34, +17, and −34 (± 5 V or so). If not, check the wiring again. Now make sure that the +5V and −5 V

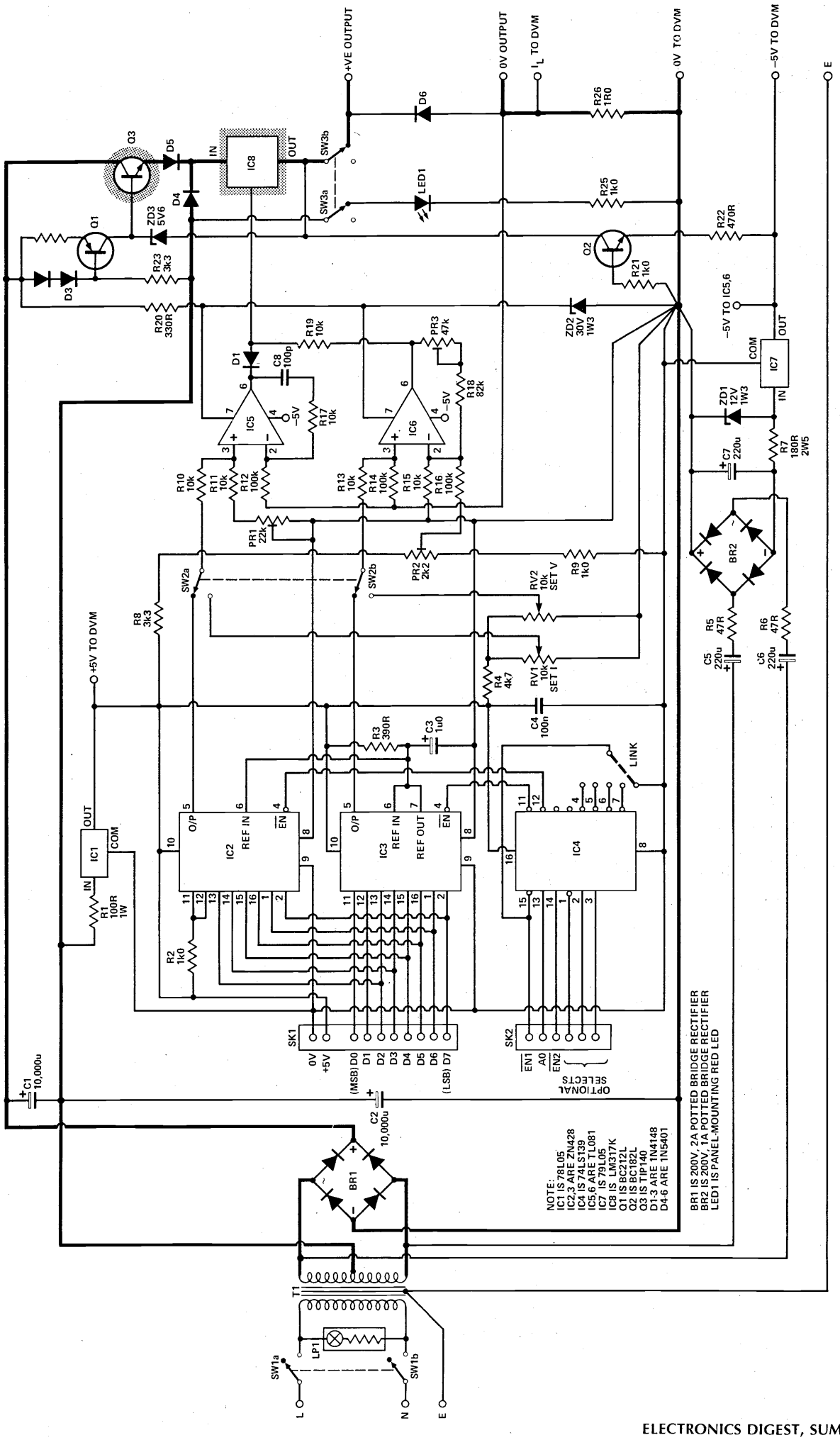



**Fig. 1 Circuit diagram for the power supply. The heavier tracks carry the high current. Note the central earthing system.**

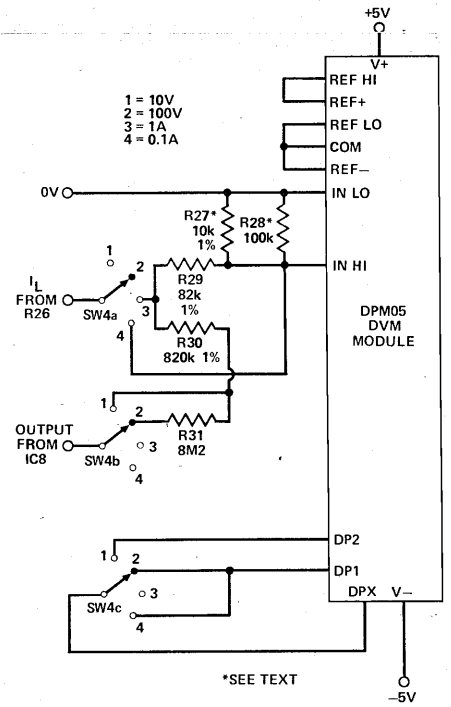
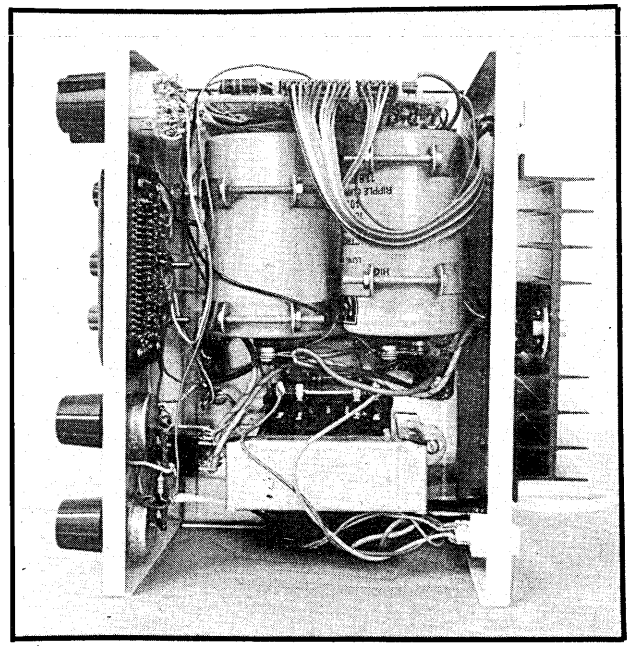



Fig. 2 Circuit for the optional DPM.



This is how we packed everything into the case specified in Buylines but it's pretty tight! You may wish to use the next size up.

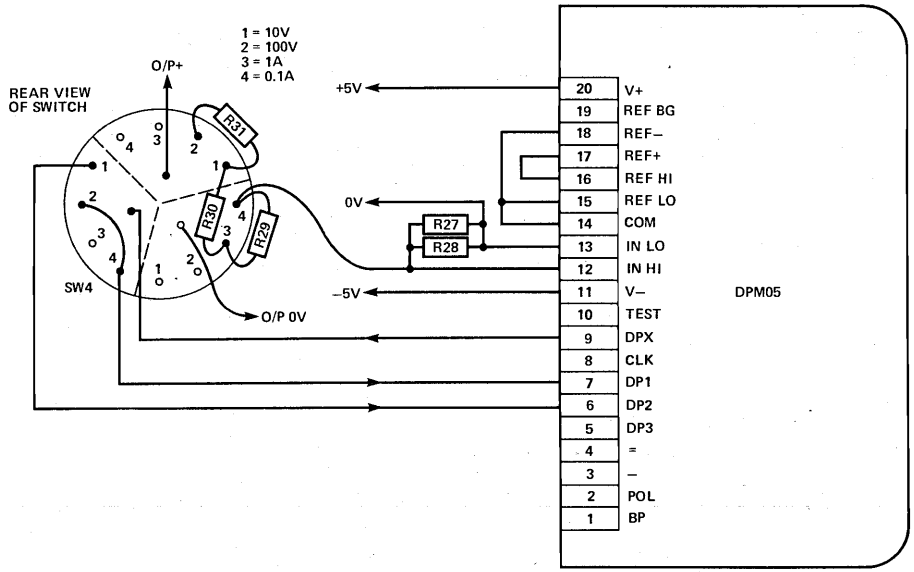



Fig. 3 How to wire up the range switch and its resistors to the DPM.

## HOW IT WORKS

### POWER REGULATOR STAGE

The 12-0-12 V from the transformer is rectified by BR1 and smoothed by C1 and C2. This is connected to give a 0 V rail, a 17 V and 34 V positive supply rails. Also driven from the transformer, BR2, C5, C6, C7, R5 and R6 act as a full wave voltage doubler to provide a negative supply rail for the control circuitry.

The 17 V rail is connected to the input side of the LM317 via D4 while the 34 V rail is connected to the same point via Q3 and D5. Q1, R24, D2, 3 and R23 form a simple constant current source which tends to turn Q3 on. ZD3 acts to maintain the base of Q3 at 5V6 above the output level. By emitter follower action in Q3, the input to the LM317 will be maintained at about 5V6—1V8 higher than the output terminal. However, this is only true while D5 is forward biased, ie while $V_{OUT} \times (5.6 - 1.8)$ is greater than 17 V. When the output voltage is less than the level required, current will no longer be supplied via Q3 and D5 but will come via D4.

### AUXILIARY SUPPLIES

The op-amps are supplied from +30 V and —5 V to enable them to cover the output voltage range while staying within their absolute maximum supply capability. These voltages are generated by a resistor and zener diode combination for the 30 V supply and an IC regulator for the —5 V supply. This latter device is preceded by a resistor and zener diode combination to drop the output from the voltage doubler circuit down to a safer level. The logic circuitry is supplied with +5 V from IC1.

### THE VOLTAGE REGULATOR CIRCUIT

IC8 is the main regulator device in this project; it is treated as a self-protecting transistor except that its 'base' voltage is about 1V2 below the output level. The current into or out of this terminal is very small. IC6 is connected as an amplifier with a gain of 10. It also removes the effect of the current sense resistor R26 on the final output, and allows the output to be set to 0 V out for 0 V in by means of PR2.

The input to IC6 is selected by SW2 and is a 0 to 2V55 variable DC level from either RV2 for local (manual) control or from IC3 for remote (programmable) control. The output from IC6 goes via R19 to the common terminal of IC8: the voltage at this point determines the output voltage from the unit.

### THE CURRENT REGULATOR CIRCUIT

IC5 is the circuit element which detects the voltage generated by the output current flowing through R26. This voltage is compared with the current limit input voltage selected by SW2 and if the current limit is exceeded, the output of IC5 is driven towards the negative rail. This eventually pulls the common terminal of IC8 low enough via D1 to stabilise the current at its preset limit.

R10, 11 and PR1 set the sensitivity of the circuit while R12, 17 and C8 reduce the high frequency gain of the circuit to help stability.

### THE PROGRAMMABLE INPUT SECTION

IC2 and 3 are eight bit A-to-D converters with internal latches, although only six bits of IC2 are programmable as greater resolution was not felt to be usable without greater complexity in the current limit circuit. This also means that the unit is not completely off under any circumstances. All eight bits are available in IC3 for setting the output voltage. R3 and C3 supply the reference voltage generator in IC3 which in turn supplies the D-to-A converters in both IC2 and 3. Address decoding and device selection is available in IC4 which is a dual 1-of-4 selector. Two outputs from one section of this IC drive the enable inputs of the DACs and the three inputs to this section can be used on their own with external address decoding or in conjunction with an output from the other section (selected by a suitable link on the PCB) if required.

### THE PANEL METER MODULE

This optional part of the circuit allows the user to monitor the output voltage or current from the unit: SW4 is used to select the mode and range required. R27 to 31 form a divider network to scale the meter to the correct measuring range. The module used has a nominal input of 100 mV for a reading of 1000. It also has provision for selecting the decimal point position and this is made use of by a section of SW4. It is necessary to use +5 V and —5 V to supply the meter as it will not read correctly if the input terminals are at or near the negative supply. The module has its own internal band-gap voltage reference and if accurate resistors are used for the divider network a highly accurate readout will be obtained.

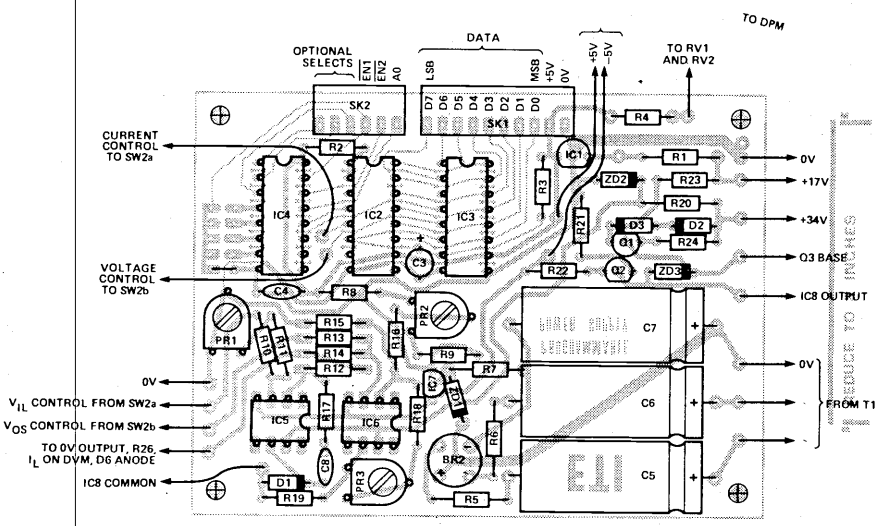
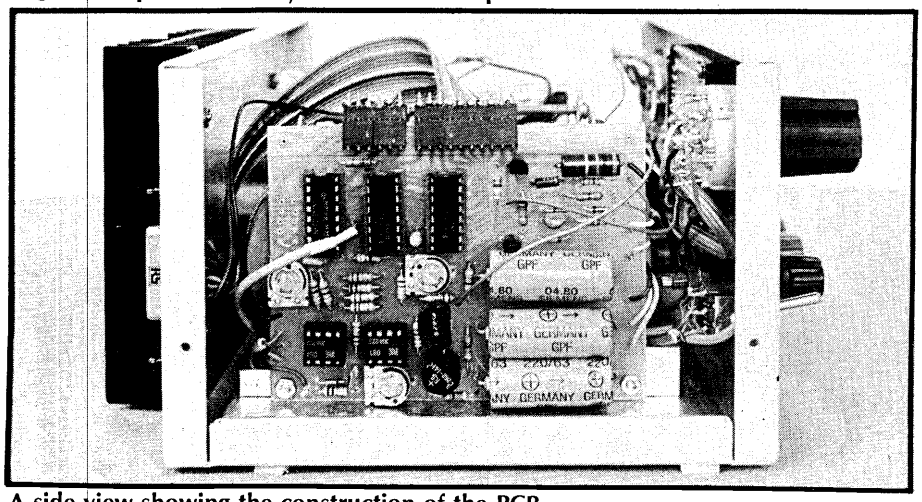



**Fig. 4 Component overlay. Note some components are mounted off-board.**



**A side view showing the construction of the PCB.**

rails are correct on the PCB. If not, check component polarities and placement. Rotate SW4 through all its positions and check that it reads at or somewhere near zero in all positions. With SW4 in the 10 V range (or a 10 V meter connected at the output) adjust PR2 for 0 V output. Connect a voltmeter to RV2 wiper and adjust RV2 until it reads 2V55 with respect to the 0 V rail. Adjust PR3 until the output reaches 25V5 (SW4 in 100 V position).

Now connect the voltmeter to RV1 wiper and adjust RV1 until it reads 2V55. Turn RV2 to minimum and connect an ammeter to the output terminals. Advance RV2 a little and a current of between 1 and 2 amps should flow. Adjust PR1 until this current is 1A6.

If all these steps have been accomplished without smoke, the supply should be operational. The only thing left to check is the digital control section. Connect all the data inputs on SK1 to 0 V and connect $\overline{EN1}$ to 0 V using the link indicated beside IC4 in Fig. 4. Now, with A0 at 0 V take $\overline{EN2}$ low momentarily. Repeat this with A0 high. Now check that the outputs from IC2 and 3 are at 0 V (IC2 will be 30 mV or so positive). Set SW2 to 'remote' and readjust PR2 if necessary.

Set all the data inputs to +5 V and latch in this new information using A0 and $\overline{EN2}$ as in the previous paragraph. Readjust PR3 and (with care) PR1. The unit should now be ready for use. Note, when doing the above procedures it may be advisable to provide pull-up resistors on unconnected inputs.

The A0 line selects the voltage D-to-A converter when low and the current D-to-A converter when high. $\overline{EN1}$ could be the select line while $\overline{EN2}$ would be the R/$\overline{W}$ on 6502-type systems, or $\overline{WR}$ on the Z80 etc. For most applications it is advisable that there should be some means of isolation between the control processor and this unit to prevent earth loops and other undesirable effects; this may well be in the form of opto-isolators. Pads are provided to allow the $\overline{EN1}$ link to be repositioned to use the other section of IC1 and gain extra decoding capability, should this be necessary.

## PARTS LIST

Resistors (all ¼W, 5% except where stated)

| | |
|---|---|
| R1 | 100R 1 W |
| R2,9,21,25 | 1k0 |
| R3 | 390R |
| R4 | 4k7 |
| R5,6 | 47R |
| R7 | 180R 2W5 |
| R8,23 | 3k3 |
| R10,11,13, 15,17,19 | 10k |
| R12,14,16 | 100k |
| R18 | 82k |
| R20 | 330R |
| R22 | 470R |
| R24 | 120R |
| R26 | 1R0 2W5 |
| R27* | 10k 1% |
| R28* | 100k |
| R29 | 82k 1% |
| R30 | 820k 1% |
| R31 | 8M2 5% or better |

*R27 and R28 may be replaced with a single 9k1 1% resistor

Potentiometers

| | |
|---|---|
| RV1,2 | 10k linear |
| PR1 | 22k miniature horizontal preset |
| PR2 | 2k2 miniature horizontal preset |
| PR3 | 47k miniature horizontal preset |

Capacitors

| | |
|---|---|
| C1,2 | 10,000u 25 V can electrolytic |
| C3 | 1u0 35 V tantalum |
| C4 | 100n ceramic |
| C5-7 | 220u 63 V axial electrolytic |
| C8 | 100p ceramic |

Semiconductors

| | |
|---|---|
| IC1 | 78L05 |
| IC2, 3 | ZN428 |
| IC4 | 74LS139 |
| IC5,6 | TL081 |
| IC7 | 79L05 |
| IC8 | LM317K |
| Q1 | BC212L |
| Q2 | BC182L |
| Q3 | TIP140 |
| D1-3 | 1N4148 |
| D4-6 | 1N5401 |
| ZD1 | 12 V 1W3 zener |
| ZD2 | 30 V 1W3 zener |
| ZD3 | 5V6 400 mW zener |
| LED1 | panel-mounting red LED |
| BR1 | 200 V, 2 A potted bridge rectifier |
| BR2 | 200 V, 1 A potted bridge rectifier |

Miscellaneous

| | |
|---|---|
| SW1 | two-pole mains-rated on-off rocker switch |
| SW2 | two-pole miniature changeover toggle switch |
| SW3 | two-pole on-off toggle switch (5 A @ 30 V) |
| SW4 | three-pole four-way rotary switch |
| FS1 | 20 mm 1 A fuse |
| T1 | 0-12, 0-12 V, 50 VA mains transformer |
| LP1 | mains neon indicator |

PCB (see Buylines); DPM05 LCD DVM module (see Buylines); PCB plugs and sockets if required (one off 10-way and two off three-way); 15 way D-range connector (see Buylines); three off screw terminals; three knobs; can capacitor clips; tag strip; transistor mounting kits and heatsink compound for LM317 and TIP140; heatsink (2.0° C/W); cable and hardware; case, Newrad type S2/37 (see Buylines).

# MICROTUTOR PART 1

**BASIC, Pascal and the like are OK but if you want to get the most out of your micro then machine code is where it's at. This machine code tutor will help you throw off your chains; you have nothing to lose but your SYNTAX ERRORS. Design and development by Tangerine Computer Systems.**

Real computing was able to escape from the lofty realms of the mainframe and into the hands of the layman because of two major developments. The first was when large-scale integration techniques enabled the production of small, versatile microprocessor chips which were adapted into small, relatively low-cost microcomputers. (The microprocessor was originally conceived for use in industrial machine control.) The second was the invention of the BASIC computing language, which allowed the complete beginner to write working computer programs easily and without any knowledge of the system hardware he was working on.

Unfortunately such ease of use can only be obtained by means of a compromise, and the major drawback of the average microcomputer is speed — or rather lack of it. BASIC is painfully slow when compared to the speed of operation of the microprocessor, because each BASIC statement requires a series of machine code subroutines to be executed. Obviously better use can be made of the system hardware if programs are written directly in the language of the microprocessor — machine code. Direct access to the processor means access to swift, versatile and efficient programming.

## All Systems Go

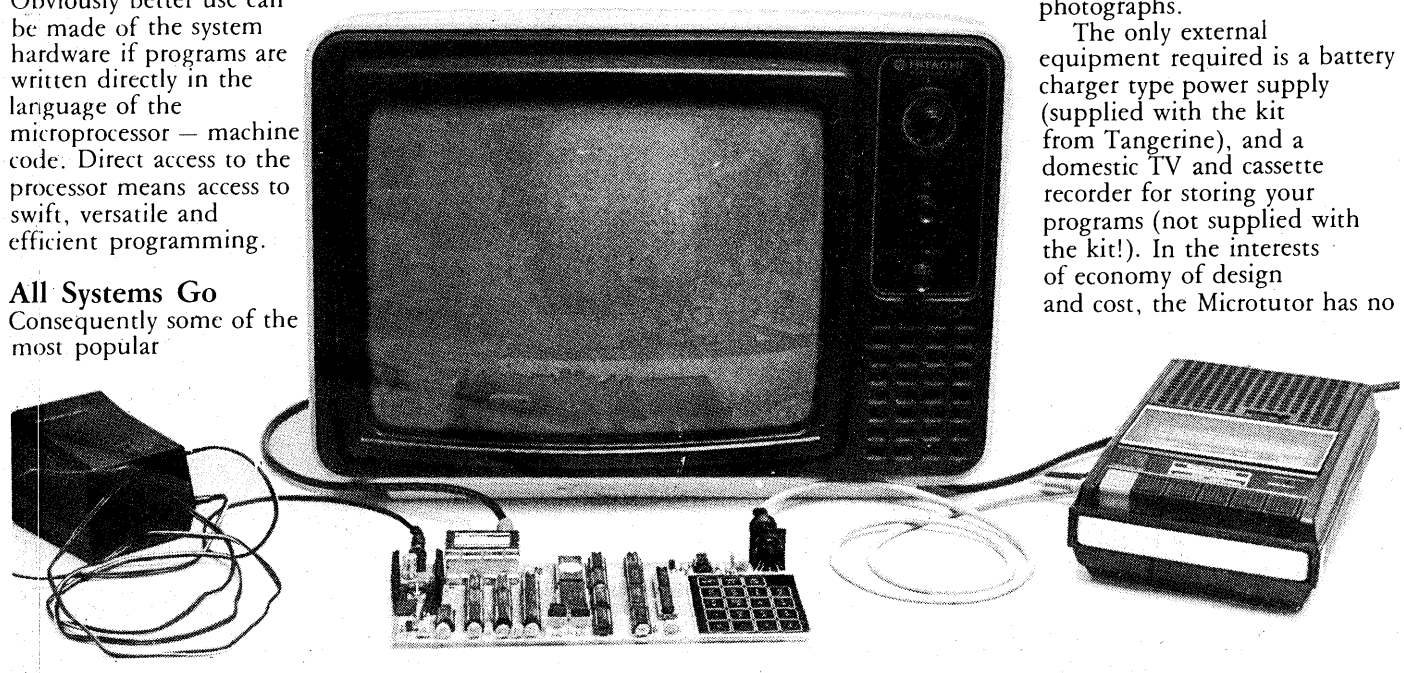
Consequently some of the most popular micro systems haven't had BASIC as a language at all! Typical products include KIM, SYM, the late-lamented MK14 and the AIM 65, still a well-known 6502 machine code development system. And now there's the ETI Microtutor, a project we're hoping will coax more of our readers away from the security of BASIC and into machine code, the real stuff of microcomputing.

The Microtutor has been designed for us by Tangerine Computer Systems, who were responsible for the highly popular Space Invasion game we published in November 1980. The monitor program for the system is based on TANBUG, as used in the Microtan 65 computer from the same company; this is a remarkably powerful monitor with many useful programming commands available to make machine code easier to manipulate. The hardware is designed around the 6503, a member of the 6502 family with the same instruction set but an address range of only 4K as opposed to the more usual 64K. Thus the microprocessor is more compact (only 28 pins instead of 40), but any software written for it will also run on the Microtan 65. Furthermore, once you're familiar with the instruction set of the 6500 family you should be able to write machine code programs for home computers based on the 6502 such as the PET, Acorn System 1, Acorn Atom, Superboard, UK101, Apple, VIC and the BBC machine, which should give you plenty of scope! (Indeed, the PET is designed so that the entire system can be reconfigured from software, but you'll have to become pretty good to attempt that!)

## Give Us The Tool

The Microtutor is a very sophisticated and professional-looking piece of equipment. The hex keypad is mounted directly on the PCB, as are the power supply regulator, cassette interface, and a VDU with UHF output on channel 36 for direct connection to a domestic TV set (one up on the AIM 65!). The PCB is incredibly compact and a masterpiece of design, as you can see from the photographs.

The only external equipment required is a battery charger type power supply (supplied with the kit from Tangerine), and a domestic TV and cassette recorder for storing your programs (not supplied with the kit!). In the interests of economy of design and cost, the Microtutor has no



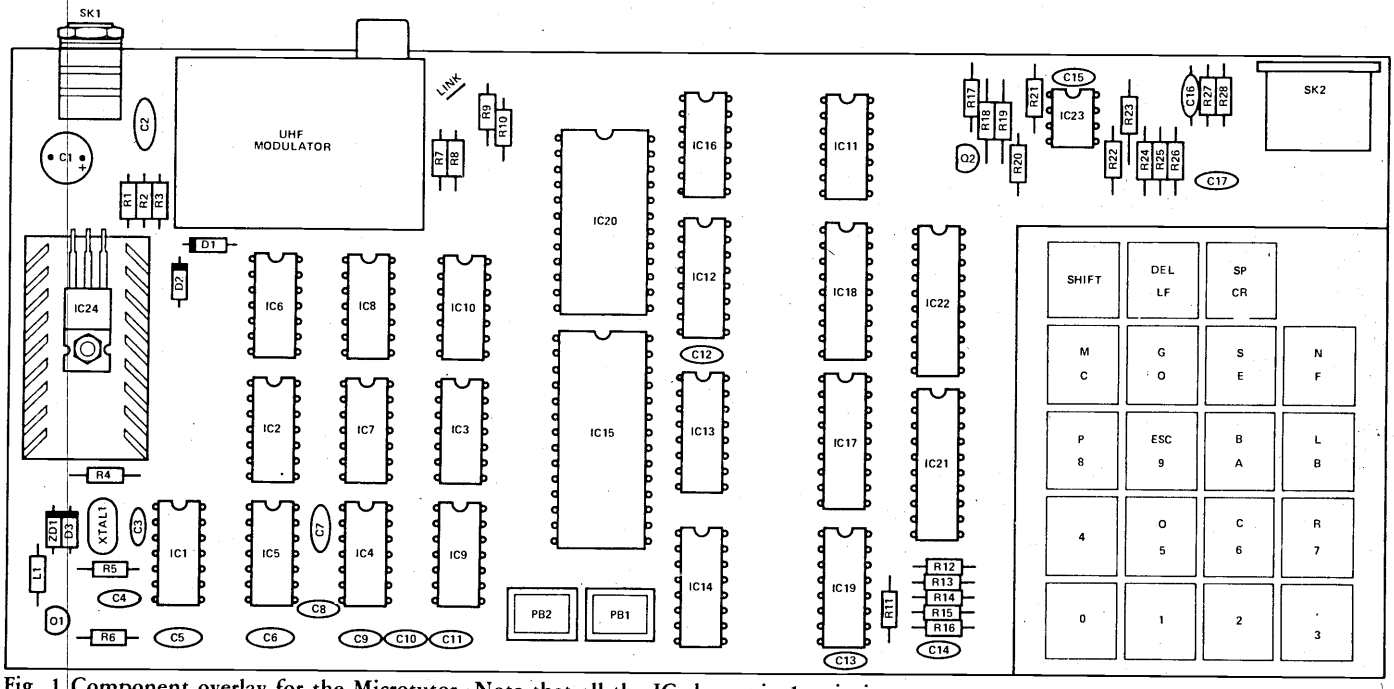



Fig. 1 Component overlay for the Microtutor. Note that all the ICs have pin 1 pointing upwards.

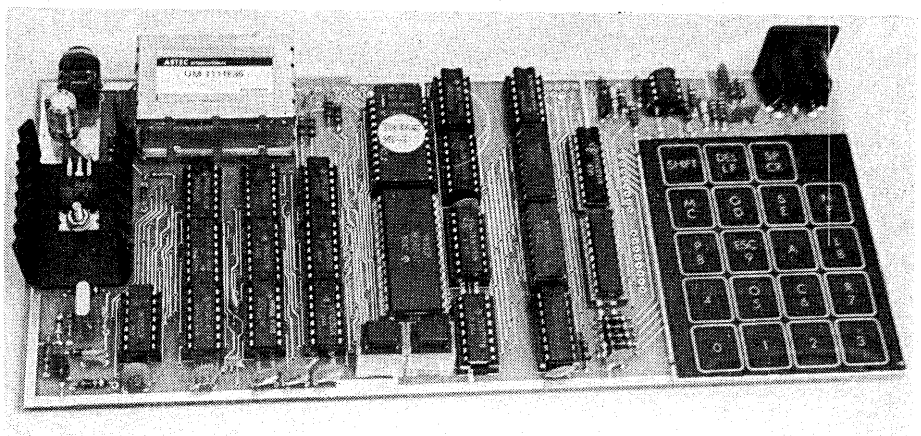
graphics and the ASCII set repeats through the entire range of 256 character codes. Only upper case is available (see Table 1).

Apart from a teaching aid for machine code programming, the Microtutor is also intended to be used for I/O experiments; thus all the bus lines and control signals are brought out to pads on the PCB so that external circuitry can be connected.

The only thing to remember about machine code is that it is very unforgiving of errors. Unlike BASIC, a programming blunder won't produce a polite message on the screen and a chance to try again; more often than not your program and data may be corrupted, or the thing gets locked into an endless loop until it disappears up its own data bus. When developing software for a home computer there is often little choice at this point but to switch off the machine to cure things, causing loss of program and laborious re-entry. We've had some nasty experiences in the office with the PET this way! On the Microtutor things are more civilised; two buttons are

## SPECIFICATION

| | |
|---|---|
| CPU: | 6503 (addresses 4K) |
| ROM: | 2K containing monitor program. EPROM is 2716 (5 V version) |
| RAM: | 1K. Used for user program and memory-mapped VDU |
| I/O: | 1K space available |
| Display: | 16 rows by 32 characters, upper case only |
| UHF Output: | Channel 36 |



The PCB is very compact and uses thin tracks, so a fine bit is essential on your soldering iron.

## PARTS LIST

| | |
|---|---|
| Resistor (all ¼W, 5%) | |
| R1 | 75R |
| R2 | 470R |
| R3 | 220R |
| R4-17 22 27 | 1k0 |
| R18, 24 | 10k |
| R19, 23 | 120k |
| R20, 21 | 2k2 |
| R25, 26, 28 | 22k |
| Capacitors | |
| C1 | 100u 10 V PCB electrolytic |
| C2, 5, 6, 9-17 | 100n ceramic |
| C3, 8 | 10n ceramic |
| C4, 7 | 100p ceramic |
| Semiconductors | |
| IC1, 8 | 74LS04 |
| IC2 | 74LS73 |
| IC3, 9 | 74LS393 |
| IC4 | 74LS21 |
| IC5, 16 | 74LS74 |
| IC6 | 74LS08 |
| IC7 | 74LS11 |
| IC10 | 74LS00 |
| IC11 | 74LS32 |
| IC12-14 | 74LS157 |
| IC15 | 6503 |
| IC17, 18 | 2114 |
| IC19 | 86S64BWF |
| IC20 | 2716 |
| IC21 | 74LS374 |
| IC22 | 74LS244 |
| IC 23 | LM358 |
| IC24 | 7805 |
| Q1, 2 | BC184L |
| D1-3 | 1N4148 |
| ZD1 | 6V8 400mW zener |
| Miscellaneous | |
| XTAL1 | 6 MHz crystal |
| L1 | 100uH choke |
| SK1 | miniature charger socket (PCB-mounting) |
| SK2 | 5-pin DIN socket (PCB-mounting) |
| PB1, 2 | push-buttons (PCB-mounting) |

PCB (see Buylines); hex keypad; UHF modulator type UM1111E36; IC sockets; PCB-mounting heatsink for regulator.

provided, reset and interrupt. Both wil get you back into the monitor from a faulty program if it gets out of hand, without clearing the memory; interrupt will do so without losing any breakpoints you may have set (breakpoints are explained fully later).

## Construction

The PCB is double-sided but through-hole plated, so components only have to be soldered on the underside. Fit all the low profile components first, ie the link, resistors, diodes and choke, then the keypad. This is stuck to the component side of the board with double-sided sticky pads; insert the connecting wires through the holes in the PCB first first, secure the keypad in position with the pads and then solder the connections.

Now you can fit all the IC sockets and capacitors but don't insert the ICs until later. Solder the choke, crystal and transistors in place, then the UHF modulator. The voltage and its heatsink are bolted to the PCB and no insulating washers are required.

Finally mount the two sockets (power and cassette) and the large push-buttons, then insert all of the ICs paying great attention to the device type.

## Ready To Go

Once you've double-checked everything, you're ready to connect the power supply and TV set. The modulator is connected to the TV using UHF coaxial cable with a phono plug at one end and the usual coax plug at the TV end. Switch on the set and tune the TV until you get a steady black-and-white picture — at this point the screen will contain garbage. Press the RESET button and the screen should announce TANBUG. You may now use the keypad, as described below, to enter the wonderful world of machine code.

As an example of the sort of power you've got at your fingertips, here's a direct comparison. Some time ago it was necessary for someone in the office to check all four-digit numbers for certain combinations of digits. A BASIC program on the PET took about five minutes to write and about 15 minutes to run. The same problem, when solved by the Microtutor, took somewhat longer to write the program, but checked through all 10000 numbers in three seconds!

## All About TANBUG

The TANBUG monitor program is located in 2K bytes of read only memory (ROM) at the top of the address space ie pages 8-15. It contains facilities to enter, modify, run and debug programs. TANBUG will only operate in the memory map of the Microtan system, it is not a general purpose 6502 software package and has been specifically written for Microtan/Microtutor.

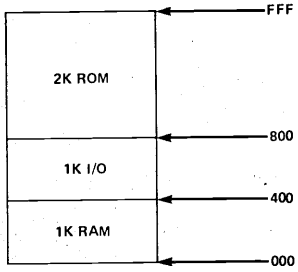



Fig. 3 Memory map.

Locations 2ØØ-3FF (pages 2 and 3) are the visual display memory — TANBUG writes to these locations whenever a command is typed to the monitor. Locations in pages 4 to 7 are the addresses of the peripheral attachments, eg keyboard. Locations 1ØØ-1FF (page 1) are used as the stack by the microprocessor. Since the stack is of the push-down variety it follows that the whole of this will not be used as stack storage in the majority of programs. TANBUG requires to use locations 1FØ-1FF as stack storage (only 16 locations). The rest of this area is free for user programs. Locations 4Ø-FF are also available as user RAM, the preceding locations Ø-3F being reserved for use by TANBUG. User programs which do not use the stack may therefore be loaded anywhere in the area 4Ø-1EF. For user programs which do use the stack, the user must calculate how many stack locations are required and reduce the upper limit accordingly.

**TABLE 1**

| Hex | Character | Hex | Character |
|---|---|---|---|
| 20 | Space | 40 | @ |
| 21 | ! | 41 | A |
| 22 | " | 42 | B |
| 23 | # | 43 | C |
| 24 | $ | 44 | D |
| 25 | % | 45 | E |
| 26 | & | 46 | F |
| 27 | ' | 47 | G |
| 28 | ( | 48 | H |
| 29 | ) | 49 | I |
| 2A | * | 4A | J |
| 2B | + | 4B | K |
| 2C | , | 4C | L |
| 2D | — | 4D | M |
| 2E | . | 4E | N |
| 2F | / | 4F | O |
| 30 | 0 | 50 | P |
| 31 | 1 | 51 | Q |
| 32 | 2 | 52 | R |
| 33 | 3 | 53 | S |
| 34 | 4 | 54 | T |
| 35 | 5 | 55 | U |
| 36 | 6 | 56 | V |
| 37 | 7 | 57 | W |
| 38 | 8 | 58 | X |
| 39 | 9 | 59 | Y |
| 3A | : | 5A | Z |
| 3B | ; | 5B | [ |
| 3C | < | 5C | \ |
| 3D | = | 5D | ] |
| 3E | > | 5E | ^ |
| 3F | ? | 5F | — |

The keypad is used as follows, its layout being shown in Fig. 1. TANBUG interrogates the keypad for a depressed key, then translates the matrix encoded signal into an ASCII character which it puts up on the visual display. Because of the limited number of keys it has been necessary to incorporate a shift function on the keypad. So, to obtain the character P for example, the user presses and releases SHIFT, then depresses and releases P. The SHIFT key contains a self-cancelling facility — if the user presses SHIFT twice in succession the pending shift operation is cancelled. So as an example, using the two keys SHIFT and 8 the operations SHIFT P yields P on the display. SHIFT SHIFT P yields 8 on the display. DEL deletes the last character typed. Repeated deletes erase characters back to the beginning of the line.

Having described some of the background to TANBUG it is now possible to describe the commands and syntax of TANBUG, ie how to use it. An example is shown later on. All numerical values of address, data and monitor command arguments are in hexadecimal. The symbol <CR> means on depression of the carriage return key, <SP> the space key, and <LF> line feed. In all examples, text to be typed by the user will be in bold type, while TANBUG responses will not. ■ indicates the cursor. <ADDR> means a hexadecimal address, <ARG> means hexadecimal data and <TERM> means one of the terminators <CR>, <SP>, or <LF>.

All commands are of the form

<COMMAND><TERM>

or

<COMMAND><ARG> TERM>

or <COMMAND><ARG>,

<ARG><TERM>

or <COMMAND><ARG>,

<ARG>, <ARG><TERM>

where <COMMAND> is one of the mnemonic commands and <ARG> is a hexadecimal argument applicable to the command. It should be noted at an early stage that the longest argument will contain four hexadecimal characters. If more are typed all but the last four are ignored. As an example consider the memory modify command M1234ØØ78 <CR>. In this case location ØØ78 will be modified or examined as all but the last four characters are ignored.

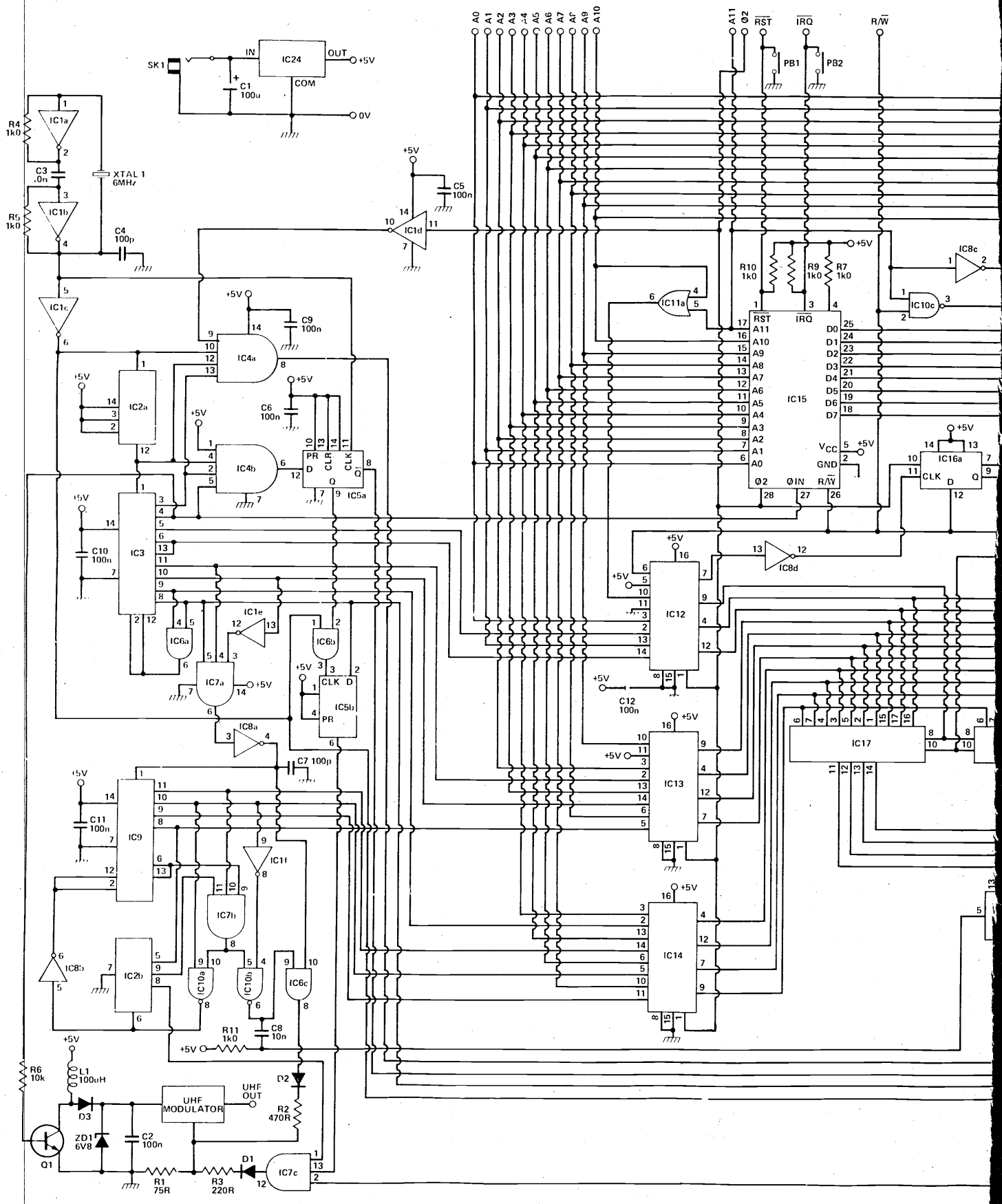
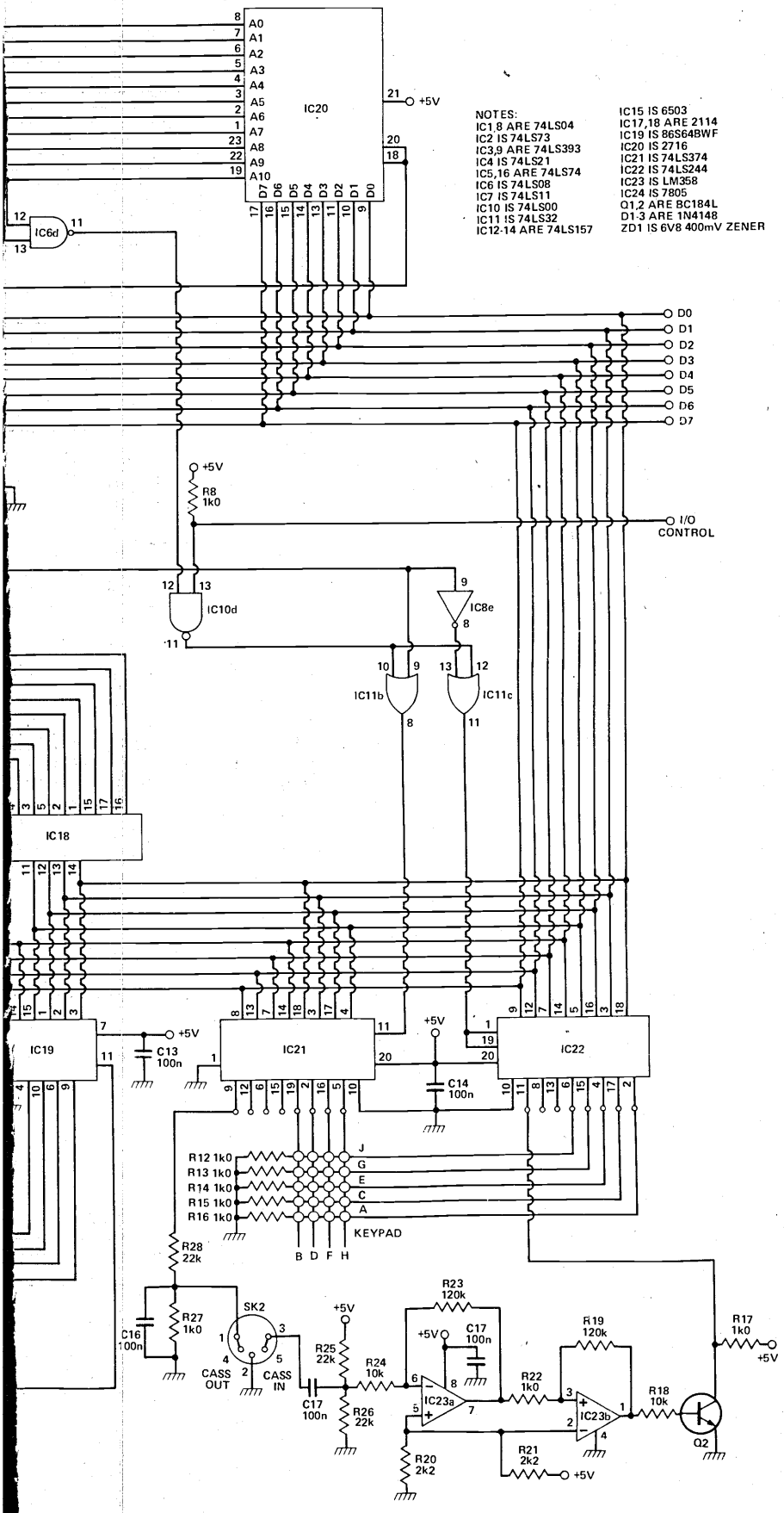



Fig. 2 Complete circuit diagram for the Microtutor.




NOTES:
IC1,8 ARE 74LS04
IC2 IS 74LS73
IC3,9 ARE 74LS393
IC4 IS 74LS21
IC5,16 ARE 74LS74
IC6 IS 74LS08
IC7 IS 74LS11
IC10 IS 74LS00
IC11 IS 74LS32
IC12-14 ARE 74LS157
IC15 IS 6503
IC17,18 ARE 2114
IC19 IS 86S64BWF
IC20 IS 2716
IC21 IS 74LS374
IC22 IS 74LS244
IC23 IS LM358
IC24 IS 7805
Q1,2 ARE BC184L
D1-3 ARE 1N4148
ZD1 IS 6V8 400mV ZENER

## HOW IT WORKS

The CPU (IC15) and the video display share the same memory. Access to the memory is switched at the processor's Ø2 clock rate (as on the Microtan 65), using the address selector IC12-14. This alternately connects the CPU and CRT address onto the RAM chips IC17, IC18. IC16a is a R/$\overline{W}$ synchroniser to ensure the RAM chips get proper set-up and hold times on the R/$\overline{W}$ line after it passes through the address selector IC12.

The master clock oscillator is built around three of the inverters in IC1; XTAL1 sets the operating frequency at 6 MHz. The clock signal enters the counter chain down the left of the circuit diagram at pin 1 of IC2a. IC3 generates all the character addresses along the video line for CRT refresh, with IC4a and IC4b/IC5a generating two of the timing signals required by the character generator IC19. The count length of IC3 is determined by the AND gate IC6a. IC7a generates the line sync pulses while IC5b/IC6b produce the line blanking. The line sync pulses also clock counter IC9, which together with IC2b generates all the line addresses. IC2b, IC7b and the associated logic controls the count length of the line counter and produces the frame sync. IC2b also produces the frame blanking directly.

IC19 is the character generator which receives data from the memory chips IC17, 18 during the time interval that the CRT refresh address is switched through. Data is latched and the character pattern is clocked out under the control of the various input signals to this chip. The character data is mixed with the line and frame blanking in IC7c, while the line sync and frame sync pulses are mixed in the AND gate IC6c to give a composite sync. The output of IC7c is mixed with the composite sync in the diode circuit (D1, D2 and associated resistors) to give an analogue video signal which is fed to the UHF modulator.

The modulator requires a supply of about 6V5. To avoid the need for a separate PSU, this is derived using the DC-DC converter based on L1 and Q1, which is driven via R6 from the CRT counter chain.

IC20 is the 2K EPROM containing the monitor program. For the EPROM to be enabled, A11 must be high and R/$\overline{W}$ in the read mode; these lines drive NAND gate IC10c which provides the chip select signal to pins 18 and 20 of the EPROM. Gates IC8c, IC6d, IC10d, IC8e and IC11b,c enable the I/O port, controlling the hex keypad and the cassette interface. Any location between 400 and 7FF (Hex) will address the port provided the I/O control line has not been pulled low externally. When reading, IC22 is enabled; when writing, IC21 is enabled. IC23 and associated components form the input amplifer for the cassette interface.

The only expansion the Microtutor has is for I/O experiments; the address bus, data bus, R/$\overline{W}$, Ø2 and I/O control lines all come out to pads on the PCB. If the user builds an external I/O circuit then that circuit must operate the I/O control, which is active low, to disable the keypad circuits, otherwise bus contention will result. The monitor program assumes that they keypad is located at 7FF. Any location in the I/O address space will activate the keypad port if I/O control is not pulled low, however. For further I/O experiments, all eight bits from IC21, 22 are brought out to pads. This means that users can experiment with bigger keypads or utilise the unused bits. $\overline{RST}$ and $\overline{IRQ}$ also come out to pads on the PCB.

<TERM> is one of the terminating characters <CR>, <SP>, <LF> or <ESC>. In fact TANBUG accepts any of the "control" characters (HEX code ϕ-2ϕ) as terminator. TANBUG will reply with a ? if an illegal command is encountered.

## Starting The Monitor

Press the reset button on the Microtutor. TANBUG will scroll the display and respond with

TANBUG
■

Note that on initial power-up the top part of the display will be filled with spurious characters. These will disappear as new commands are entered and the display scolls up. On subsequent resets the previous operations remain displayed to facilitate debugging.

## Memory modify/examine command M

The M command allows the user to enter and modify programs by changing the RAM locations to the desired values. The command also allows the user to inspect ROM locations, modify registers and so on. To open a location type the following

M <ADDR> <TERM>

TANBUG then replies with the current contents of that location. For example, to examine the contents of RAM location 1ϕϕ type M1ϕϕ<CR> TANBUG then responds on the display with

**M1ϕϕ**,ϕE,■

assuming the current contents of the location were ϕE.

There are now several options open to the user. If any terminator is typed the location is closed and not altered and the cursor moves to the next line scrolling up the display by one row. If however, a value is typed followed by one of the terminators <CR>, <LF> or <ESC> the location is modified and then closed. For example, using <CR>

**M1ϕϕ**,ϕE,**FF**
■

location 1ϕϕ will now contain FF. If however <SP> is typed, the location is re-opened and unmodified.

**M1ϕϕ**,ϕE,**FF**
Mϕ1ϕϕ,ϕE,■

This facility is useful if an erroneous value has been typed. The terminators <LF> and <ESC> modify the current location being examined, then open the next and previous locations respectively ie using <LF>

**M1ϕϕ**,ϕE,**FF**
Mϕ1ϕ1,AB,■

and using <ESC>

**M1ϕϕ**,ϕE,
MϕϕFF,56,■

Using <LF> makes for very easy program entry, it only being necessary to type the initial address of the program followed by its data and <LF>, then responding to the cursor prompt for subsequent data words.

Note that locations 1FE and 1FF should not be modified. These are the stack locations which contain the monitor return addresses. If they are corrupted TANBUG will almost certainly "crash" and it will be necessary to issue a reset in order to recover.

## List command L

The list command allows the user to list out sections of memory onto the display. It is possible to display the contents of a maximum of 120 consecutive memory locations simultaneously. To list a series of locations type

L <ADDR>, <NUMBER>
<TERM>

where <ADDR> is the address of the first location to be printed and <NUMBER> is the number of lines of eight consecutive locations to be printed. TANBUG pauses briefly between each line to allow the user to scan them. For example, to list the first 16 locations of TANBUG (which resides at Cϕϕ-FFF) type

LC00,2 <CR>.

The display will then be

**LCϕϕ,2**
Cϕϕ A2 FF 9A E8 86 17 20 B7
Cϕ8 FF 8D F3 BF A2 0E BD DF
■

If zero lines are requested (ie <NUMBER> = ϕ) then 256 lines will be given.

## Go command G

Having entered a program using the M command and verified it using the L command, the user can use the G command to start running his own program. The command is of the format G <ADDR> <TERM>. For example, to start a program whose first instruction is at location 1ϕϕ type G1ϕϕ<CR>. When the user program is started the cursor disappears. On a return to the monitor it re-appears.

The G command automatically sets up two of the microprocessor's internal registers:

a) The program counter (PC) is set to the start address given in the G command.
b) The stack pointer (SP) is set to location 1FF.

The contents of the other four internal registers, namely the status word (PSW), index X (IX), index Y (IY) and accumulator (A), are taken from the monitor pseudo registers (described next). Thus the user can either set up the pseudo registers before typing the G command, or use instructions within his/her program to manipulate them directly.

## Register modify/examine command R

Locations 15 to 1B within the RAM reserved for TANBUG are the user psuedo registers. The user can set these locations prior to issuing a G command; the values are then transferred to the microprocessor's internal registers immediately before the user program is started. The pseudo register locations are also used by the monitor to save the user internal register values when a breakpoint is encountered. These values are then transferred back into the microprocessor when a P command is issued, so that to all intents and purposes the user program appears to be uninterrupted.

The R command allows the user to modify these registers in conjunction with the M command. To modify/examine registers, type R <CR> and the following display will apppear (location 15 containing ϕϕ say).

**TABLE 2**

| | | |
|---|---|---|
| 15 | Low order byte of program counter | (PCL) |
| 16 | High order byte of program counter | (PCH) |
| 17 | Processor status word | (PSW) |
| 18 | Stack pointer | (SP) |
| 19 | Index X | (IX) |
| 1A | Index Y | (IY) |
| 1B | Accumulator | (A) |

**R**
Mϕϕ15,ϕϕ,■

Now proceed as for the M command.

Naturally the M command could be used to modify/examine location 15 without using the R command — the R command merely saves the user the need to remember and type in the start location of the pseudo registers. Psuedo register locations are shown in Table 2. Two typical instances of the use of the R command are:—

a) Setting up PSW, IX, IY and A before starting a user program.
b) Modifying registers after a breakpoint but before proceeding with program execution (using the P command) for debugging purposes.

Note that when modifying registers in case (b) care must be taken if PCL, PCH or SP are modified, since the proceed command P uses these to determine the address of the next instructions to be executed (PCL, PCH) and the user stack pointer (SP).

# MICROTUTOR PART 2

## This month we look at the remaining commands provided by the TANBUG monitor on our machine code trainer for easy programming. Design and development by Tangerine Computer Systems.

Having shown how to enter and run machine code programs last month, we now consider the commands which aid in debugging and program writing, and save/load operations using the cassette port.

### Single Instruction Mode S

Single instruction mode is a very powerful debugging aid. When set, TANBUG executes the user program one instruction at a time, re-entering the monitor between each instruction and printing out the status of all the microprocessor's internal registers as they were after the last instruction executed in the user program. The S command is used in conjunction with the proceed command P and the normal mode command N.

### Normal Mode Command N

The N command is the complement of the S command and is used to cancel the S command so that the microprocessor executes the user program in the normal manner without returning to the monitor between each instruction. RESET automatically sets the normal mode of operation.

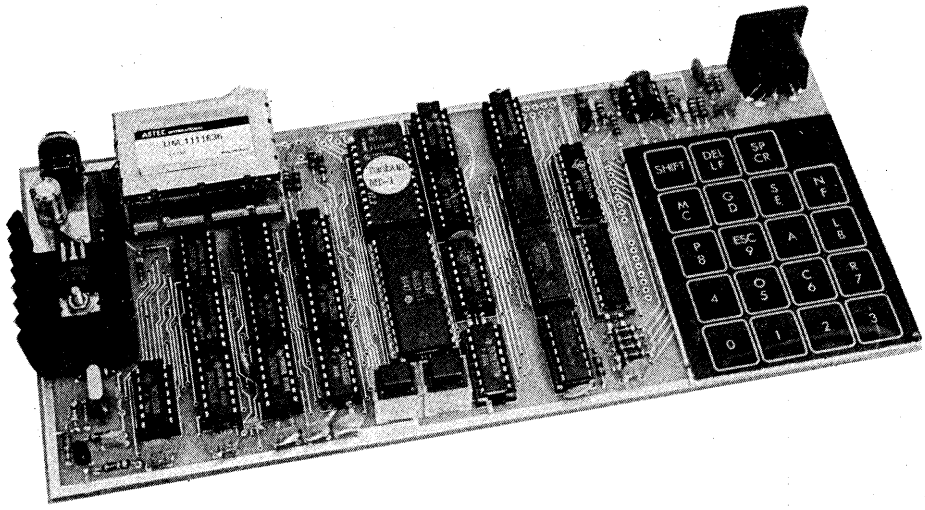
### Proceed Command P

The P command is used to instruct TANBUG to execute the next instruction in the user program when in single instruction mode. Pseudo register contents are transferred into the microprocessor's internal registers and the next instruction in the user's program is executed with the monitor. P may also be used with an argument, thus: P <NUMBER> <CR> where NUMBER is less than or equal to FF. In this case the program executes the specified number of instructions before returning to the monitor.

Each time the monitor is re-entered after execution of an instruction or instructions, the status of the microprocessor internal registers as they were in the user program are printed across the screen in the following order:

- Address of next instruction to be executed.
- Processor status word.
- Stack pointer.
- Index register X.
- Index register Y.
- Accumulator.

Note that these are in the same order as the pseudo registers described last month.



Whenever the user program is entered, the cursor is removed from the display. Whenever the monitor is entered, the cursor returns to the display as a user prompt. While in the monitor between user instructions, any monitor command can be typed. A program must always be started by the G command, then P used if in single instruction mode. A P command used before a G command is issued is likely to cause a program 'crash' and should not be attempted.

As an example consider the simple program which repeatedly adds 1 to the accumulator.

| Address | Data | Mnemonic | Comment |
|---|---|---|---|
| 100 | 69 | ADC 1 | : add 1 to acc. |
| 101 | 01 | | |
| 102 | 4C | JMP 100 | |
| 103 | 00 | | |
| 104 | 01 | | |

Set the single instruction mode and start the program. (The user may wish to initially set the accumulator to 00 by using the M command).

```
S
G100
0102      20      FF      00      01
```

TANBUG then responds with the characters shown above.

- 0102 is the address of the next instruction to be executed.
- 20 is the processor status word value.
- FF is the low byte value of the stack pointer. The high byte is always set to 1, the stack is therefore pointing at location 1FF.
- 00 is the value of the index X register.
- 00 is the value of the index Y register.
- 01 is the value of the accumulator. It is a 1 as 1 has been added to the accumulator and it is assumed that the user cleared the accumulator before starting the program.

Since the cursor has re-appeared, TANBUG is ready for any monitor command. For example, registers or memory locations can be modified, or the program may be re-started from scratch by typing G100<CR> again. If the user wishes to continue

```
TANBUG
LA3,9
00A3 A0 0F 20 73 FE 88 10 FA
00AB A9 E0 85 A0 A9 02 85 A1
00B3 A0 0F A9 20 8D E0 03 D0
00BB 3B 8A 48 98 48 20 FA FD
00C3 68 A8 68 AA 91 A0 A5 01
00CB C9 4C F0 4F C9 39 F0 13
00D3 C9 31 F0 29 C9 34 F0 38
00DB C9 36 F0 37 A9 4F 91 A0
00E3 AA D0 D6 A5 A0 38 E9 20
?
```

**Fig. 1 Screen display after listing the first 72 bytes of a program which starts at address 00A3.**

then type P<CR>. The resulting display is

```
S
G100
0102   20  FF  00  00  01
P
0100   20  FF  00  00  01
■
```

Since the instruction at location 102 was 'Jump to 100', the status printout shows that this has indeed occurred. Registers, since they were not modified by any program instruction, remain unchanged. To proceed further type P<CR> again:

```
S
G100
0102   20  FF  00  00  01
P
0100   20  FF  00  00  01
P
0102   20  FF  00  00  02
■
```

The add instruction has been executed again, so the accumulator has incremented by 1 to become 2. Now typing P4<CR> gives this display:

```
S
G100
0102   20  FF  00  00  01
P
0100   20  FF  00  00  01
P
0102   20  FF  00  00  02
P4
0102   20  FF  00  00  04
■
```

TANBUG allowed execution of four instructions before again returning to the monitor. The four instructions were two add instructions and two jump instructions thus giving the accumulator the value 4.

Typing N<CR> then P<CR> removes the single instruction mode and causes the program to proceed. It now does not return to the monitor but continues to race around this small program loop continually adding and jumping back. To re-enter the monitor, either press the reset button or the interrupt button. The interrupt will cause the processor to break into the program and enter the monitor with the display as described above.

It can be seen that the S and P commands are particularly useful when tracing a program which contains instructions that transfer program control eg jumps, branches and subroutines, since these commands allow the user to interrogate the order of execution of his/her program.

## Breakpoint Command B

A breakpoint is a complementary debugging aid to single instruction mode. Instead of stepping singly through all instructions in a program, the breakpoint facility allows the user to specify the address at which he requires the monitor to be re-entered from his/her program. As an example, consider a long program in which a fault is suspected to exist near the end. It would be very tedious and time consuming to single step through the program to the problem area. A breakpoint can be set just previous to where the fault is suspected to exist and the program started with the G command. Normal execution occurs until the breakpoint is reached, then the monitor is re-entered with the same status print-out as for single instruction mode. Any monitor commands can then be used and the program continued.

The format of the breakpoint command is

B <ADDR> <NUMBER> <CR>

where <ADDR> is the address of any instruction op code (but not argument), <NUMBER> is any number from 0–7 defining one of eight breakpoints. B<CR> removes all breakpoints. As an example consider the following program:

```
100  E8          LOOP: INX
101  C8                INY
102  69  01            ADC #1
104  4C  00  01        JMP LOOP
```

Firstly set index X, index Y and the accumulator to 00 using the R command. To set breakpoint 0 at the jump instruction and start the program type B104,0<CR>G100 <CR> . The display will then be

```
B104,0
G100
0104   20  FF  01  01  01
■
```

The jump instruction was reached and the breakpoint re-directed control back to TANBUG. If it was required, single instruction mode could be set for further debugging. However, assume that we wish to execute the loop again by typing P<CR>.

```
B104,0
G100
0102   20  FF  01  01  01
P
0104   20  FF  02  02  02
■
```

The proceed command P has gone once through the breakpoint and then re-entered the monitor. If P<NUMBER> <CR> was typed it would have proceeded through the breakpoint <NUMBER> times.

Up to eight breakpoints can be set at eight different locations. The B<CR> command removes all

```
TANBUG
BA3,1
GA3
00A3  30  FF  22  03  03
?
```

**Fig. 2 Setting the first breakpoint at the start address and running the program gives this register printout.**

breakpoints. A single breakpoint can be removed by setting its address to 0. For example, imagine a breakpoint is set as follows; B102,2, and it is subsequently wished to remove it but leave any others unaltered; type B0,2<CR> to remove it.

**Caution:** The breakpoint system works by replacing the user's instruction with a special instruction (BRK) whose op code is 00. Replacement is carried out when G or P is typed. On return to the monitor the original op code is replaced. It is therefore possible to corrupt the user program under some circumstances. The following points should therefore be observed:

- Breakpoints must only be set at the op code part of a user instruction and nowhere else.
- If the user program utilises the BRK instruction as part of the user code, then this will cause a break in the normal manner, but note that the BRK instruction will be interpreted by the monitor as a 1-byte no operation.
- If breakpoints are set in the user program and a reset is issued while the microprocessor is executing the user program rather than the monitor, the breakpoints are lost and those locations at which breakpoints were set in the user program will be corrupted. These locations must be re-entered using the M command before restarting the user program.
- Setting more than one breakpoint at the same address causes the user program to be corrupted.
- To use breakpoints, the user must not have modified the interrupt link, ie the interrupt code within TANBUG must be executed.

The status of breakpoints may be inspected by using the M command to examine the breakpoint status table. This is located in RAM locations 20-2F and are as in Table 1.

**TABLE 1**

| Address | Contents |
|---|---|
| 20 | PCL B0 |
| 21 | PCH B0 |
| 22 | PCL B1 |
| 23 | PCH B1 |
| 24 | PCL B2 |
| 25 | PCH B2 |
| 26 | PCL B3 |
| 27 | PCH B3 |
| 28 | PCL B4 |
| 29 | PCH B4 |
| 2A | PCL B5 |
| 2B | PCH B5 |
| 2C | PCL B6 |
| 2D | PCH B6 |
| 2E | PCL B7 |
| 2F | PCH B7 |

For example, typing M20<CR> followed by <LF> gives

**M20,**00,
M0021,01,■

This indicates that breakpoint 0 is set to location 100 by taking the contents of location 20 as PCL and of location 21 as PCH. If the breakpoint is set at location 0 then this particular breakpoint is disabled.

## Offset Command O

The offset command O is a program writing aid. It calculates branch offsets for the user for incorporation as arguments in branch instructions. Consider the example:

```
100  E8   LOOP: INX
101  C8       INY
102  69       ADC#1
103  01         .
                .
                .
120  D0       BNE LOOP
121           (branch argument)
```

To calculate the number to enter into location 121 is quite tedious without a facility such as the O command. It is used with the following format.

O<ADDR OF BRANCH OP CODE>
<ADDR OF DEST><CR>

and in this case it would be necessary to type 0120,100<CR>. The display would be

**0120,100** = F0
■

F0 is the number that should be entered into location 121 such that if the BNE instruction is true the program counter will jump to the label LOOP.

Note that the maximum branch range is 7F forwards and backwards. If the range is exceeded a ? is displayed.

## Copy Command C

The copy command allows copying of the contents of one block of memory to another. Its format is

C<START ADDR SOURCE>
<END ADDR SOURCE>
<START ADDR DEST> <CR>

Suppose it is required to copy the block of data in locations C00–D00 into a block starting at location 200. This may be achieved by typing CC00,D00,200<CR>. The display will be

**CC00,D00,200**
■

As 200 is the starting address of the display memory the user will notice that the top half of the screen has been overwritten with all sorts of weird and wonderful characters. What this example has done is to take the first 256 bytes of TANBUG and copy them into the top half of the display. The display then scrolled leaving the top seven rows filled with these characters.

## The Interrupt Button

Depression of the interrupt button will cause a re-entry into the monitor from the user program. For example, if the trivial program

```
100  69        LOOP: ADC#1
101  01
102  4C              JMP LOOP
103  00
104  01
```

has been started by typing the G command the program continues to loop around continuously with no exit path to the monitor, except by issuing a reset. Instead of a reset the user can press the interrupt button, TANBUG responding thus

0100  20  FF  01  01  01
■

Using the interrupt button has caused a breakpoint to be executed and the monitor invoked. The register print-out above is only typical, the value of each being that when interrupt was depressed. Any monitor command may now be typed, for example P causes the user program to proceed once again.

The interrupt facility is most useful in debugging where the user program gets into an unforseen loop where breakpoints have not been set. It enables the user to rejoin the monitor without using reset and losing the breakpoints that have been set. Notes:

- Interrupts must be enabled for the interrupt facility to operate. TANBUG enables interrupts when entering a user program, so don't disable interrupts if the interrupt facility is required.
- The user must not have modified the interrupt jump link. TANBUG's interrupt code must be executed.

# MICROTUTOR PART 3

## It's all very well being able to write machine code programs but unless you can store them on tape things are going to get a bit laborious. This time we look at the cassette firmware. Design by Tangerine Computer Systems.

The TANBUG software allows files to be dumped, verified with the memory contents, and read back into memory. Files may be named with a filename of up to six characters. Files are recorded at 300 baud in CUTS format and several error checks are incorporated. Filenames are compatible with the Tangerine Microtan system (only when used with hex keypad). Some experimentation may be required to find the correct recording and playback levels. Experience shows that on most automatic level recorders, the playback volume control can be left at maximum setting.

### Dumping To Tape

The 'D' command is used to dump an area of memory to tape. Its format is.

D<START ADDR>,
<END ADDR>,
<FILENAME><CR>

The filename may be up to six characters long. Characters within it may be A–F, 0–9. To dump a program on to tape proceed as follows:

- Use the 'D' command but do not type <CR> yet.
- Start tape running in record mode.
- Hit <CR>. The VDU will respond with the filename being dumped, with an added appendix of .A to distinguish this file as being an absolute file.
- The cursor will disappear and the file will be dumped.
- The cursor will reappear when the dump is complete — the program returns to TANBUG.
- Stop the cassette.

Example: Type D400,410, F1, <CR>. The display will appear as follows:

**D400,410, F1**
F1 .A
■

The instruction dumped locations 400 to 410 inclusive, and called the file F1.

A question mark error will be generated if the command format is illegal, if the filename contains an illegal character, or if it is more than six characters long.

### Examining A Tape

The 'E' command allows you to examine a tape to see that the file has been dumped correctly, and that it can be read back. This command searches the tape for the named file, then compares what it reads with the memory content. To examine a tape:

- Position the cassette on a piece of blank tape (ie a section with no recorded signal) somewhere before the file to be examined.
- Type E, <FILENAME><CR>. The VDU responds with the filename.
- Start the tape in play mode.
- When a file is encountered, the VDU will respond with the filename and dump start address.
- If this filename is not identical with the one specified, the previous step is repeated.
- If a read error is encountered, the message F(n) is printed, indicating a filename error. The program then goes to the fourth step.
- If the filename is identical, then the comparison will be initiated.
- If the comparison is correct, the program will return a cursor prompt and return to TANBUG.
- If the comparison is incorrect three types of error may occur.

M(n) — memory error; contents of tape do not agree with contents of memory. (n) is the address of the fault.

P(n) — a parity error occurred when reading the data associated with location (n).

C(n) — a checksum error occurred at the end of the file; (n) indicates the file end address.

In all cases, after printing an error, the program continues to read data. Thus if only a few errors occur, they may be checked out by reference to their addresses.

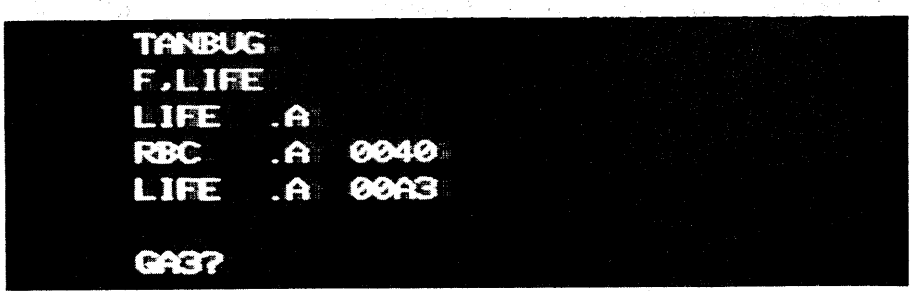
Note that a VDU scroll may induce a parity error due to the time taken for a scroll. It does mean, however, that there are more errors than acceptable and the file should be re-dumped until error-free. As an example suppose there are two files on a tape, F1 and F2. You wish to examine F2, but position the tape at the blank leader.

**E, F2**

```
F2    .A          ;prints your
                  filename.
F1    .A  0A00    ;prints first
                  filename.
F2    .A  0400    ;prints second
                  filename.
■                 ;comparison
                  correct.
```

### BUYLINES

**A complete kit of parts for the Microtutor is available from Microtanic Computer Systems Limited, 16 Upland Road, Dulwich, London SE22, at a cost of £49.95. For the lazy or unadventurous, there is a built version for £59.95. Both these prices include VAT, but package and postage is extra at 60p.**




**A typical display when using the cassette firmware. LIFE is loaded and ready to run.**

Or, with one memory error:

```
E, F2
F2      .A
F1      .A  0A00
F2      .A  0400
M0410
■
```

Location 0410 is at fault.

### Tape Directory

The 'E' command may be used to obtain a complete listing of the tape contents, by rewinding the tape to the start and looking for a non-existent filename. Note that in this case it is necessary to exit back to TANBUG at the tape end by using the RESET key since interrupts are disabled in the cassette software. This procedure may also be necessary if the examine procedure gets very badly out of step due to a large number of errors.

### Fetching A File

To fetch (load) a file into memory, the 'F' command is used as follows:

F,<FILENAME> <CR>

For example, typing F,F1<CR> looks for F1 .A and loads it into memory. Operating procedures and errors are exactly as detailed in the section on the examine command. Should an M error occur, a hardware fault is indicated because the program loads the input data to the memory location and checks it immediately afterwards.

If one program uses several different areas of memory (eg one area for subroutines and another for main code) it is necessary either to dump the whole area in one file, thus encompassing some redundant locations, or to dump in two files.

### Error Messages

The following is a resumé of the error messages that may occur during tape operations.

M(n) — when examining, memory location (n) contained a different value to that read from the tape.

— when fetching, memory location (n) failed to be updated with the value read in (hardware error).

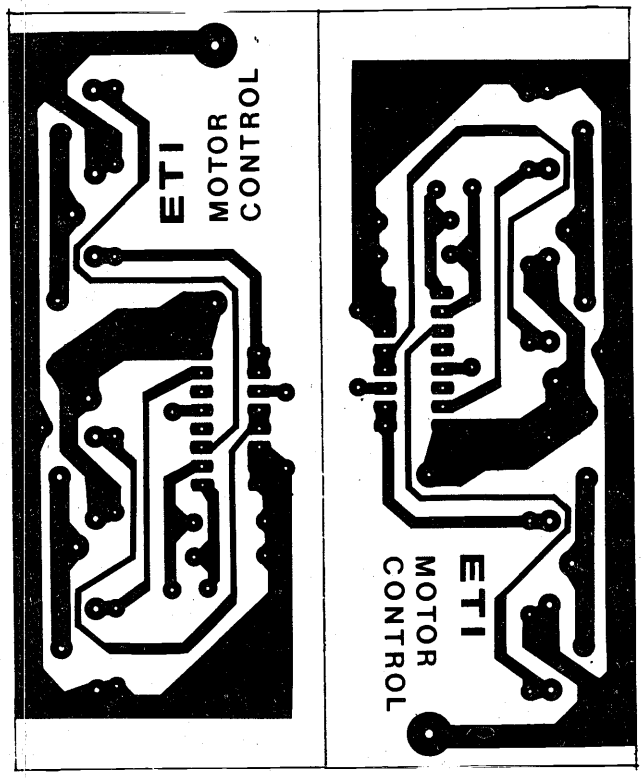
P(n) — a parity error occurred when reading the byte for location (n).

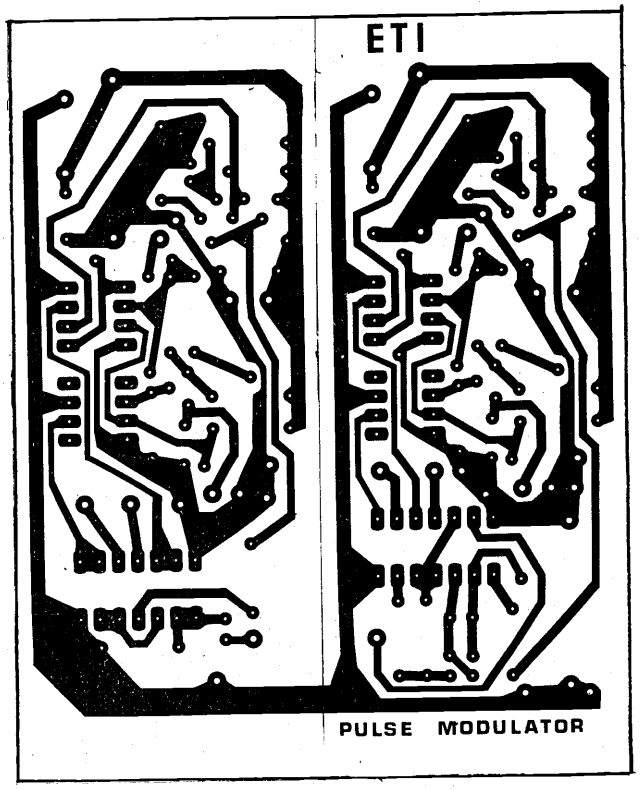
C(n) — a checksum error occurred during the tape read; (n) indicates the end of file address. Since a parity check is not infallible and will not detect two bits in error, a checksum is an additional data validity test. A checksum error will nearly always occur if a parity error occurs. If a checksum error occurs, but no parity error, the code must be listed and visually checked to determine where the error occurs.

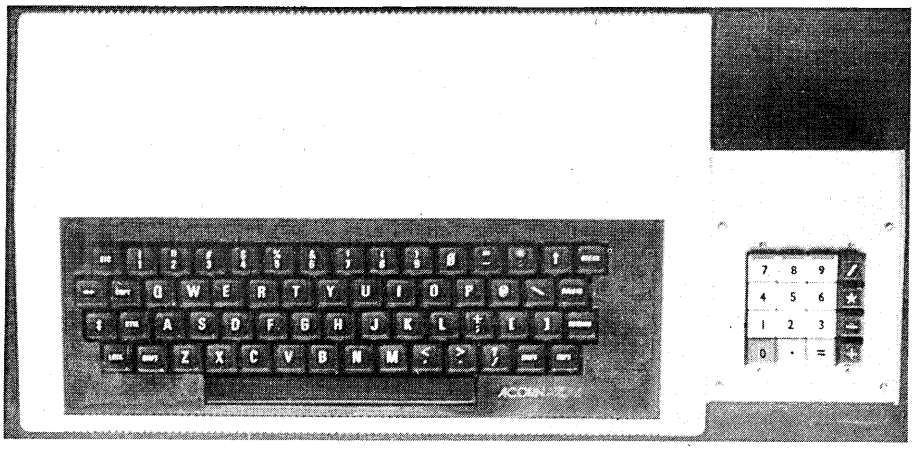
F(n) — an error occurred when reading the filename: (n) is meaningless.

If errors do occur, you should first try reading the tape again in case the error was due to mains-borne noise. If, however, the same error persistently occurs on re-tries, the tape is likely to be in error.

## PCB Foil Patterns





# ATOM CALCULATOR PAD

## And now for some Atomic fusion. Nothing to do with nuclear reactors (that's still at the breadboarding stage): this is an add-on numeric keypad. Design by Patrick Squire.

The addition of a numeric pad to the standard QWERTY keyboard greatly increases the ease of use of a microcomputer, both for programs containing extensive numerical statements and for joystick operations, where the layout of the numeric keys corresponds directly to the direction of movement of an object being controlled on the VDU. Anyone accustomed to using machines both with and without numeric pads will appreciate the advantages of the extra keys. An additional drawback of the QWERTY keyboard is that the +, * and = operations are obtained by shifted keys. This can be very frustrating and lead to errors when entering programs.

For all these reasons it is desirable to be able to add a basic calculator pad, containing the numbers 0-9, the decimal point, equals sign and the four arithmetic operators, to machines not originally supplied with such a facility. Unfortunately such an accessory is not generally available for most micros. This article describes a calculator pad designed specifically for the Acorn Atom, although the general principles employed are capable of straightforward modification to other machines which use a similar system of scanning the key matrix. The whole unit can be constructed for under £20, but it does involve soldering to the masterboard of the Atom, so construction should not be undertaken by the fainthearted.

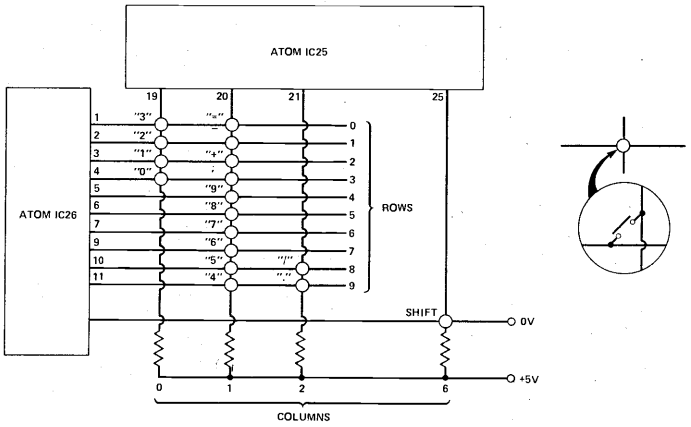



**Fig. 1 Part of the Atom keyboard matrix. Each row is connected to one or more columns by a single keyboard switch. The rows are driven by the 3-to-10 decoder IC26. The columns are read by the PIA IC25.**

### Matrix Scanning

In order to understand the operation of the unit described here, or in order to modify it for other machines or to incorporate alternative functions, it is essential to understand how the Atom reads a depressed key. Figure 1 shows the appropriate section of the Atom circuit. It can be seen that each character key connects a particular row (numbered 0 to 9) with a particular column (numbered 0 to 6). For example, key '9' connects row 4 with column 1. The SHIFT key works in a slightly different way, so I will first describe the operation of reading an unshifted key.

The Atom operating system includes a machine code routine that scans the keyboard. Normally all the rows 0-9 are held at logic high. The columns are connected by pull-up resistors to the + 5 V rail, so they are also logic high. In this state the depression of a character key has no effect, since it merely connects two lines both in the same state. When the scan commences, each row is successively driven low by IC26. The routine then interrogates each column to see if it is low. Consider the state when row 4 has been set low. If no key has been pressed the columns will all be high. If key 9 is pressed, however, column 1 will be connected to row 4, which is low. Column 1 will thus be pulled low and the Atom will know from the

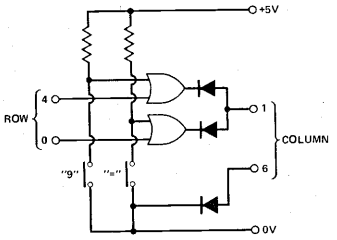



**Fig. 2 The principle of the keyboard extension. The unshifted key '9' drives column 1 low when row 4 is also low. The shifted key '=' drives column 1 low when row 1 is low; it also pulls column 6 low to give the shift operation.**

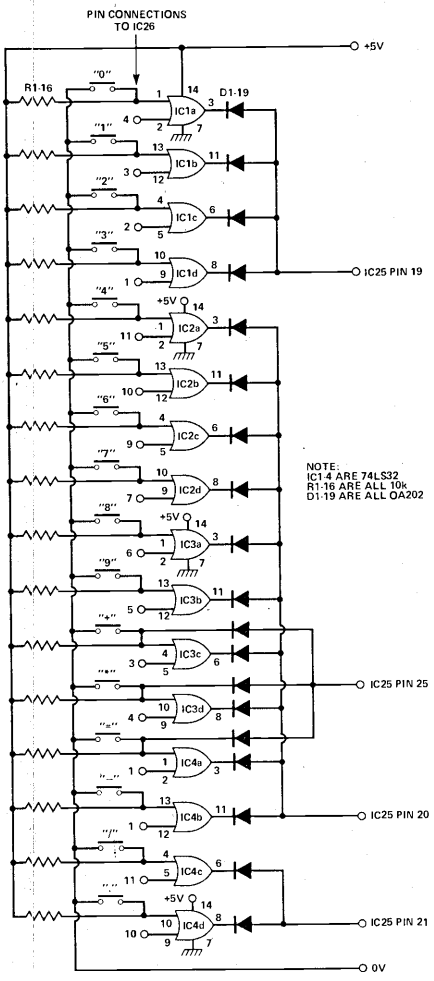



**Fig. 3 The complete circuit of the keypad extension.**

## BUYLINES

Nothing at all unusual in this project and most suppliers should have the components in stock. One source for the case we used is BI-PAK Semiconductors, PO Box 6, 63a High Street, Ware, Herts SG12 9AG (telephone 0920 3182/3442): they also stock the semiconductors and resistors. The order number for the case is 148. The PCB can be ordered using our PCB Service order form.

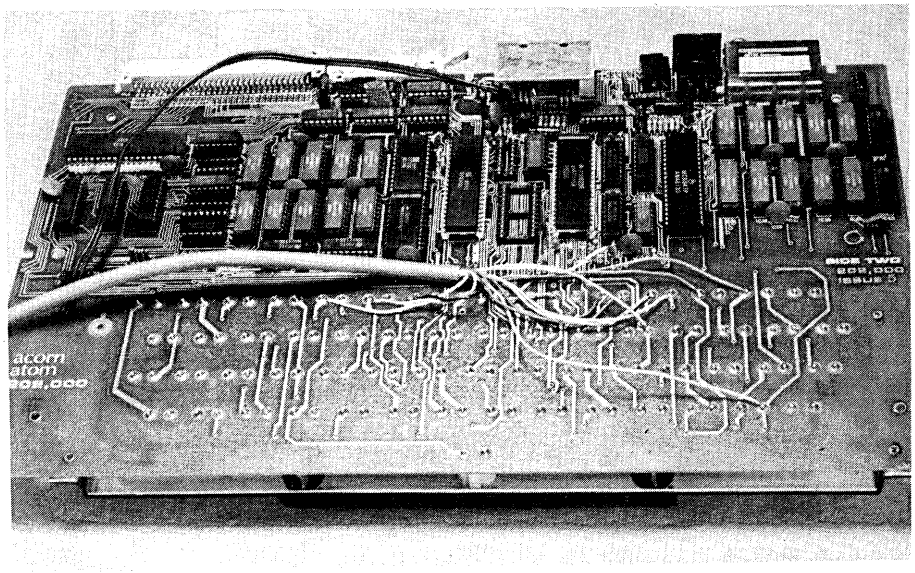
information stored in ROM that the combination 'row 4 low and column 1 low' means that key '9' has been pressed. A similar process works for all the unshifted keys.

For the shifted keys use is made of column 6. Figure 1 shows that this is normally high, but that if the SHIFT key is pressed it pulls the line low by direct connection to the 0 V line. The key scan routine therefore also interrogates column 6 and, according to its state, decides whether a dual-role key, such as '=/—' is to be interpreted as shifted or unshifted.

## Circuit Principles

The direct way of extending the ATOM keyboard would be to parallel each keyboard switch by a corresponding key on the pad. However, this would involve having 16 unconnected switches and 32 connecting wires. This would be a very inefficient system and would not be able to cope with shifted characters as single-key operations. The system employed in this design employs an OR gate for each key, and is illustrated in Fig. 2 for two keys, one the unshifted '9', the other the shifted '='.

Consider first the unshifted '9' key. When the Atom drives pin 5 of IC26 low (this corresponds to row 4 — see Fig. 1) one input of the upper OR gate is driven low. So long as key '9' remains open, however, the other input of this gate remains high. The OR action then maintains the output high, so pin 20 of IC25 (column 1) remains high. If, on the other hand, key '9' is closed both



**Wiring up to the Atom — the diagram below refers.**

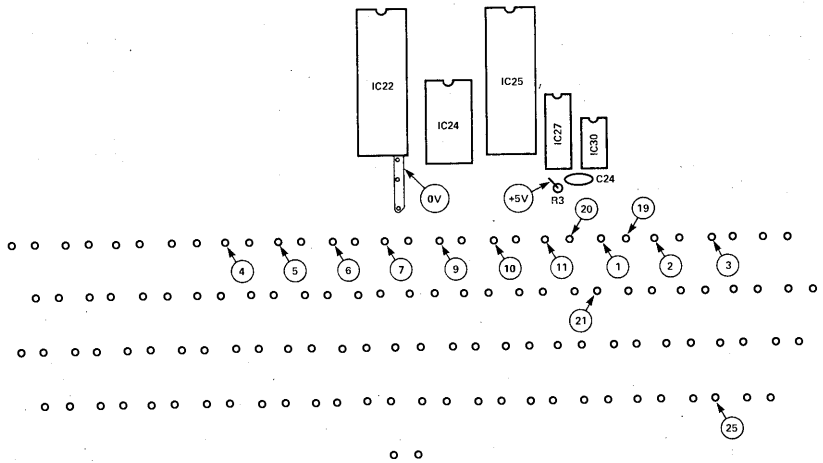



**Fig. 4 A diagram of the Atom PCB (rearside of the keyboard) showing take-off points for the wiring. The numbers refer to the pins on IC25 and IC26 (as Fig. 1 shows, this numbering is unambiguous), and correspond to the numbers on the PCB.**

## PARTS LIST

**Resistors (all ¼W, 5%)**
**R1-16 10k**

**Semiconductors**
**IC1-4 74LS32**
**D1-19 OA202**

**Miscellaneous**
**PCB (see Buylines); 16-switch keypad matrix (see text); ½ meter 15-core plus screen cable; case to suit.**

inputs to the OR gate are low, and the output is low. The fourth possibility (pin 5 of IC26 high, '9' low) also results in a high output, so pressing '9' will not have any effect unless row 4 is also low, as required. The purpose of the diodes is to isolate the outputs of the various OR gates. In the absence of the diodes, when one gate tried to go low it would be pulled up by all the other gates connected to the same line. It would also upset the operation of IC26. The diodes have a similar effect to open-collector outputs. However, open-collector OR gates are not generally available.

Now consider the operation of the shifted '=' key. The logic of the lower OR gate is identical to that just described, but in addition a diode connection to pin 25 of IC25 pulls column 6 low, thus simulating the effect of the combined operation of the SHIFT and '=/—' keys on the Atom keyboard. The complete circuit is shown in Fig. 3. Note that for 16 keys, the 16 OR gates conveniently occupy four quad OR gate ICs (type 74LS32).

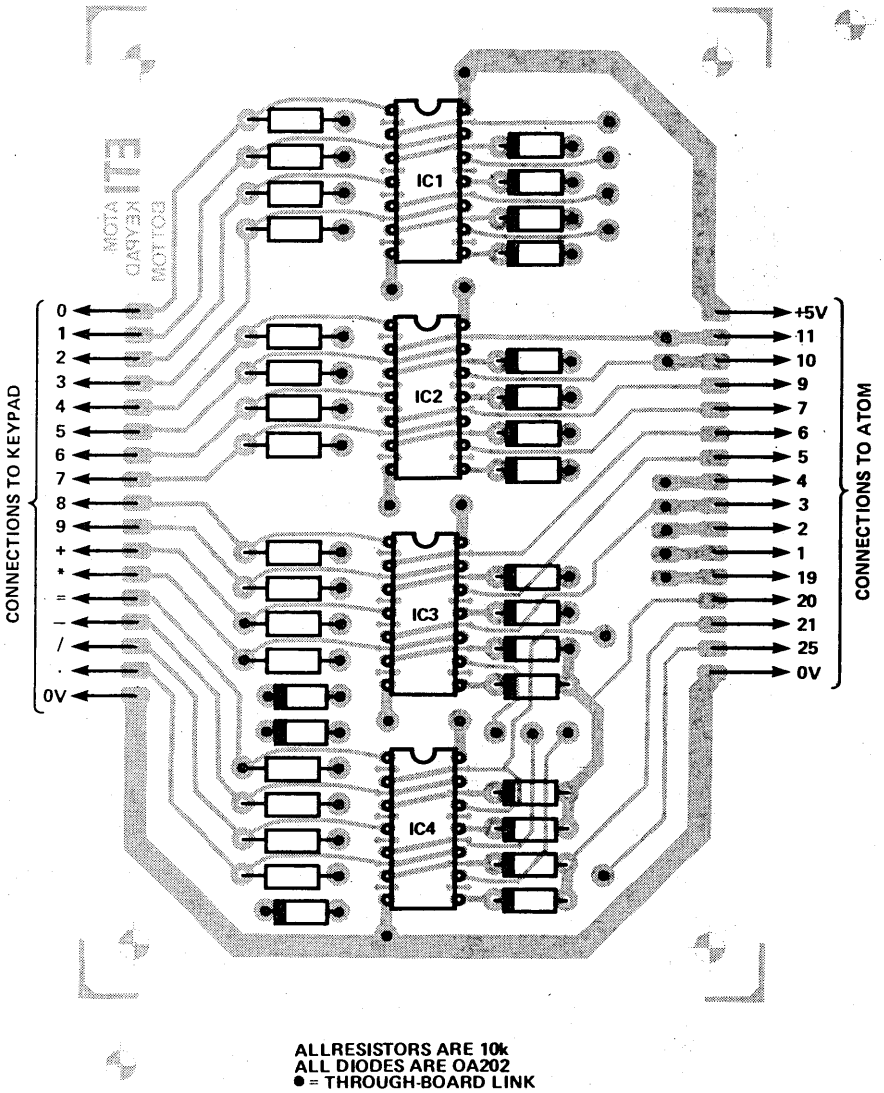



**Fig. 5 Component overlay for the project. Remember to solder both sides of the board at each point indicated by a dot.**

## Construction

The most expensive item is the keyboard itself. It should contain 16 single-poll switches with one side commoned to the 0 V rail. The author used a very cheap but entirely satsifactory unit from Watford Electronics Ltd. Perusal of the advertising columns should reveal a source of a suitable alternative for less than £10. Avoid decoded types, which are connected in a matrix fashion unsuitable for this application. As a last resort you could make your own from the readily available keypad switches.

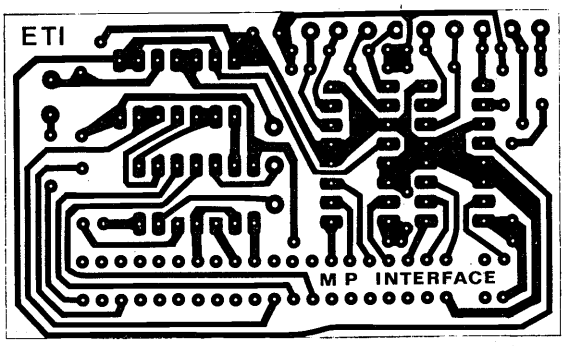
The circuit should be constructed on a printed circuit board. Veroboard can be used, but is much more difficult to wire up and looks messy with so many connections. We've designed a double-sided PCB for this project — with care this shouldn't cause grave difficulties to home constructors, and finished boards will be on sale through our PCB Service. To avoid the cost of plated-through holes, links between the two sides of the PCB are required: these are shown by dots on the overlay. If the position has a component lead, solder it on both sides of the board — otherwise fit a through-board link.

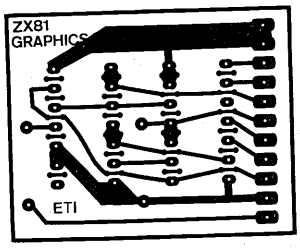
Now comes the tricky bit. The problem is to find suitable places on the Atom masterboard to pick up connections to the key matrix. After an hour or so of probing with a multimeter we came up with the connection diagram of Fig. 5. We don't need to distinguish between the sets of pins for IC25 and IC26, incidentally, as the two have no numbers in common.

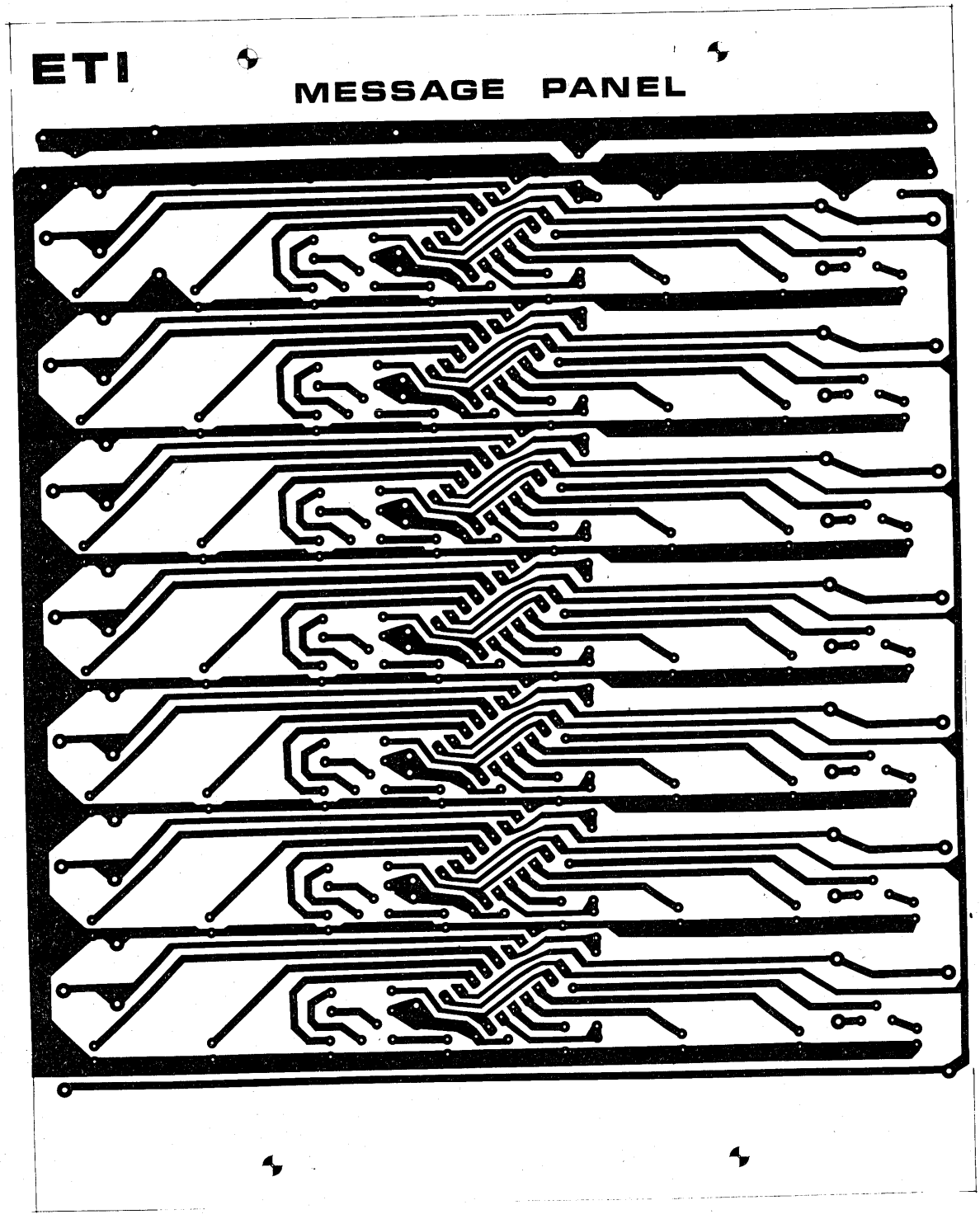
All the keyboard matrix connections can be made to the pins of the existing keyboard switches. The supply connections are somewhat trickier. The 0 V is taken from a large area near IC22 (it will be necessary to scrape away the green varnish in order to solder the lead), while the +5 V is taken from the top end of R3. The photograph will hopefully make things clear.

The number of connections required between the Atom and the PCB is 16, including the 0 V and +5 V rails. These are conveniently provided by 15 core and earth cable, of which a length of half a meter is ample. You will have to make a hole in the Atom case to bring the cable through.

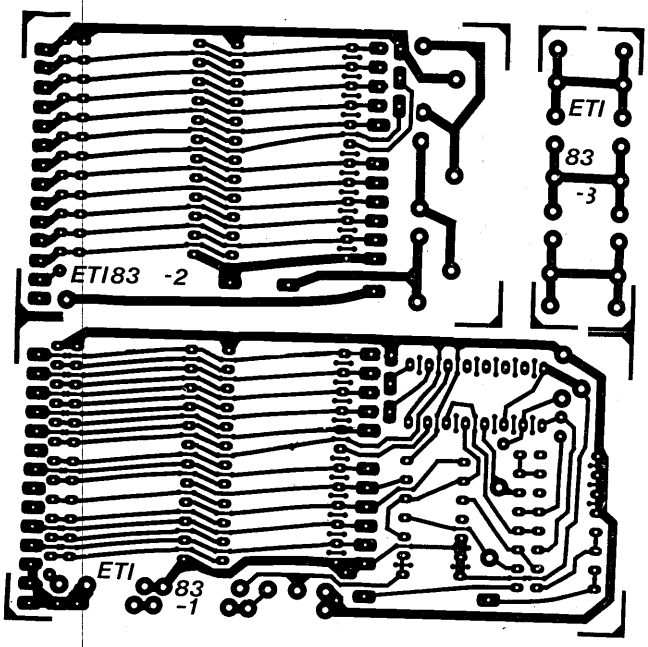
The pad is mounted in a sloping fronted Bim case, which conveniently matches the shape of the Atom case, and helps to give a uniform appearance to the finished product.

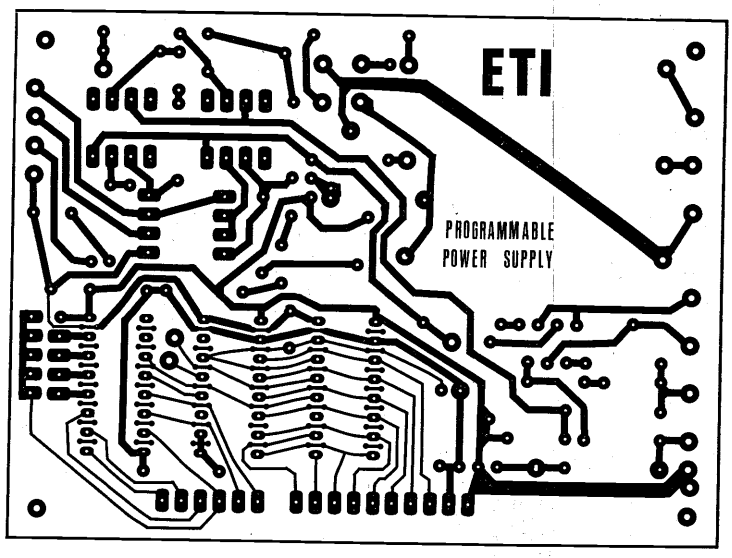

ETI
M P INTERFACE


ZX81
GRAPHICS
ETI

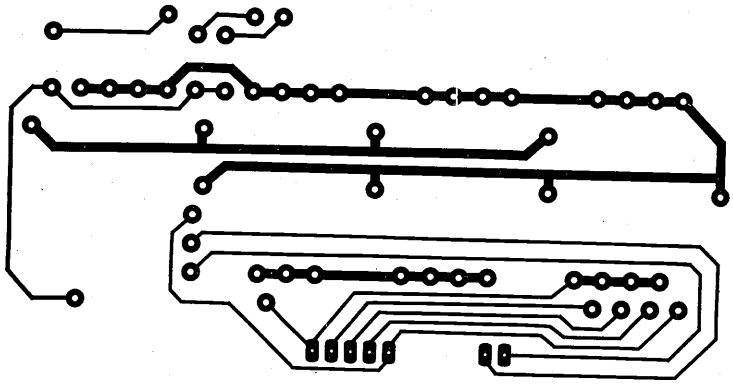

ETI
MESSAGE PANEL

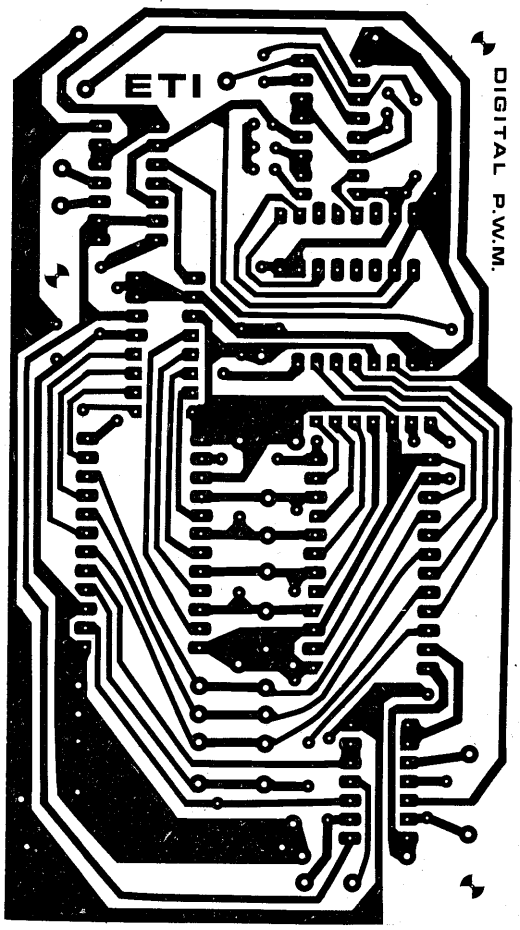
# PCB Foil Patterns



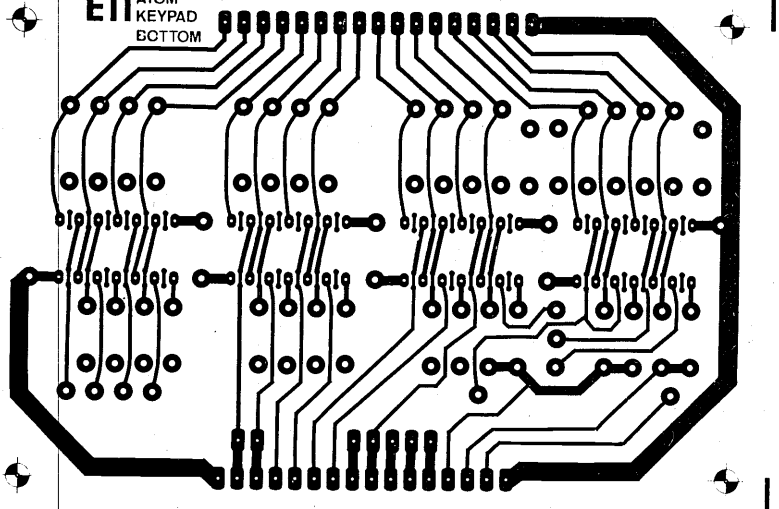



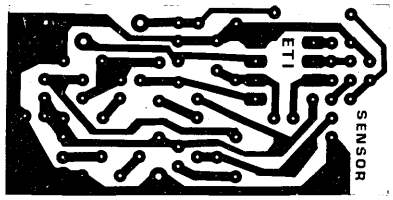

ETI ATOM KEYPAD TOP

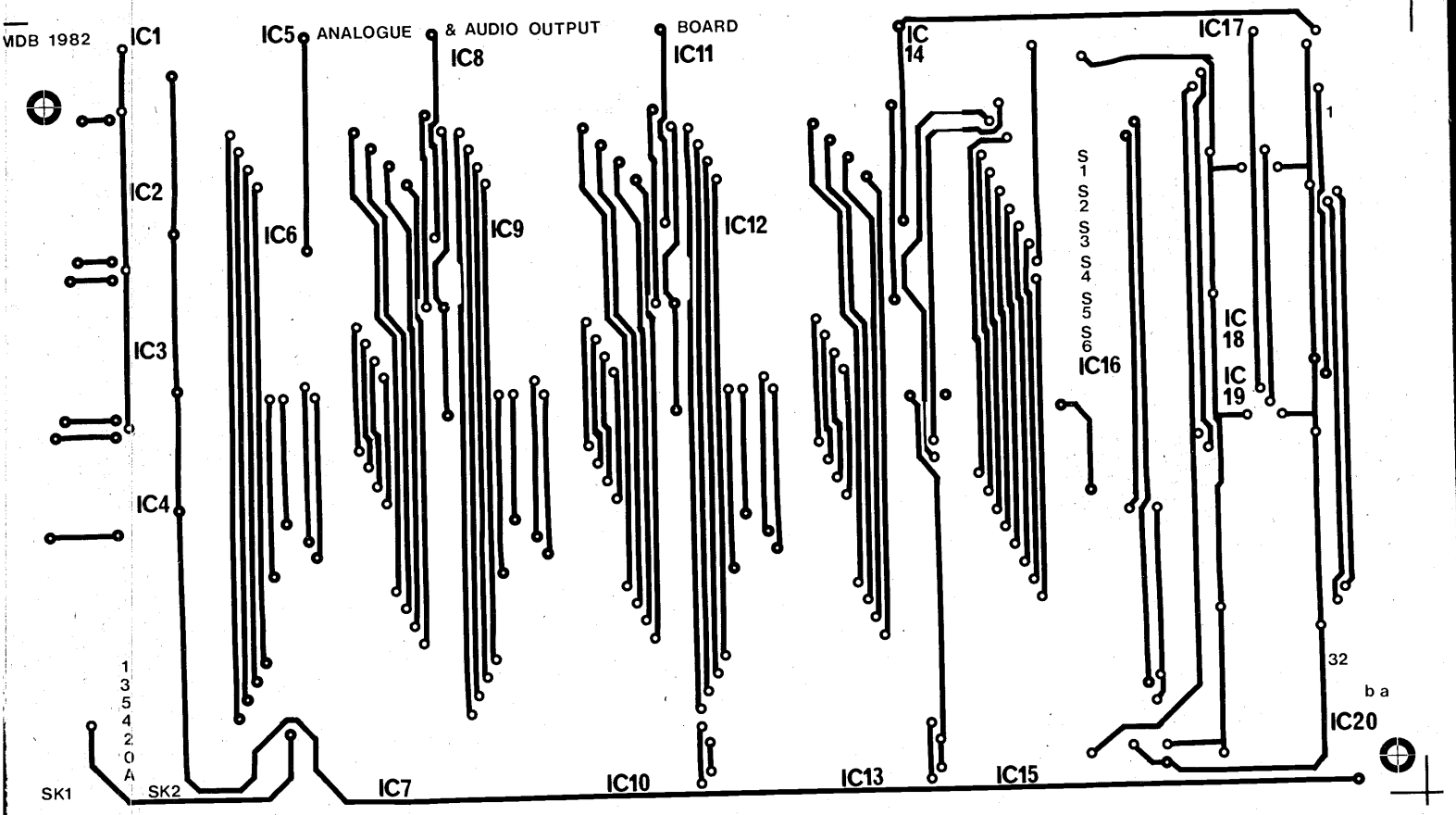


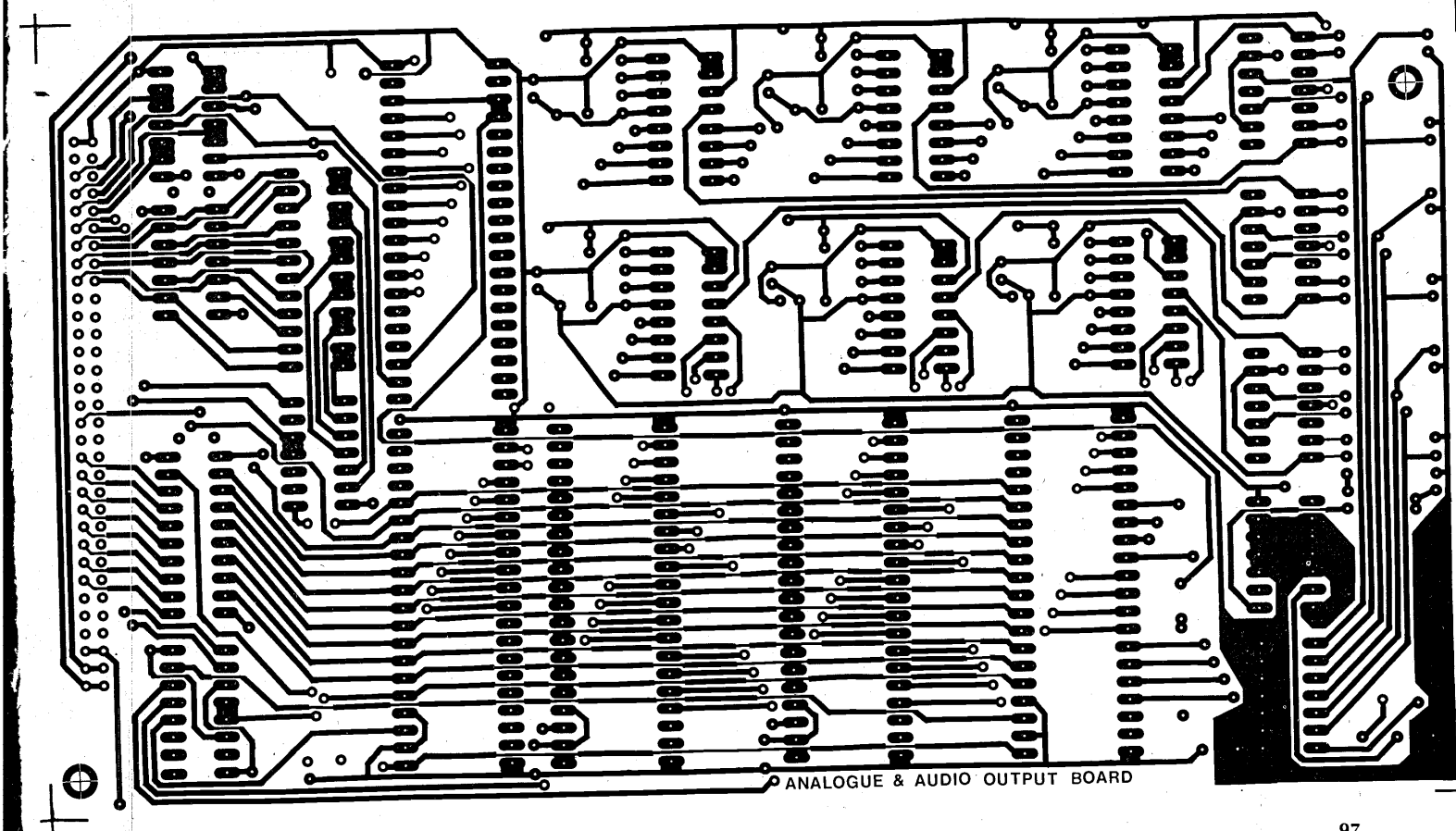



ETI ATOM KEYPAD BOTTOM

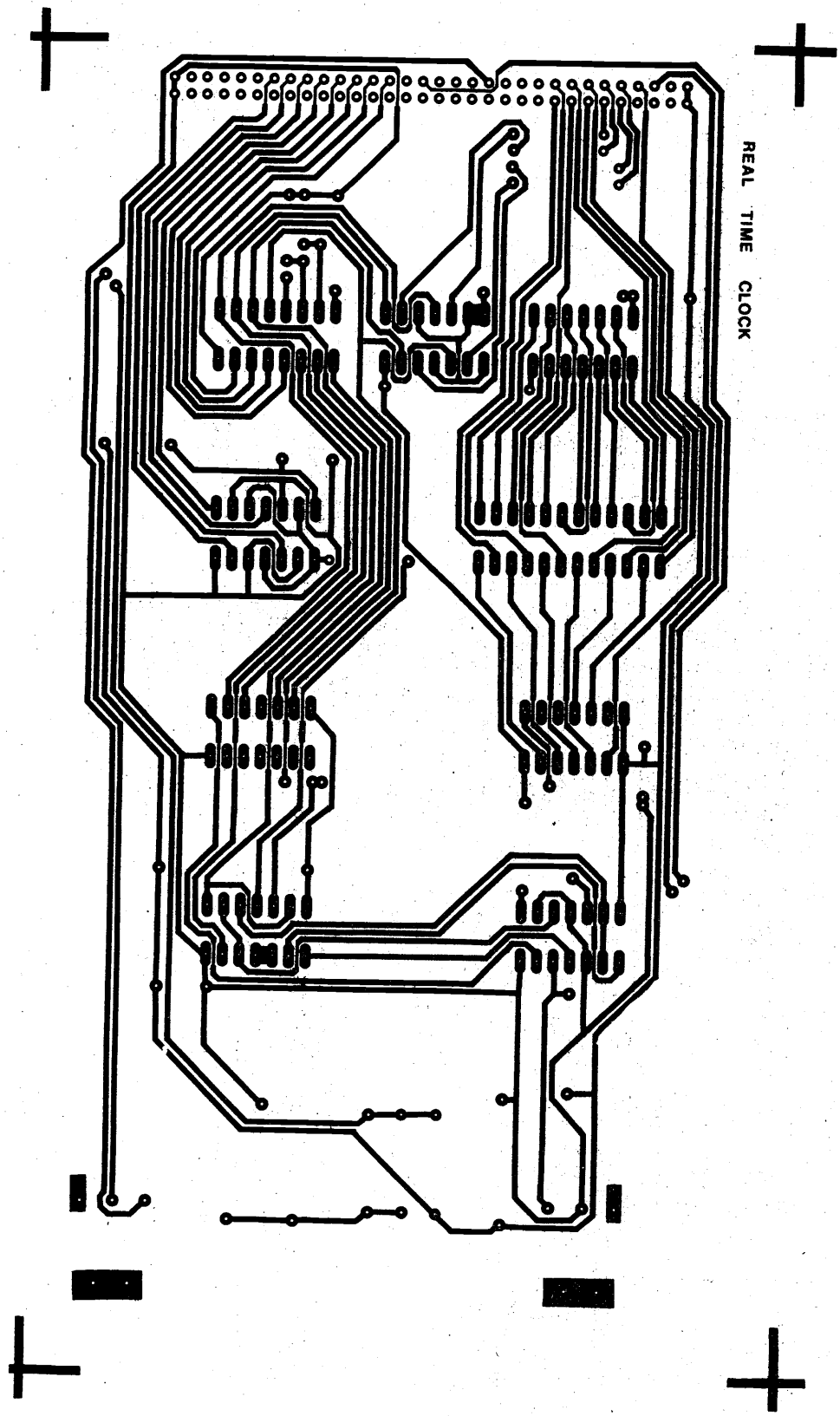





VDB 1982
IC1
IC5
ANALOGUE
& AUDIO OUTPUT
BOARD
IC8
IC11
IC14
IC17
IC2
IC6
IC9
IC12
IC3
IC4
IC16
IC18
IC19
IC20
IC7
IC10
IC13
IC15
SK1
SK2
1
32
b a
1 3 5 4 2 0 A
S1 S2 S3 S4 S5 S6


8
ANALOGUE & AUDIO OUTPUT BOARD

# PCB Foil Patterns



# electronics today international

# BOOK SERVICE

How to order: indicate the books required by ticking the boxes and send this page, together with your payment, to: ETI Book Service, Argus Specialist Publications Ltd, 1, Golden Square, London W1 R 3AB. Make cheques payable to ETI Book Service. Payment in sterling only please. All prices include P & P. Prices may be subject to change without notice.

## BEGINNERS GUIDE

- ☐ **Beginner's Guide to Basic Programming** Stephenson £5.35
- ☐ **Beginner's Guide to Digital Electronics** £5.35
- ☐ **Beginner's Guide to Electronics** £5.35
- ☐ **Beginner's Guide to Integrated Circuits** £5.35
- ☐ **Beginner's Guide to Computers** £5.35
- ☐ **Beginner's Guide to Microprocessors** £5.35

## COOKBOOKS

- ☐ **Master IC Cookbook** Hallmark £10.15
- ☐ **Microprocessor Cookbook** M. Hordeski £7.70
- ☐ **IC Op Amp Cookbook** Jung £14.25
- ☐ **PLL Synthesiser Cookbook** H. Kinley £7.70
- ☐ **Active Filter Cookbook** Lancaster £13.40
- ☐ **TV Typewriter Cookbook** Lancaster £11.15
- ☐ **CMOS Cookbook** Lancaster £11.85
- ☐ **TTL Cookbook** Lancaster £10.95
- ☐ **Micro Cookbook Vol. 1** Lancaster £15.30
- ☐ **BASIC Cookbook** K. Tracton £6.00
- ☐ **MC6809 Cookbook** C. Warren £7.25

## ELECTRONICS

- ☐ **Principles of Transistor Circuits** Amos £8.50
- ☐ **Design of Active Filters with experiments** Berlin £11.30
- ☐ **49 Easy to Build Electronic Projects** Brown £6.00
- ☐ **Electronic Devices & Circuit Theory** Boylestad £13.20
- ☐ **How to build Electronic Kits** Capel £3.55
- ☐ **How to Design and build electronic instrumentation** Carr £9.35
- ☐ **Introduction to Microcomputers** Daglecs £7.20
- ☐ **Electronic Components and Systems** Dennis £15.00
- ☐ **Principles of Electronic Instrumentation** De Sa £11.40
- ☐ **Giant Handbook of Computer Software** £12.95
- ☐ **Giant Handbook of Electronic Circuits** £17.35
- ☐ **Giant Handbook of Electronic Projects** £11.75
- ☐ **Electronic Logic Circuits** Gibson £5.55
- ☐ **Analysis and Design of Analogue Integrated Circuits** Gray £30.25
- ☐ **Basic Electronics** Grob £11.30
- ☐ **Lasers — The Light Fantastic** Hallmark £7.70
- ☐ **Introduction to Digital Electronics & Logic** Joynson £5.25
- ☐ **Electronic Testing and Fault Diagnosis** Loveday £7.85
- ☐ **Electronic Fault Diagnosis** Loveday £6.25
- ☐ **Essential Electronics A-Z Guide** Loveday £7.50
- ☐ **Microelectronics Digital & Analogue circuits and systems** Millman £12.70
- ☐ **103 Projects for Electronics Experimenters** Minis £8.30
- ☐ **VLSI System Design** Muroga £34.10
- ☐ **Power FETs and their application** Oxner £9.40
- ☐ **Practical Solid State Circuit Design** Olesky £25.00
- ☐ **Master Handbook of IC Circuits** Powers £12.85
- ☐ **Electronic Drafting and Design** Raskhodoff £22.15
- ☐ **VOM — VTVM Handbook** Risse £8.50
- ☐ **Video and Digital Electronic Displays** Sherr £28.85
- ☐ **Understanding Electronic Components** Sinclair £7.50
- ☐ **Electronic Fault Diagnosis** Sinclair £4.50
- ☐ **Physics of Semiconductor Devices** Sze £17.35
- ☐ **Digital Circuits and Microprocessors** Taub £32.00
- ☐ **Active Filter Handbook** £7.60
- ☐ **Designing with TTL Integrated Circuits** Texas £15.20
- ☐ **Transistor Circuit Design** Texas £15.20
- ☐ **Digital Systems: Principles and Applications** Tocci £12.95
- ☐ **Master Handbook of Telephones** Traister £10.00
- ☐ **How to build Metal/Treasure Locators** Traister £6.00
- ☐ **99 Fun to Make Electronic Projects** Tymony £8.50
- ☐ **33 Electronic Music Projects you can build** Winston £6.95

## COMPUTERS & MICROCOMPUTERS

- ☐ **BASIC Computer Games** Ahl £6.35
- ☐ **From BASIC to PASCAL** Anderson £9.95
- ☐ **Mastering Machine Code on your ZX81** T. Baker £7.25
- ☐ **UNIX — The Book** Banaham £8.75
- ☐ **Z80 Microcomputer Handbook** Barden £10.95
- ☐ **Microcomputer Maths** Barden £11.90
- ☐ **Digital Computer Fundamentals** Barter £9.90
- ☐ **Visicalc Book. APPLE Edition** Bell £15.55
- ☐ **Visicalc Book. ATARI Edition** Bell £15.55
- ☐ **Introduction to Microprocessors** Brunner £23.00
- ☐ **Programming your APPLE II Computer** Bryan £9.25
- ☐ **Microprocessor Interfacing** Carr £7.70
- ☐ **Microcomputer Interfacing Handbook A/D & D/A** Carr £9.50
- ☐ **Musical Applications of Microprocessors** Chamberlain £28.85
- ☐ **30 Computer Programs for the Home Owner in BASIC** D. Chance £9.25
- ☐ **Microcomputers** Dirkson £9.30
- ☐ **APPLE Personal Computer for Beginners** Dunn £9.50
- ☐ **Microcomputers/Microcomputers — An Intro** Gioone £11.80
- ☐ **Troubleshooting Microprocessors and Digital Logic** Goodman £9.25
- ☐ **Getting Acquainted with your VIC 20** Hartnell £8.50
- ☐ **Getting Acquainted with your ZX81** Hartnell £5.95
- ☐ **Let your BBC Micro Teach you to program** Hartnell £7.90
- ☐ **Programming your ZX Spectrum** Hartnell £8.50
- ☐ **The ZX Spectrum Explored** Hartnell £6.95
- ☐ **How to Design, Build and Program your own working Computer System** Haviland £9.30
- ☐ **BASIC Principles and Practice of Microprocessors** Heffer £7.15
- ☐ **Hints and Tips for the ZX81** Hewson £5.25
- ☐ **What to do when you get your hand on a Microcomputer** Holtzman £9.95
- ☐ **34 More Tested Ready to Run Game Programs in BASIC** Horn £7.70
- ☐ **Microcomputer Builders' Bible** Johnson £12.40
- ☐ **Digital Circuits and Microcomputers** Johnson £14.55
- ☐ **PASCAL for Students** Kemp £7.20
- ☐ **The C — Programming Language** Kernighan £18.20
- ☐ **COBOL** Jackson £9.25
- ☐ **The ZX81 Companion** Maunder £9.50
- ☐ **Guide to Good Programming Practice** Meek £6.40
- ☐ **Principles of Interactive Computer Graphics** Newman £13.95
- ☐ **Theory and Practice of Microprocessors** Nicholas £11.35
- ☐ **Exploring the World of the Personal Computer** Nilles £12.95
- ☐ **Microprocessor Circuits Vol. 1. Fundamentals and Microcontrollers** Noll £9.80
- ☐ **Beginner's Guide to Microprocessors** Parr £5.35
- ☐ **Microcomputer Based Design** Peatman £11.30
- ☐ **Digital Hardware Design** Peatman £9.80
- ☐ **BBC Micro Reavealed** Ruston £9.45
- ☐ **Handbook of Advanced Robotics** Safford £14.45
- ☐ **1001 Things to do with your own personal computer** Sawusch £8.50
- ☐ **Easy Programming for the ZX Spectrum** Stewart £7.15
- ☐ **Microprocessor Applications Handbook** Stout £34.40
- ☐ **Handbook of Microprocessor Design and Applications** Stout £37.60
- ☐ **Programming the PET/CBM** West £17.80
- ☐ **An Introduction to Microcomputer Technology** Williamson £8.20
- ☐ **Computer Peripherals that you can build** Wolfe £12.40
- ☐ **Microprocessors and Microcomputers for Engineering Students and Technicians** Wooland £7.10

## REFERENCE BOOKS

- ☐ **Electronic Engineers' Handbook** Fink £56.45
- ☐ **Electronic Designers' Handbook** Giacoletto £59.55
- ☐ **Illustrated Dictionary of Microcomputer Technology** Hordeski £8.45
- ☐ **Handbook for Electronic Engineering Technicians** Kauffman £27.50
- ☐ **Handbook of Electronic Calculators** Kauffman £35.00
- ☐ **Modern Electronic Circuit Reference Manual** Marcus £44.00
- ☐ **International Transistor Selector** Towers £10.70
- ☐ **International Microprocessor Selector** Towers £16.00
- ☐ **International Digital IC Selector** Towers £10.95
- ☐ **International Op Amp Linear IC Selector** Towers £8.50
- ☐ **Illustrated Dictionary of Electronics** Turner £12.95

## VIDEO

- ☐ **Servicing Home Video Cassette Recorders** Hobbs £12.95
- ☐ **Complete Handbook of Videocassette Recorders** Kybett £9.25
- ☐ **Theory and Servicing of Videocassette Recorders** McGinty £12.95
- ☐ **Beginner's Guide to Video** Matthewson £5.35
- ☐ **Video Recording: Theory and Practice** Robinson £14.40
- ☐ **Video Handbook** Van Wezel £21.90
- ☐ **Video Techniques** White £12.95

---

**Please send me the books indicated. I enclose cheque/postal order for £.....Prices include postage and packing**

**I wish to pay by Access/Barclaycard. Please debit my account.**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 2 | 2 | 4 | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 9 | 2 | 9 | | | | | | | | | | | | |

Signed.....

Name .....

Address.....

.....

.....

# hobbyboard

NEW

# COMPUTING AND ELECTRONICS

## BBC USER PORT ADD-ON Range

### BBC 1MHz I / O INTERFACE

Provides 8 Control Outputs and 4–6 Signal Inputs for Robotic, Industrial & many other control applications. Available in Kit form or fully assembled & tested.
**HB/2031 P.O.A.* Kit HB/2030 £26.50***

### BBC 1MHz 16 Bit VIA

The User Port only provides 8 Bit Control, yet many applications require 16 Bit Data Control. This Adaptor enables projects using 16 Bits to be developed via the MHz Port. Available in Kit form or fully assembled.
**HB/3500 P.O.A.* Kit HB/3505 P.O.A.***

### BBC USER PORT SERVO INTERFACE

A simple Kit to enable the 8 Bit User Port to operate all types of Servo motors – adapts to any 8 Bit User Port.
**HB/3550 P.O.A.* Kit HB/3555 P.O.A.***

### BBC USER PORT DECODER

Converts the 8 Bit User Port to provide 8 Addressed Outputs, 4 Data Inputs and a 4 Bit Data Input – many different Add-on units & projects may then be connected to enable one or all projects to be run simultaneously under program control.
**HB/3560 P.O.A.* Kit HB/3565 P.O.A.***

### NIKAM 6502 MICRO CONTROLLER

A star product which enables programs developed on the BBC Assembler to be down- loaded via the RS 423 Port and blown onto EPROM. The Micro Controller will then autostart on power up to perform as a dedicated Stand Alone unit. Applications include Robotics, Data Logging, Timing, Machine Control, Speech & Sound Synthesis etc.

**HB/3570 P.O.A.* Kit HB/3575 P.O.A.***

### ELECTRON I / O INTERFACE

This useful Interface provides the electron with a normal BBC-type 8 bit User Port **PLUS** capability of 16 Bit Port or Printer Output plus 8 Addressed Outputs, 4 Data Inputs and a 4 Bit Data Input. 'Enormous potential for all types of applications.'
**HB/3580 P.O.A.* Kit HB/3585 P.O.A.***

## Easy Add-ons for ZX Spectrum & ACE

**17 exciting electronic projects to build and run your own micro.**

- **LIGHT PEN**
- **PICTURE DIGITISER**
- **KEY PAD**
- **MODEL CONTROLLER**
- **WEATHER STATION**
- **+ OTHER EXCITING & INTERESTING PROJECTS**

**REALISE THE REAL WORLD POTENTIAL OF YOUR MICRO**

A newly released book written by well known author Owen Bishop and published by Bernard Babani gives full descriptive details on how to build all 17 projects - all are fairly simple and inexpensive to construct - The most complex component (the DECODER) is supplied in kit form ready to assemble with all components and plated through PCB. - Components for the projects are readily available locally or found in your workshop drawers.
Simple programmes are included to get you started but of course the more experienced programmer can have hours of fun writing complex programmes. Please state computer when ordering.

| | |
|---|---|
| Order ref HB/2000 "EASY ADD-ONS" BOOK + DECODER/KIT | **£24.00** |
| Order ref HB/2001 "EASY ADD-ONS" BOOK ONLY | **£3.00** |
| Order ref HB/2002 DECODER KIT ONLY | **£22.00** |
| Order ref HB/2003 DECODER PCB ONLY | **£8.00** |

**AVAILABLE FEBRUARY**
Easy Add-on Projects for
BBC, Electron, Commodore 64, VIC 20

NEW

## Computer Cables & Connectors

We now offer an extensive range of computer cables & connectors including a spectrum user port extender cable. The following are just a few of our extensive range, send for new cable & connector price list.

### Spectrum User Port Extender

This 56 way IDC connector & ribbon cable assembly specially produced to fit the Sinclair Spectrum overcomes the interconnection problems associated with user port add-ons. Available as a double ended assembly with a user-port PCB male converter or single ended for wiring onto your own equipment.
**HB/2069 6" double ended extender & PCB £8.85***
**HB/2068 9" single ended assembly £4.76***
**HB/2093 56 way IDC connector only £3.34***

### BBC Printer Lead

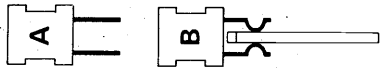
26 way BBC/Centronics parallel printer lead **HB/3001 £7.00***

### VIC/CBM64 Cassette Interface

Works with most data recorders & will load commercial programs!
**HB/3000 £12.50***

### Edge Connectors

Double sided edge connectors 0.1" pin to pin and 0.2" row to row. All sizes are available in straight wire wrap Style 'A' or with the leads cranked to clip on the edge of a 1.6 mm circuit board Style 'B'. The range is chosen for compatability with commonly found home computers, as well as for general use.
28 + 28 way, tinned contacts, keyed pin 5 suitable for use with Sinclair SPECTRUM




28 + 28/5 Connector Style 'A' **HB/440 £1.70***
28 + 28/5 Connector Style 'B' **HB/441 £1.75***
23 + 23 way, tinned contacts, keyed pin 3 suitable for use with Sinclair ZX81
23 + 23/3 Connector Style 'A' **HB/438 £1.40***
23 + 23/3 Connector Style 'B' **HB/439 £1.65***

Universal user port extension PCB as supplied in Micro Interface Kit. **HB/2012 £0.87***

Interface Prototype board can be used with style 'A' connectors & user port extension to give flexible add-on system **HB/2091 £4.25***

### Cambridge Computing ZX / Spectrum Intelligent Joystick I / F

● Treble your game scores overnight ● Suitable for Atari type joysticks. The kit provides an interface to enable ALL games programs to be played with joystick control. Tell it once & the keys for that game are remembered forever.
**Spectrum Kit & Joystick HB/2061 £27.31***
**Less Joystick £20.80***
**ZX81 Kit & Joystick HB/2060 £27.31***
**Less Joystick £20.80***
***Also available fully assembled.***
**Joystick, Interface & Tape £34.90***
**Interface & Tape only £27.90***

### ZX81 Hi Resolution Graphics Kit

Improves screen resolution to 256 x 176 pixels enabling superior graphics to be easily programmed. Plugs directly into ZX81 ROM socket & is complete with extensive software tape.
**Order Ref. HB/2070 £22.00***

### BBC Micro Interface (1 MHz I / O)

Initially designed to interface the Acorn BBC Micro with turtle type robotics this unit, supplied in kit form, provides 8 Control Outputs and 4 Data Inputs providing a usefull interface to operate numerous control applications.
**Order Ref. HB/2030 £26.50***
*Interfaces available for other computers, send for details.*

## Printed Circuits
**A full range of kits & materials to make your own PCB's.**

### Direct Etch Kit

COPY DIRECT FROM MAGAZINE or OWN DESIGN Simple system – Complete kit containing PCB, Pattern Transfer & Etch Resist Sheets, Tray & Etchant, Copper Cleaning Block, Gloves & full instructions. **HB/1 £18.00***

### Photo Resist Kit

Complete kit containing artwork PCB, and all the necessary process materials. **HB/2 £29.00***

### DIY UV Exposure Unit

Perfect results everytime. Kit contains: Lamp, Holder. & Shade together with full instructions for DIY Unit which offers PCB, Precision Photo, Lable & Panel manufacture. **UV/1 £27.00***

### MATERIALS – a small selection from our catalogue.

| | | |
|---|---|---|
| HB/019 | 4 x 3 inches single sided (3) | £1.26* |
| HB/020 | 6 x 4 inches single sided (2) | £1.68* |
| HB/021 | 6 x 9 inches single sided (1) | £1.89* |
| HB/025 | 4 x 3 inches double sided (3) | £1.44* |
| HB/026 | 6 x 4 inches double sided (2) | £1.92* |
| HB/027 | 6 x 9 inches double sided (1) | £2.16* |
| | Assorted single & double sided approx 1 sq. foot | £2.25* |
| | Ferric Chloride etchant granules to make 1 litre | £1.25* |
| HB/016 | Protective Gloves (3 pairs) | £1.05* |
| HB/084 | 1 litre storage bottle | £0.50* |
| HB/014 | plastic etching tray | £2.54* |
| HB/088 | Instruction sheet for PCB making | £0.60* |
| HB/630 | "How to Design & Make Your Own PCB" Book | £1.95* |
| HB/017 | Copper cleaning block | £1.49* |
| HB/001 | Etch resist transfers (assorted pads) | £2.08* |
| HB/004 | Transfer spatula | £0.70* |
| HB/083 | Etch resist pen | £1.35* |
| HB/209 | 0.020 inch track tape | £1.05* |
| HB/211 | 0.031 inch track tape | £1.05* |

## Panels & Labels
**Make your own with PHOTOTOOL. Full range of separate materials available.**

### Fototool Kit

Containing artwork, film and all the necessary process materials required for professional quality labels and panels. CAN ALSO BE USED TO PRODUCE PRECISION PCB PHOTOMASTERS.
**HB/3 £27.00***

### UV Exposure Unit and Artbox (Ref. UV2)

A portable ready made unit containing two 8 watt UV tubes giving a 6" x 9" exposure area which may also be used as a light box with the UV filter supplied.
**UV/2 £64.00***

### Artwork Aids

A full range of PCB & Panel Draughting Aids. – *See Catalogue.*

### Hobbyboard Catalogue

**The complete Printed Circuit Workshop. Newly Published Full Catalogue price £1.50 (refundable with 1st order over £10).**

***Prices inclusive of VAT, Carriage 60p in U.K. Overseas orders please add extra carriage to published prices.**


**KELAN ENGINEERING LTD.**
Circuit Products & Components Division.
27-29 Leadhall Lane · Harrogate.
North Yorkshire HG2 9NJ.
Tel: (0423) 870938.
A FARNELL ELECTRONICS COMPANY