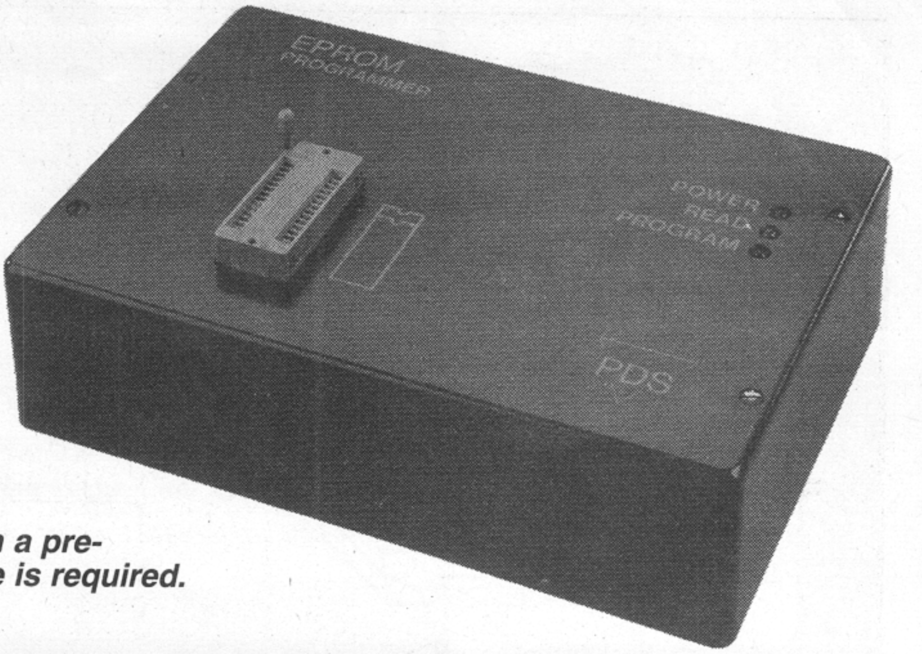


A project as an aid to other projects. Paul Stenning provides an excellent design and an invaluable aid when a pre-programmed device is required.

EPROM Programmer

PROJECT 1

There has long been a requirement for a cheap straight-forward EPROM Programmer, suitable for occasional home use, without the extra features found on many commercial units. The EPROM Programmer presented here will program the standard 27 family of devices, from 2716 to 27512, and can be used with any computer which has an RS232 serial port. I have avoided the usual "Catch 22" situation of requiring a programmed EPROM to make the EPROM programmer work! This design uses readily available components to reduce the likelihood of obsolescence.

The unit is powered from an external PSU, since this is cheaper than buying the individual parts! One of the low cost unregulated types with an integral mains plug is suitable, providing it is capable of supplying between 10 and 18 Volts DC at 500mA without too much ripple, the voltage regulator IC21 will run cooler if the voltage is nearer the lower end of this range. The types sold for powering electronic keyboards appear to be the most suitable. The prototype is powered by an old Sinclair ZX81 PSU (type UK1200).

The programmer itself is dumb and is fully controlled by the host computer via the serial port. Control software can be written in BASIC, and a suitable listing for IBM PC compatible computers is given later. Additional software listings are given for initial testing, and to convert to and from standard Intel-HEX files. A disk is available from the author, containing these programs, as well as comprehensive menu driven control software and a selection of useful utilities, see Buylines for details.

Please note that the programming algorithms used may not be exactly as specified in some EPROM manufacturer's data sheets. Because of this the unit cannot be guaranteed to program every device successfully, however no problems have been experienced to date.

HOW IT WORKS

HARDWARE

The circuit may appear complicated initially, however it comprises of several relatively straight-forward sections. It is not necessary to understand the operation of the circuit to build and operate the unit with the software given, however a good understanding is most useful if you wish to write your own control software.

Note that when a number is followed by an "h" in the following description, for example 27h, it is a hexadecimal number, and when a number is followed by a "b" it is binary, other numbers are decimal. Any signal name which is followed by a "-", for example STROBE-, is active low, on the circuit diagram this will be shown with a bar over the name.

Useful information on EPROM pin-outs and programming requirements can be found on pages 498-499 of the 1993 Maplin catalogue, the project was designed around this data (note that the Maplin programming information for the 2732 is incorrect).

IC2 and surrounding components generate the Baud Rate clock signal (CLOCK1) for IC3, with the rate selectable by LK1. The CLOCK2 signal is used to produce the programming timing pulses, see later. IC3 is a 6402 "Universal Asynchronous Receiver/Transmitter" (UART). In this application it is configured for 8 data bits, 1 stop bit, no parity checking. Note that unused pin 2 is taken low, this pin is used on the RCA CDP1854 to select standard operation mode, this device is otherwise pin compatible can therefore be used in place of the 6402. IC3 is reset by IC4:A and associated components.

When serial data is received on pin 20 of IC3, it is converted to a parallel output on pins 5-12 (R0-R7), and pin 19 goes high. A short time later (set by R4 and C8) pin 18 is taken low, which clears the high on pin 19. The pulse so generated on pin 19 is referred to as the STROBE signal, and indicates to other parts of the circuit that data has been received and is valid.

If pin 23 of IC3 (SEND) is pulsed low, data on pins 26-33 (D0-D7) is transmitted in serial form on pin 25.

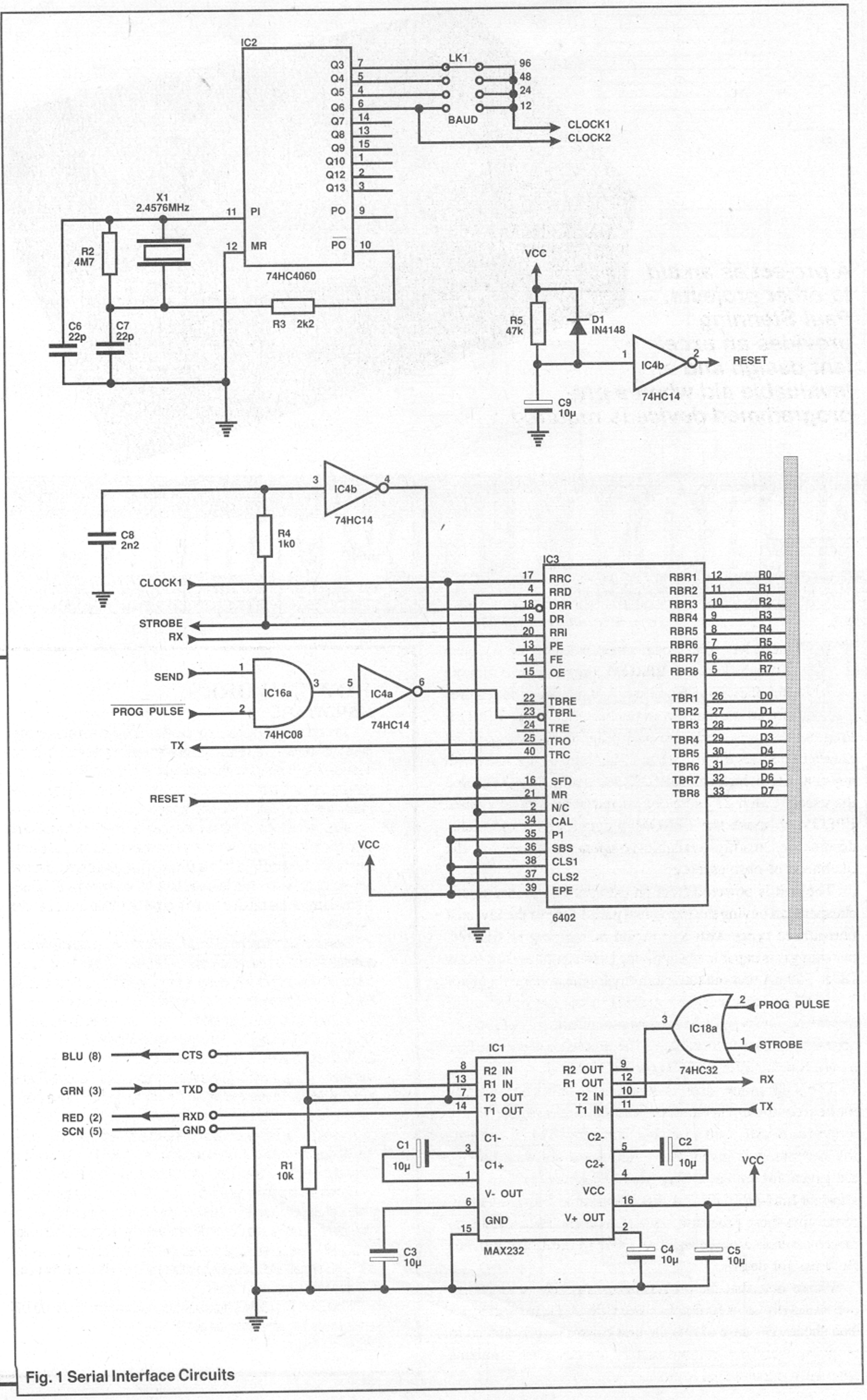


Fig. 1 Serial Interface Circuits

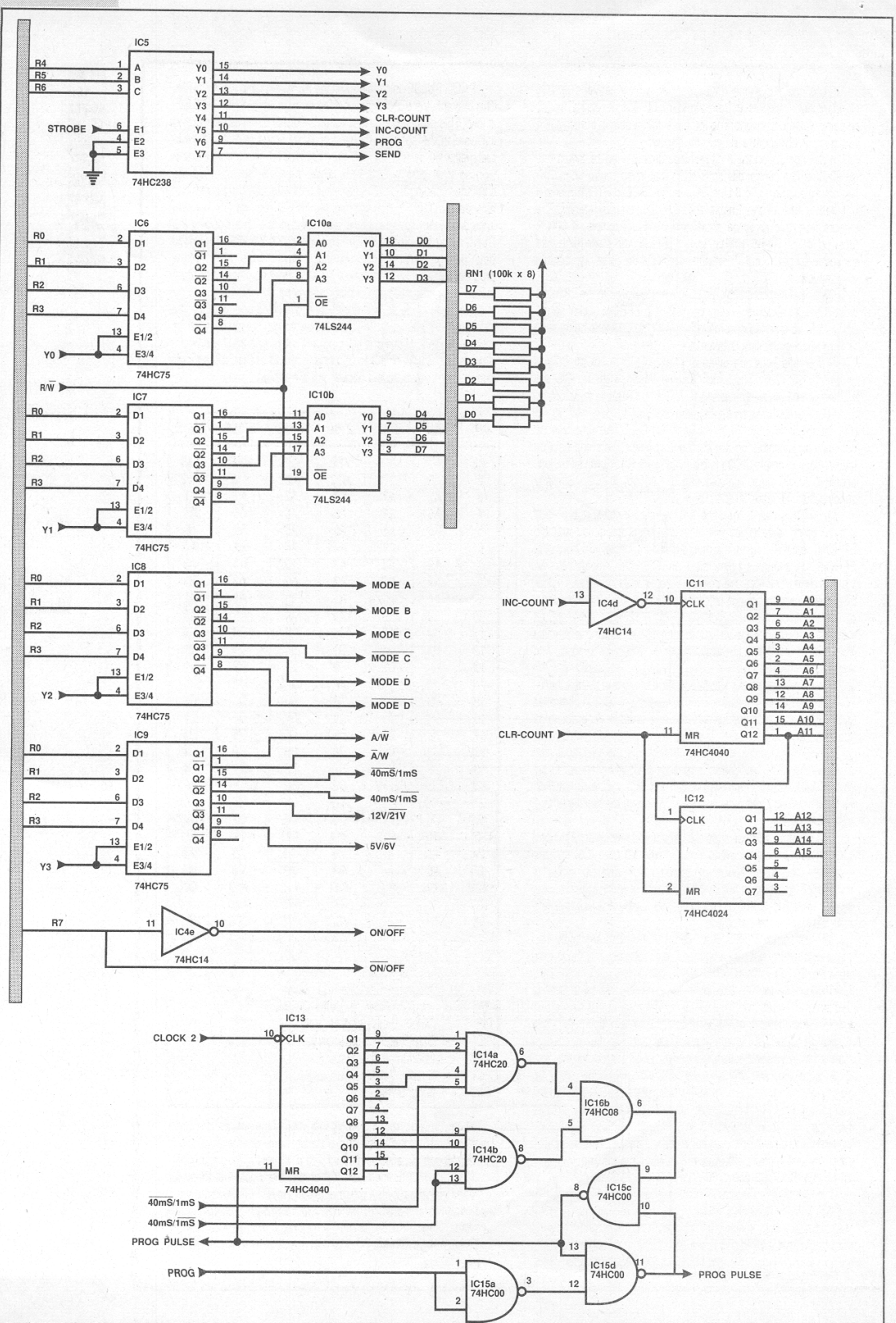


Fig. 2 EPROM Programmer Logic

IC1 is the serial line driver/receiver, which converts RS232 standard serial signals to/from standard 5V TTL levels. This IC contains voltage convertors to produce the required transmission voltages (+/- 9V) from a single +5V supply.

In this application the 8 bit received data is used as 4 data bits on R0-R3 and 4 control bits on R4-R7. 3 of these control lines (R4-R6) are taken to a "3 To 8 Line Decoder", IC5. The STROBE signal is taken to one of the Enable lines on IC5, thereby causing the outputs from this IC to be short pulses. For example, if 00h is received, lines R4-R6 will be low, STROBE will pulse high and consequently pin 15 of IC5 will pulse high and the remaining outputs will stay low.

The remaining control bit (R7) controls the power to the EPROM, when it is high all power is removed from the EPROM socket so the device can be inserted or removed. When the device is being read or programmed this line is taken low.

The 4 data lines are taken to the inputs of 4 "Quad D-Type Latches", IC6-IC9. The Latch Enable lines on these IC's are connected to 4 of the outputs of IC5 (Y0-Y3). Data can therefore be stored in these latches by sending the required data on lines R0-R3 and the latch number (0-3) on lines R4-R6. The data will be continuously available on the latch output lines. Referring back to the previous example, if 00h is received, lines R0-R3 will all be low, and line Y0 will pulse high, storing 0h (0000b) in IC6. Similarly sending 35h will store 5h (0101b) in IC9

Latch IC6 is used to hold the least significant nibble (4 bits - half a byte) of the data which will ultimately be programmed into the EPROM, whilst IC7 holds the most significant nibble. IC8 holds the EPROM type information (see later), and IC9 holds 4 setup bits. Bit 1 of IC9 controls whether the unit is in Read or Write (Program) mode, bit 2 sets the programming pulse length to either 1 or 40 milliseconds, bit 3 sets the programming voltage to either 12 or 21 volts, and bit 4 sets the supply voltage whilst programming to either 5 or 6 volts. Note that the latches have active-high and active-low outputs, and one or both may be used.

The data outputs from IC6 and IC7 are taken to a tri-state buffer (IC10), which in turn drives the EPROM data lines. IC10 is controlled by the R/W- line, such that it's outputs are enabled in Program mode and tri-state in Read mode.

The EPROM data lines are also connected to the data input lines on the UART (IC3), and the Y7 output from IC5 is connected to pin 23 of IC3 (SEND). Therefore if 70h is received, the data currently on the EPROM Data Lines is transmitted back along the serial interface to the computer.

The EPROM Address lines are controlled by counters IC11 and IC12. These are connected to lines Y4 and Y5 from IC5, and are therefore cleared to 0000h by sending 40h and the count is incremented by sending 50h. This approach is quicker than having to send the actual address each time since it only requires one byte to be sent along the serial link.

Line Y6 from IC5 (PROG) is used to start a programming pulse. The programming pulse (either 1ms or 40ms) is produced by dividing down the CLOCK2 signal. This signal has a frequency of 19.2kHz (2.4576MHz divided by 128), which equates to a period of 52.1µs. This is fed to counter IC13, which is normally held reset by the Set/Reset flip-flop built from IC15c and IC15d. When the PROG line pulses high the flip-flop changes state and the counter starts counting the CLOCK2 pulses. If a 1ms pulse is required, pin 5 of IC14a will be high and pins 12 and 13 of IC14b will be low. When the count reaches 19 (52.1µs x 19 = 989.9us), all the inputs of IC14a will be high, it's output will therefore go low, changing the state of the flip-flop again, via IC16d, and resetting the counter. A 0.99ms pulse will therefore be present on the output of the flip-flop, 0.99ms being well within the 1ms +/- 5% specification. If a 40ms pulse is required IC13 counts 768 CLOCK2 pulses (52.1us x 768 = 40.01ms).

The PROG PULSE and STROBE signals are coupled via IC18a and IC1 to the RS232 CTS (Clear To Send) line. This prevents the host computer from sending further data whilst a program pulse is occurring or while STROBE is still high. This means that data can be sent as fast as possible and no delays are needed in the software.

PROG PULSE- is coupled with the SEND signal to IC3 by IC16a. This prevents the SEND signal getting through whilst PROG PULSE- is low. This situation will never occur in normal use, however during initialisation the software requests a 40ms program pulse immediately followed by a SEND. If the RS232 CTS line is present and working then data will be sent, since the CTS line will stop the computer sending the SEND request until after the prog pulse has finished. If CTS is not working the SEND request will be sent immediately but no data will be returned due to IC16a stopping the SEND pulse reaching IC3. The software notes the lack of received data, prints a warning and then adds delays itself to allow for the program pulse. You should try to get CTS working properly as there will be a significant speed penalty otherwise.

We now come to the connection switching required for the different EPROM types. The table below shows the pin-outs for the 6 types of EPROM this unit will program. 2716 and 2732 are 24 pin devices and fit into pins 3 to 26 of the EPROM socket, the pin-outs shown below relate to the socket, not the device.

PIN	2716	2732	2764	27128	27256	27512
#1	**	**	VPP	VPP	VPP	A15
2	**	**	A12	A12	A12	A12
3	A7	A7	A7	A7	A7	A7
4	A6	A6	A6	A6	A6	A6
5	A5	A5	A5	A5	A5	A5
6	A4	A4	A4	A4	A4	A4
7	A3	A3	A3	A3	A3	A3
8	A2	A2	A2	A2	A2	A2
9	A1	A1	A1	A1	A1	A1
10	A0	A0	A0	A0	A0	A0
11	D0	D0	D0	D0	D0	D0
12	D1	D1	D1	D1	D1	D1
13	D2	D2	D2	D2	D2	D2
14	GND	GND	GND	GND	GND	GND
15	D3	D3	D3	D3	D3	D3
16	D4	D4	D4	D4	D4	D4
17	D5	D5	D5	D5	D5	D5
18	D6	D6	D6	D6	D6	D6
19	D7	D7	D7	D7	D7	D7
#20	CE/PP+	CE/PP-	CE	CE	CE/PP-	CE/PP-
21	A10	A10	A10	A10	A10	A10
#22	OE	OE/VPP	OE	OE	OE	OE/VPP
#23	VPP	A11	A11	A11	A11	A11
24	A9	A9	A9	A9	A9	A9
25	A8	A8	A8	A8	A8	A8
#26	VCC	VCC	NC	A13	A13	A13
#27	**	**	PP-	PP-	A14	A14
28	**	**	VCC	VCC	VCC	VCC

**	=	No Pin
NC	=	No Connection
PP+	=	Program Pulse, Positive Going
PP-	=	Program Pulse, Negative Going
OE	=	Output Enable (Active Low)
CE	=	Chip Enable (Active Low)
VPP	=	Programming Voltage
VCC	=	Supply Voltage
GND	=	Ground (0V)

It can be seen from the above that most of the pins are the same for all devices. Only 6 pins require special attention, these are 1, 20, 22, 23, 26 and 27, and are marked with a "#" next to the pin number.

The four MODE lines from IC9 control the function of these six pins and are set up by the software to suit the EPROM type required. Note that two of the active-low lines are also used. The table below shows the logic levels on each of these lines, for each of the 6 types of EPROM, together with the code that needs to be sent to select that type.

EPROM TYPE	A	B	C	C-	D	D-	SETUP CODE
2716	1	1	1	0	1	0	2Fh
2732	1	0	1	0	0	1	25h
2764	1	1	0	1	0	1	23h
27128	1	1	0	1	0	1	23h
27256	1	0	0	1	0	1	21h
27512	0	0	0	1	0	1	20h

(Columns A to D- are headed MODE LINE.)

I will now describe what this means for each type of EPROM.

2716:

PIN 1 - not used.

PIN 20 - MODE B and MODE D are both high, so IC17b pin 6 remains high, IC16c pin 8 follows PROG PULSE, as does IC17c pin 8 which is coupled via D6 to PIN 20. PIN 20 is only pulled up in Program mode, so in Read mode it is permanently low.

PIN 22 - MODE A and MODE D are both high, so IC15b pin 6 is low and Q1 and Q2 remain off. R-/W is coupled to the pin via D2.

PIN 23 - MODE D is high, so Q5 and Q6 are on, coupling VPP to PIN 23.

PIN 26 - MODE C is high, so Q7 and Q8 are on, coupling V+ to PIN 26.

PIN 27 - not used.

2732:

PIN 1 - not used.

PIN 20 - MODE B and MODE D are both low. IC16c pin 8 will remain low and U17:B pin 6 will follow PROG PULSE. IC17c pin 8 will therefore be PROG PULSE inverted, and is coupled via D6 to PIN 20. PIN 20 is only pulled up in Program mode, so in Read mode it is permanently low.

PIN 22 - MODE D and MODE C- are both low and MODE A is high, so IC15b pin 6 is high. This switches on Q1 and Q2, coupling VPP to PIN 22

PIN 23 - MODE D is low, so Q5 and Q6 remain off. A11 is coupled to PIN 23 via D4.

PIN 26 - as 2716.

PIN 27 - not used.

2764 and **27128**:

PIN 1 - MODE A is high. This switches on Q3 and Q4, coupling VPP to PIN 1.

PIN 20 - MODE B is high and MODE C is low. IC16c pin 8 remains low and U17:B pin 6 remains high, therefore IC17c pin 8 remains low.

PIN 22 - MODE A and MODE C- are both high, so IC15b pin 6 is low and Q1 and Q2 remain off. R-/W is coupled to PIN 22 by D2.

PIN 23 - as 2732.

PIN 26 - MODE C is low so Q7 and Q8 remain off. A13 is coupled to PIN 26 by D5. Note that in a 2764 EPROM there is no connection to PIN 26.

PIN 27 - MODE B is high and A14 will remain low since it is outside the addressing range. IC17d pin 11 will therefore follow PROG PULSE-.

27256:

PIN 1 - as 2764.

PIN 20 - as 2732.

PIN 22 - as 2764.

PIN 23 - as 2732.

PIN 26 - as 2764.

PIN 27 - MODE B is low so IC16d pin 11 will remain low. IC17d pin 11 will follow A14.

27512:

PIN 1 - MODE A is low so Q3 and Q4 will remain off. A15 is coupled to PIN 1 by D3.

PIN 20 - as 2732.

PIN 22 - MODE A is low so IC15b pin 6 will be high, switching on Q1 and Q2, coupling VPP to PIN 22.

PIN 23 - as 2732.

PIN 26 - as 2764.

PIN 27 - as 27256.

Note that D6 and D7 are germanium types, since silicon types would cause the logic 0 inputs to the EPROM to be at the limit of the specifications.

The two logic gates which would otherwise be unused, are used to control Read and Program LED's. IC18b pin 6 goes low when R-/W and ON-/OFF are both low, lighting D9 (Read) via Q10, whilst IC18c pin 8 goes low when R/W- and ON-/OFF are low, lighting D8 (Program) via Q9. When both LED's are off it is safe to fit or remove the EPROM, the other lines would have been set low by the software before taking ON-/OFF high.

IC21 is the main 5 Volt regulator and powers all the logic IC's. D11 protects the whole circuit against reverse polarity, since it is easily possible to reverse the polarity on the sort of PSU used. LED D10 indicates that the unit is powered up.

IC20 supplies the power to the EPROM, and is enabled by the ON/OFF- line via Q17 and Q18. Q19 raises the GND pin of IC20 by 1 Volt, giving 6 Volts. When 5 Volts is required Q19 is shorted out by Q20 controlled by IC18d, this happens when 5V/6V- or R/W- is low.

IC19 is a step-up switching regulator which produces the programming voltage required. The voltage is set by shorting out sections of the resistor chain with transistors. 2716 EPROMs require 25 Volts, not 21 Volts, so the MODE D- line is used to set this. The R-/W line controls Q15 and Q16 which connect the power to the EPROM when the unit is in Program mode.

PARTS LIST

RESISTORS (all 1/4W 5% or better)

R1,6-19,27-33,36-38,41,42	10k
R2	4M7
R3	2k2
R4	1k0
R5	47k
R20,21	22k
R22,23,40	330R
R24	0R47 (or 2 x 1R0 in parallel)
R25	1k2
R26	8k2
R34	5k6
R35	1k5
R39	470R
RN1	100k x 8 SIL resistor network
RV1	470R or 500R horizontal preset
RV2-4	4k7 or 5k0 horizontal preset

CAPACITORS

C1-5,9,15,16	10µ 16V radial
C6,7	22p 0.2" pitch ceramic
C8	2n2 0.2" pitch ceramic
C10	220p 0.2" pitch ceramic
C11	470n 0.2" pitch
C12	47µ 35V radial
C13	47µ 16V radial
C14,17	1µ0 16V radial
C18-20,22-27	100n 0.2" pitch
C21	220µ 25V radial

INDUCTORS

L1	470µH 2.3A bobbin type

SEMICONDUCTORS

IC1	MAX232
IC2	74HC4060
IC3	6402
IC4	74HC14
IC5	74HC238
IC6-9	74HC75
IC10	74LS244
IC11,13	74HC4040
IC12	74HC4024
IC14	74HC20
IC15	74HC00
IC16	74HC08
IC17,18	74HC32
IC19	TL497
IC20,21	7805
Q1,3,5,7,15,17	ZTX750/751/752
Q2,4,6,8,13,14,16,18-20	BC547/548/549
Q9-12	BC557/558/559

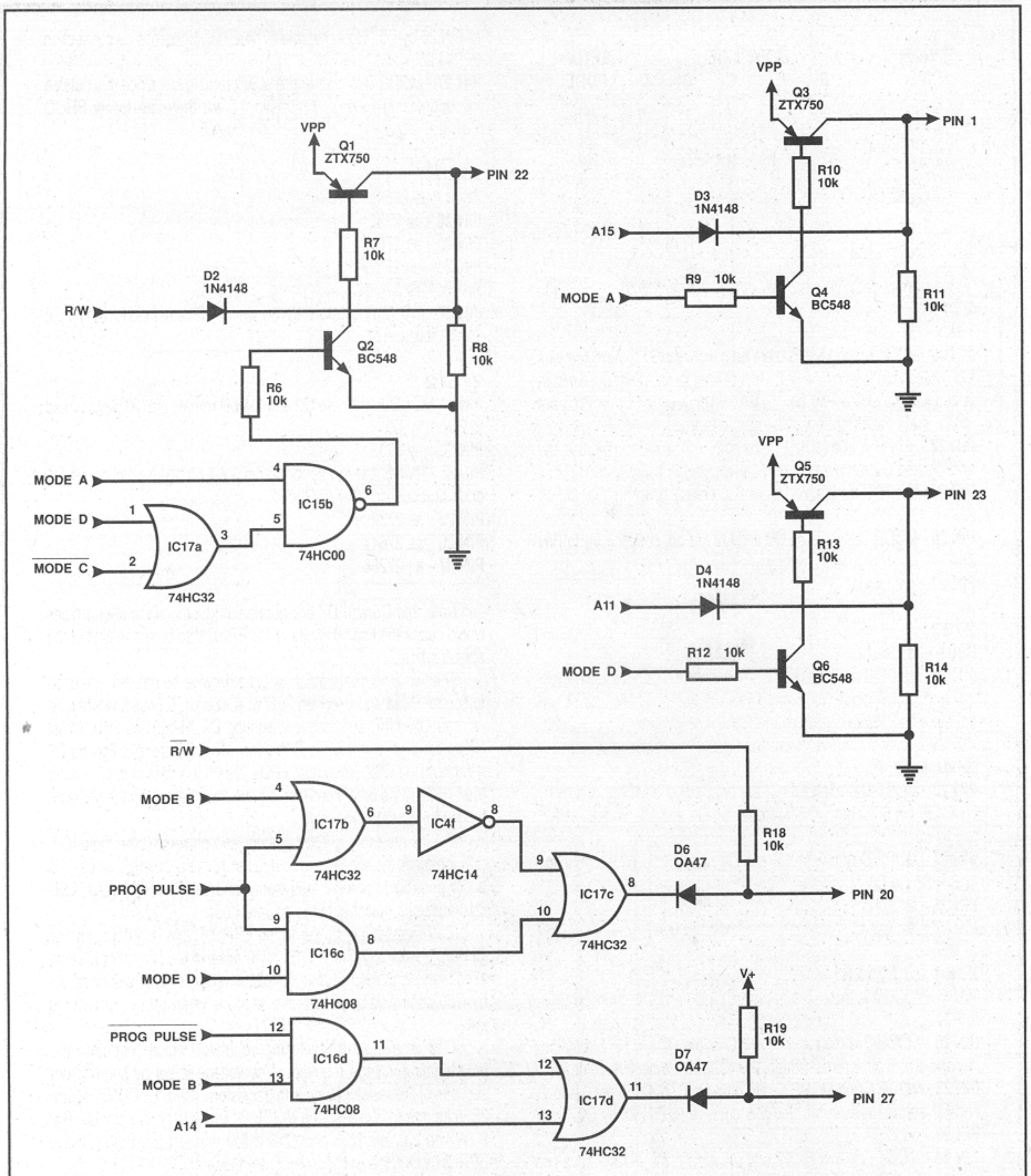


Fig. 3a Control logic for EPROM socket

D1-5	1N4148 silicon signal diode
D6,7	OA47 germanium signal diode
D8	Red LED
D9	Yellow LED
D10	Green LED
D11	1N4001

MISCELLANEOUS

SK1	28 way ZIF socket
XT1	2.4576MHz crystal

Power Supply (10-18V DC @ 500mA), DIL IC sockets (1 x 40 way, 1 x 14 way and 3 x 28 way), Heatsink for IC21, PCB, Case type MB6, M3 hardware, Connectors for power and RS232, Veropins, Tinned copper wire (24SWG) or through-PCB pins.

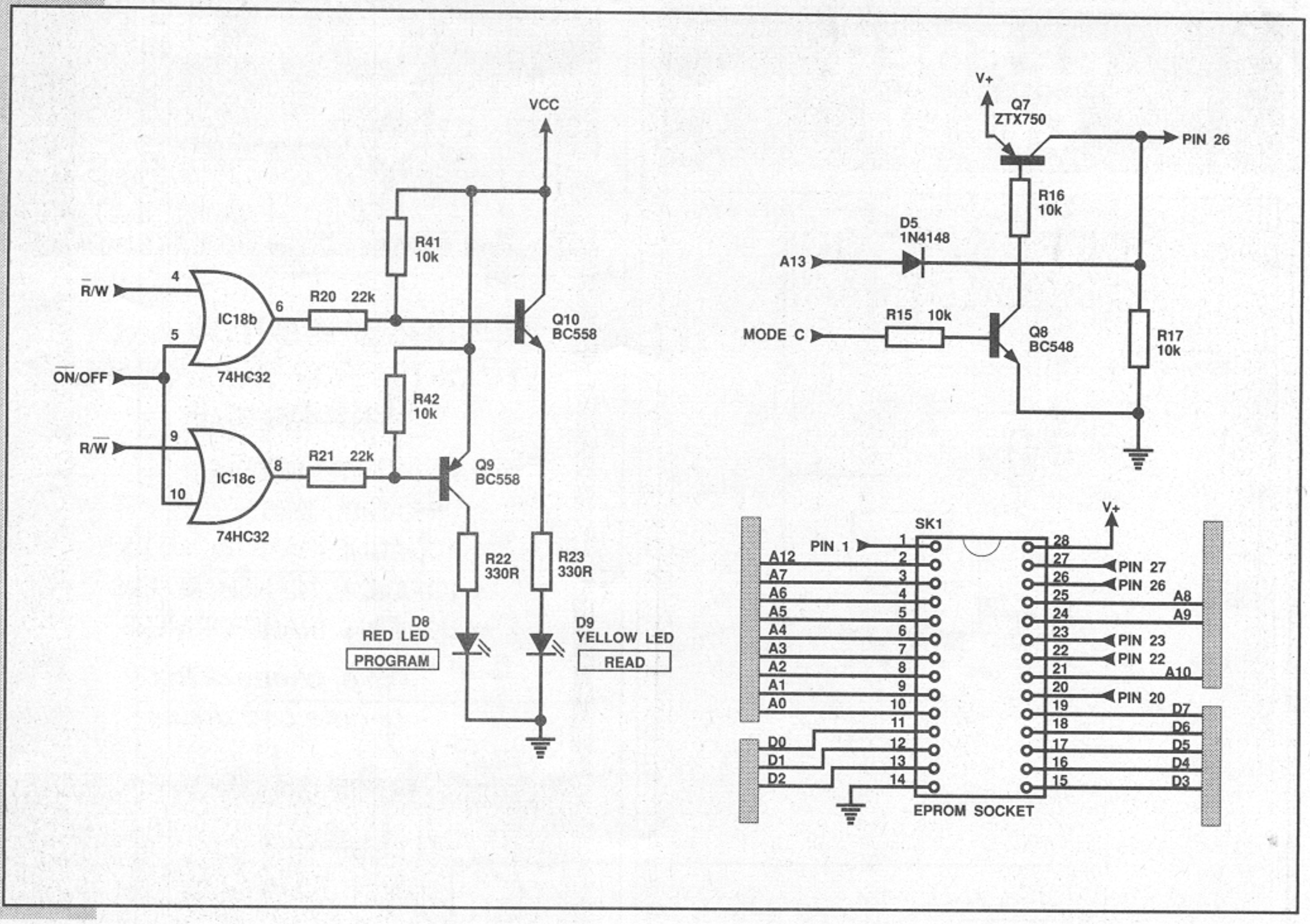
BUYLINES

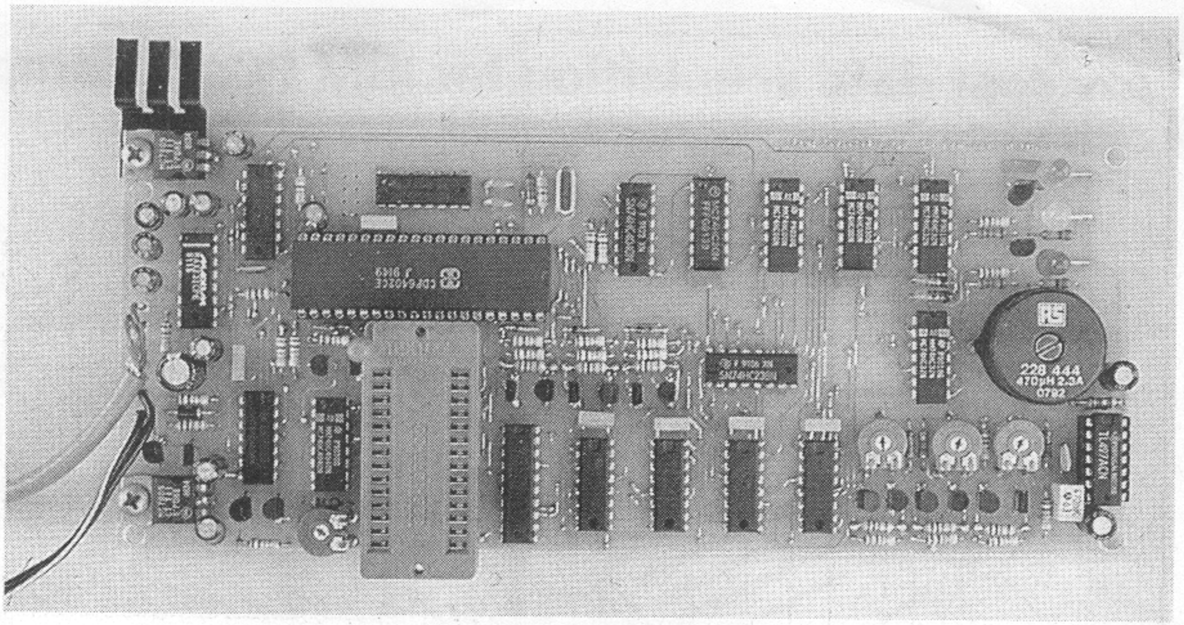
All components are available from Maplin, the majority can probably also be obtained from your usual supplier. Small 0.47R resistors do not appear to be readily available - use two 1R0 components in parallel. The PCB will be available from the ETI PCB service, next month. Before purchasing a power supply, check the latest bargain list from Greenweld (0703 236363), they often list suitable units for about £3.

The software listed in this article, together with a comprehensive menu driven control program and some useful bits and pieces (IBM PC or compatible only) is available from the author at the following address:

Paul Stenning, 1 Chisel Close, Hereford, HR4 9XF. Please send a blank PC formatted disk (3.5" or 5.25"), together with a cheque or postal order for £10, a return address label and adequate return postage (overseas 2 International Reply Coupons). If you do not have a disk send £12 and I will supply one (please specify size). B.A.E.C. members - see newsletter for a special offer!

The author would also be interested to hear from users of other computers, who have either written suitable control software or who are looking for some - he will attempt to put one in touch with the other! Please write with an SAE.


VCC
R41 10k
R/W
IC18b
74HC32
R20 22k
Q10 BC558
ON/OFF
R42 10k
IC18c
74HC32
R21 22k
Q9 BC558
R22 330R
R23 330R
D8 RED LED
PROGRAM
D9 YELLOW LED
READ
V+
Q7 ZTX750
PIN 26
R16 10k
D5 1N4148
A13
R15 10k
MODE C
Q8 BC548
R17 10k
SK1
V+
PIN 1
PIN 27
PIN 26
PIN 23
PIN 22
PIN 20
A12
A7
A6
A5
A4
A3
A2
A1
A0
D0
D1
D2
A8
A9
A10
D7
D6
D5
D4
D3
EPROM SOCKET



EPROM Programmer

2

Paul Stenning continues with his excellent design of EPROM programmer.

The construction of this PCB is rather fiddly and great care should be taken. All the components except the power input socket (SK2) and the RS232 socket (SK3) are mounted on the PCB. This is a double sided board, about 210∞88mm in size, which is available from the ETI PCB service. Note that the holes in the PCB are not plated through. The PCB overlay is shown in Figure 6. Due to the complexity of the PCB, the construction should be carried out in the following order.

Firstly enlarge IC20 and IC21 mounting tab holes, L1 mounting hole and the corner fixing holes to 3mm. Also enlarge the holes for presets RV1-4 to 1.2mm, and the holes for IC20, IC21, D11 and the Veropins to 1.0mm

Next fit the through-board connections in the positions marked with a single small circle on the overlay, there are 122 in total. Tinned copper wire should be used here, suitable pins may be available but check they will fit the holes in the PCB (0.8mm) before ordering. Now fit the transistors, resistor network and non-polarised capacitors. The resistor network must be fitted the correct way round as shown on the overlay. Note that many of the component leads will also need to be soldered on the top of the PCB - wherever there is a pad it should be soldered to. This also applies to the resistors, diodes and presets which can now be fitted. Note that the presets can be fitted on the back of the PCB if required, this may ease adjustment once the PCB is mounted in the case.

Next fit all the DIL IC's except IC3 and IC19. Note that since many connections need to be soldered on the top of the PCB it is not possible to use conventional IC sockets, although some of the more expensive turned pin types may be suitable. IC sockets should now be fitted in positions IC3, IC19 and SK1.

It is now possible to fit the remaining components in any convenient order. Temporarily solder the LEDs at the full length of their leads, and adjust them later when the PCB is being fitted in the case. L1 should be mounted with an M3 nut, bolt and shake-proof washer (do not over-tighten) or a dab of glue. IC20 and IC21 should be mounted with M3 nuts and bolts, IC21 would benefit from a small heatsink or bracket of some sort. Veropins should be fitted for the off-board connections. Fit a wire link in LK1 position, between the lower two homes for 9600 baud, or as shown on the overlay for other rates.

Testing

The PCB should be tested before fitting into the case. Do not fit IC3 or IC19 yet. Connect the unit to a power supply via a test meter set to 500mA DC or greater. Switch on and watch the meter, if the reading exceeds 200mA switch off immediately and find out why! Make a note of this current. If all is well remove the meter and connect the power directly. Now set the meter to 10V DC or thereabouts and check V_{cc} on the power pins of one of the TTL IC's, this should be between 4.75V and 5.25V. Also check for about +9V on pin 2 of IC1 and about -9V on pin 6 of IC1.

If you have a 'scope, look at the DC input and check that the troughs of any ripple do not go below 10V. If there is significant ripple from the power supply (greater than about 1V pk-pk), try connecting a 1000μ/25V capacitor directly across the DC input.

You could now fit the remaining ICs, adjust the voltages, and try the unit in use - and probably get away with it! However I would strongly urge that the following step-by-step checks are carried out to ensure the unit is fully functional. A 'scope or logic probe would be most useful, although most of the checks can be done with just a test meter.

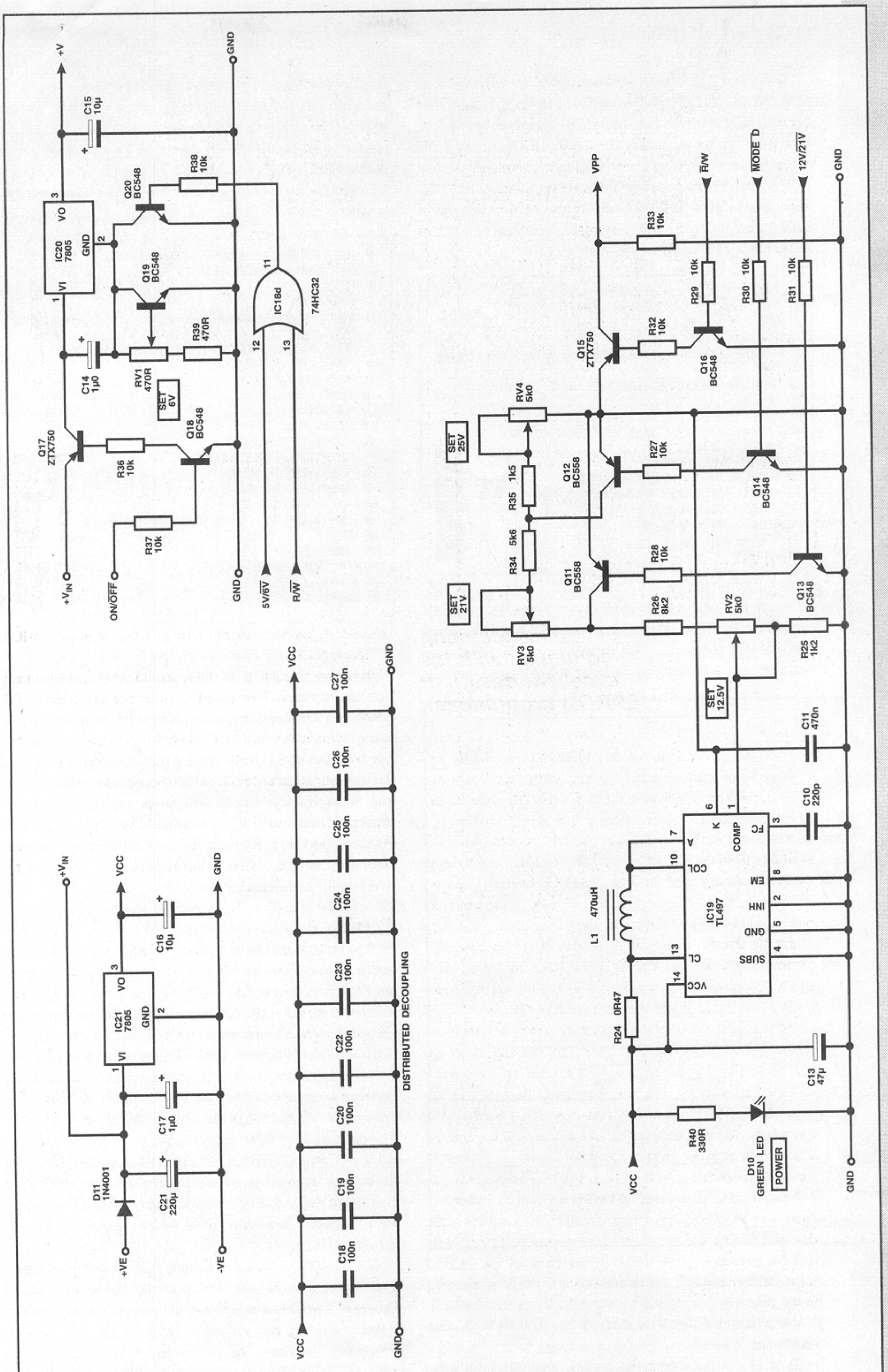


Fig.4 EPROM Programmer Power supplies

If you have an IBM PC or compatible, start BASICA or G.W.BASIC (or QBASIC if you have MS-DOS 5), and enter the test program given in Listing 1 (if you have obtained the disk from the author simply run TEST1.EXE, which is the compiled version). If you have a different computer you may have to modify or re-write the program as necessary, the notes in the 'How it Works - Software' section may be of some help. It may be worth trying to get access to a PC, to avoid having too many unknowns!

```
10  REM *** EPROM Programmer BASICA Test Program 1 Version 1.00
20  REM *** Copyright (C) Paul Stenning and ETI, 1992.
30  REM
40  SCREEN 0 : CLS
50  PRINT "EPROM Programmer BASICA Test Program 1 - (C) Paul Stenning & ETI, 1992"
60  PRINT
70  OPEN "COM1:9600,N,8,1,CS200,CD0,DS0" FOR RANDOM AS #1 LEN = 1
80  INPUT ; A$
90  IF A$ = "" THEN PRINT "QUIT" : CLOSE #1 : END
100 IF LEN(A$) <> 2 THEN PRINT TAB(10); "INPUT ERROR" : GOTO 80
110 PRINT #1, CHR$(VAL("&h" + A$));
120 TIMEOUT = TIMER + 0.1
130 IF EOF(1) AND TIMER < TIMEOUT THEN GOTO 130
140 IF TIMER >= TIMEOUT THEN PRINT TAB(10); "**" : GOTO 80
150 A$ = HEX$(ASC(INPUT$(1, #1)))
160 IF LEN(A$) < 2 THEN A$ = "0" + A$
170 PRINT TAB(10); A$
180 GOTO 80
```

LISTING 1

Insert IC3 (the 6402), connect the programmer to the computers RS232 serial port (see Figure 5), switch it on and then run the software. The software does nothing more exciting than wait for you to enter a 2 digit hex number (followed by <Enter>) and then sends it to the programmer. It then attempts to read back a number, if it's successful it prints the number otherwise it prints **. To exit just press <Enter> on it's own.

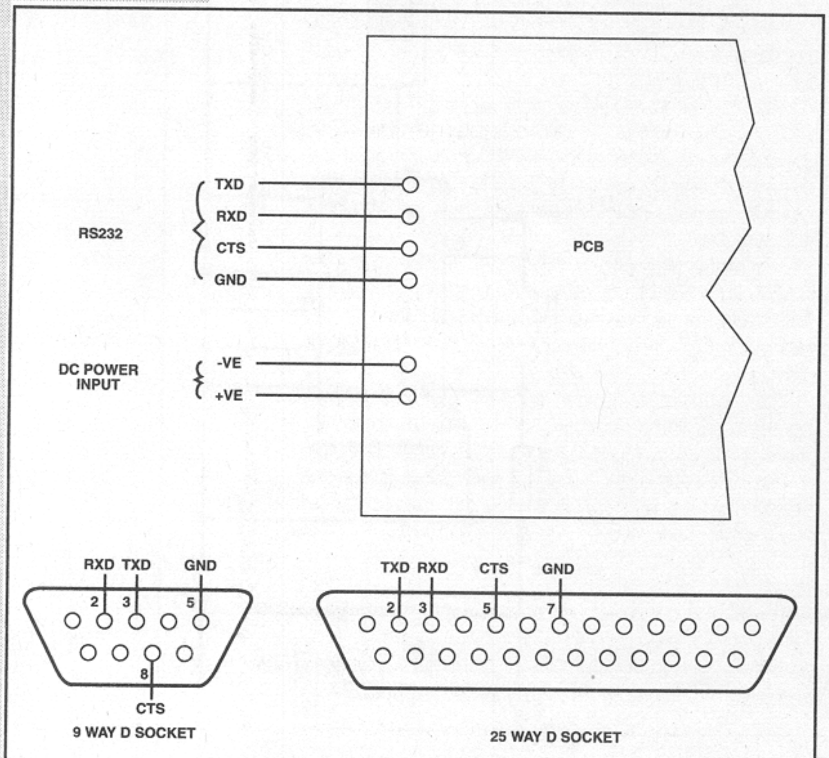


Fig.5 Socket connections

Type 'FF' (don't type the quotes, and follow it with <Enter>). Check the logic levels on pins 5 to 12 of IC3, they should all be logic 1. Note that logic 1 is anything over 3.5V and logic 0 is anything under 0.5V. Now type '00' and the logic levels should all be 0. To be certain, type '55' and the levels should be 01010101, then type 'AA' and the levels should be 10101010. If you have a 'scope or logic probe check for a short positive going pulse on pin 19 whenever a number is sent.

```
10  REM *** EPROM Programmer BASICA Test Program 2 Version 1.00
20  REM *** Copyright (C) Paul Stenning and ETI, 1992.
30  REM
40  SCREEN 0 : CLS
50  PRINT "EPROM Programmer BASICA Test Program 2 - (C) Paul Stenning & ETI, 1992"
60  PRINT
70  OPEN "COM1:9600,N,8,1,CS200,CD0,DS0" FOR RANDOM AS #1 LEN = 1
80  PRINT #1, CHR$(VAL("&h20"));
90  PRINT #1, CHR$(VAL("&h31"));
100 PRINT #1, CHR$(VAL("&h40"));
110 FOR COUNT = 0 TO 65535
120 LOCATE CSRLIN, 1 : PRINT COUNT;
130 IF COUNT = 0 THEN GOSUB 210
140 IF COUNT = 21845 THEN GOSUB 210
150 IF COUNT = 43690 THEN GOSUB 210
160 IF COUNT = 65535 THEN GOSUB 210
170 PRINT #1, CHR$(VAL("&h50"));
180 IF INKEY$ = CHR$(27) THEN PRINT TAB(8); "ABORTED": GOTO 200
190 NEXT
200 CLOSE #1 : END
210 PRINT TAB(8); "Press Any Key to Continue..."
220 IF INKEY$ = "" THEN GOTO 220
230 RETURN
```

LISTING 2

The most likely cause of problems here is the RS232 wiring. Are you using the right port (COM1) on your PC? You can edit line 70 of the program if you are using a port other than COM1. Have you set LK1 to 9600 (or lower if your type of computer won't work at 9600)? Try swapping wires 2 and 3 in the RS232 lead. If the program appears to lock try disconnecting the CTS wire (Ctrl-Break will stop the software in this case). Check the link settings on your serial communications port - if you have the 'Everex EV170 Magic I/O Card' (used in many early XT and 286 AT machines) and can't get it to work, write to the author!

Type '0F' and check pins 9, 10, 15 and 16 of IC6, they should all be at logic 1. Now type '00' and they should all be at logic 0. Typing '05' should give 0101 and '0A' should give 1010. Now repeat the above, replacing the first character with a '1' and checking the levels on IC7, then '2' and IC8, and finally '3' and IC9.

Now type '00', '10', '30' then '70'. After you typed the '70' the screen should show '00', the others should have given '**'. The 'Program' (red) LED should also be on. This set '00' on the data bus, selected write mode to enable U10, then read the data back down the RS232 link. The most likely cause of problems here is the RS232 link again.

Now type '31' then '70'. The screen should show 'FF' and the 'Read' (yellow) LED should be on. IC10 is now disabled so it's outputs are tristate and pulled up by RN1.

Typing '30' then '70' should return '00' again. Typing '0F', '1F' then '70' should return 'FF', typing '05', '15' then '70' should give '55', and typing '0A', '1A' then '70' should give 'AA'. Also check that the appropriate data is actually reaching the EPROM socket pins as shown below:

Data Line	D7	D6	D5	D4	D3	D2	D1	D0
Socket Pin	19	18	17	16	15	13	12	11
Type	**Expected Logic Level**							
'00', '10'	0	0	0	0	0	0	0	0
'05', '15'	0	1	0	1	0	1	0	1
'0A', '1A'	1	0	1	0	1	0	1	0
'0F', '1F'	1	1	1	1	1	1	1	1

HOW IT WORKS

SOFTWARE

In the following section a reasonable understanding of programming in BASIC is assumed. The software was written for Microsoft BASICA, as supplied with Compaq DOS 3.31. It has also been tested with QBASIC supplied with MS-DOS 5 and with Microsoft QuickBASIC V4.5. Users of other BASIC dialects may have to modify the code to suit.

The first test program is shown in Listing 1. Line 70 opens COM1 (the first serial port) at <9600> Baud, <N>o parity checking, <8> data bits, and <1> stop bit. The timeout on CTS (clear to send) is set to 200 milliseconds, CD (carrier detect) and DSR (data set ready) are disabled. Another serial port could be used in place of COM1 if required, by editing this line.

Line 80 accepts an input from the keyboard, the semicolon causes the cursor to remain on the same line after <Enter> is pressed. Line 90 terminates the program if no value is entered. Line 110 converts the entered data from a two character string to a single byte and sends it down the serial port. Note that in BASICA Hex numbers are indicated by preceding them with "&h", hence the value of "&hFF" is 255.

Lines 120 to 170 responsible for waiting up to 0.1 seconds for data to be sent back up the serial port and displaying it. TIMER is a BASICA variable which contains the number of seconds since midnight to 2 decimal places (updated 18.2 times per second), this is used in lines 120 and 130 to control the timeout. EOF(1) will have a value of 0 if data is present, otherwise it will be 1. Line 140 prints "**" if a timeout has occurred, otherwise lines 150 to 170 read the value, convert it from a single byte to a two character string (using the HEX$ function) and print it. Line 180 loops back round for another go!

The second test program, shown in Listing 2, is used to test the address counter system. This clears the counters and then repeatedly increments them, by sending the appropriate codes. The operation should be evident, given the information above.

The main control program is shown in Listing 3. This software is about the minimum required to make sensible use of the programmer. It is written in a manner which should make the functioning relatively easy to understand, and is not intended to be an example of good programming!

The subroutines at lines 7000 to 7060, and 8000 to 8020 fetch a byte from the serial port and send a byte to the serial port respectively. Their operation is as described in the Listing 1 details above. These subroutines are called frequently by the remainder of the program.

Line 100 opens the serial communications as before. Lines 120 to 300 attempt to establish communications with the programmer and test whether or not the CTS connection is present and working. Line 120 sets the program pulse duration to 40 milliseconds, initiates a program pulse immediately followed by a send instruction. If CTS is present the send instruction will not be sent until the program pulse has finished so data will be received, otherwise no data will be sent (see "How it Works -Hardware"). The integer variable PAUSE% is set to 1 if there is no CTS line, causing the software to add suitable delays itself - note that this will slow the operation of the software quite drastically.

Lines 150 to 280 send values to the data latches and then attempt to read them back - this is to establish that communication is reliable.

Lines 310 to 780 request information from the user regarding the EPROM type and programming requirements, whilst lines 790 to 820 set up the programmer accordingly.

Lines 1000 to 1230 form the main menu. Note that CHR$(27) gives the value of the Escape key.

The Read, Program and Verify sections use ASCII-HEX data files in the programmers own format (conversion programs to and from Intel-HEX are given later). The format is easy to produce and edit manually.

The first line is the name of the EPROM type - "2716", "2732" etc. The remaining lines each start with the address in Hex (4 digits), followed by four spaces, followed by 16 bytes in Hex (2 digits) each separated by one space. The addresses must be sequential, starting at 0000. A small section is shown below:

```
2716
0000    00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF
0010    F0 23 DE F4 5A 22 3D 7E EA A2 C0 C0 38 24 AA 00
:                                                    :
:                                                    :
07E0    00 23 48 DE 4A D7 E1 4C 9A 8B BB DE 09 FF FF FF
07F0    FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

It should also be noted that this format is not particularly efficient with disk space - the file for a 27512 will take up about 250K. A file compression utility, such as PKZIP or LHARC, will dramatically reduce the size for storage if disk space is a problem.

With the information that has gone before, the operation of the remaining sections of the software should be fairly apparent.

The section from 2000 to 2300 reads the contents of the EPROM to a file. Lines 2100 and 2110 give a quarter second delay to allow the power supply rails to come up.

The section from 3000 to 3390 programs the EPROM from the contents of a file. Lines 3270 to 3290 add a delay (100 milliseconds) to allow for the programming pulse if the CTS line is not present, this will occur whether the programming pulse is 1 or 40 milliseconds. This delay may be optimised but it would be better to get CTS working in the first place.

The section from 4000 to 4330 verifies (or compares) the contents of the EPROM with a file, whilst the section from 5000 to 5240 checks the device is blank (all locations contain "FF"). The section from 6000 to 6220 allow the programming voltages to be checked.

Other programs are available to convert the EPROM programmer data files to and from standard Intel-HEX data files respectively. This is not the place for an explanation of the Intel-HEX file format, so please just accept that the programs work! Details of Intel-HEX and other standard file formats are on the disk available from the author, together with various conversion programs etc.

Now we come to the address bus. Type in the program given in Listing 2 (save the other program first as it will be needed again).

The program configures the programmer for 27512 EPROM's so all 16 address lines are bought to the EPROM socket and should be checked there. The program clears the address counter and then repeatedly increments the count, pausing at selected points to enable the checks to be made, as shown below:

Addr Line	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Socket Pin	1	27	26	2	23	21	24	25	3	4	5	6	7	8	9	10
Count	Expected Logic Level															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21845	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
43690	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
65535	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

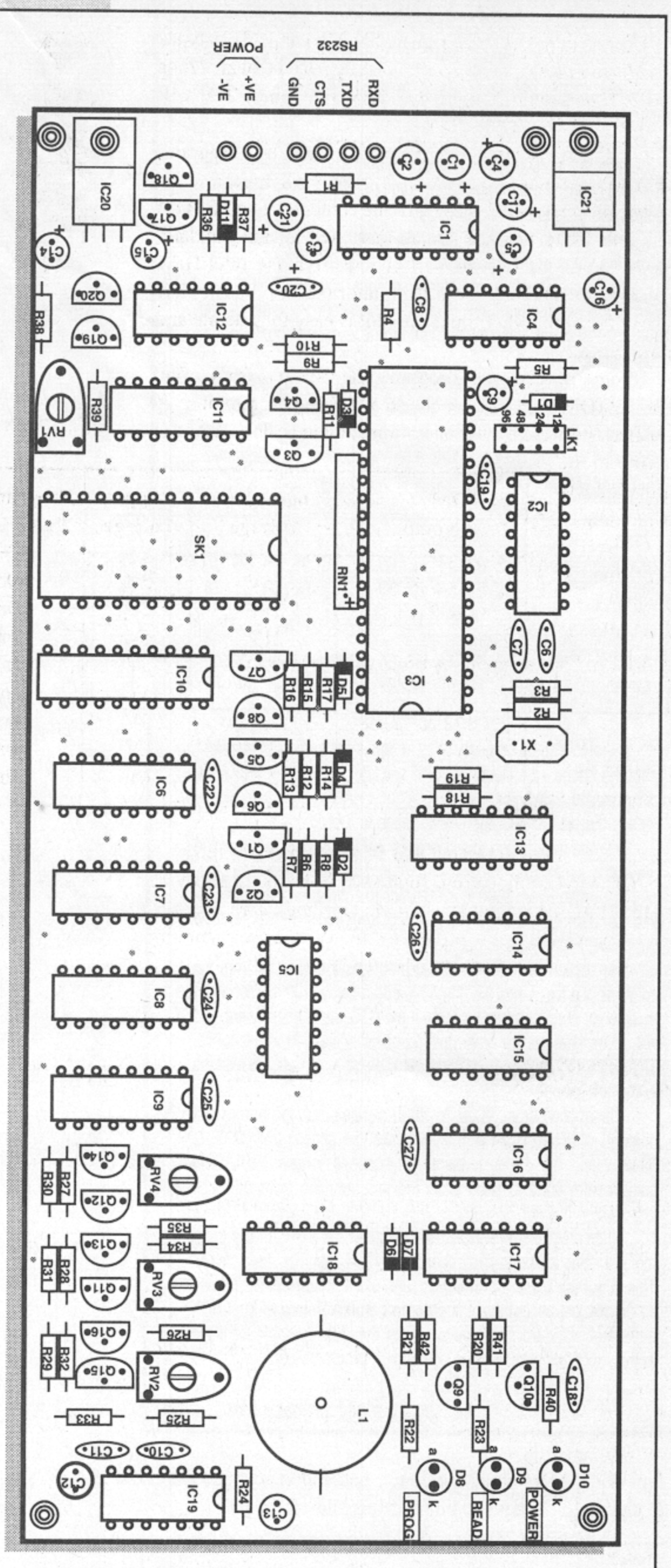


Fig.6 Component Overlay

The logic levels should be checked on the pins of the EPROM socket when the program pauses, if a level is incorrect check on the appropriate pin on IC11 or IC12, then trace the fault as necessary. Note that the program may run quite slowly. A complied version (TEST2.EXE) which runs considerably faster is on the disk available from the author.

Now re-load the first test program. Connect a 'scope or logic probe to IC15 pin 11. Type '30'. Now when you type '60' you should observe a 40 millisecond positive going pulse. Move the probe to pin 8 of IC15 and the pulse should be negative going. Now type '32' and repeat the above checks, the pulse should now be 1 millisecond. With a logic probe you will probably only be able to detect the presence of the pulse and will have to assume it is the correct length, with a test meter you probably won't be able to see anything!

Now switch off and insert IC19. Connect a test meter set to about 500mA DC in line with the power input and switch back on. If the current is more than 100mA greater than it was before switch off and find out why! The most likely cause is a short circuit on VPP somewhere. If all is well remove the meter and connect the power directly. Set all four presets to the centre position.

Set a test meter to the 10V DC range and connect between pins 28 (+ve) and 14 (-ve) of the EPROM socket. Type '23' then '38', the meter should read 5V +/-0.25V. Now type '30' and the reading should rise, adjust RV1 for a reading of 6.1V +/-0.1V. Type '80' and the voltage should drop to zero. Set the meter to the 30V DC range and transfer the +ve meter probe to pin 1 of the EPROM socket. Type '34' and then adjust RV2 for a reading of 12.6V +/-0.1V. Type '30' and adjust RV3 for 21V +/-0.25V. Type '2F' and adjust RV4 for 25V +/-0.25V. Type '31' and the voltage should drop to zero.

The only thing left to check now is the various configurations for the different types of EPROM's. As described in 'How it Works', the functions of six of the EPROM socket pins vary depending upon the type of EPROM. The address lines have already been checked at the EPROM socket, as has the programming voltage to pin 1. The checking of the remaining combinations is detailed below.

Type '40' to clear the address counters. Connect a 'scope or logic probe to pin 20 of the EPROM socket. Type '30' then '2F'. A 40 millisecond positive going pulse should be observed when '60' is typed. Type '20', and the pulse when typing '60' should now be negative going. Now type '23' and the line should remain at logic 0 when '60' is typed. Move the probe to pin 22. Type '2F' then '31' and the line should be at logic 0. Type '30' and it should go to logic 1. Now type '20' and it should rise to 21V. Move the probe to pin 23 which should be at logic 0. Type '2F' and the line should rise to 25V. Move the probe to pin 26, which should be at 6V. Type '20' and it should go to logic 0. Finally move the probe to pin 27 and type '23'. A 40 millisecond negative going pulse should be seen when you type '60'.

If you have reached the end of all this successfully you can be confident that your EPROM programmer is 100% functional!

The Case.

The prototype was mounted in a plastic case (type MB6) having external dimensions of 220∞150∞64mm. The removable panel is considered to be the bottom, and may be fitted with self-adhesive feet if required. The top surface needs cut-outs for the EPROM ZIF (Zero Insertion Force) socket and the LEDs, as well as four fixing holes for the PCB. You may also wish to make four small holes to enable adjustment of the presets.

The rectangular cut-out for the ZIF socket may be made by drilling a line of shall holes around the edge then breaking out the centre part and filing to shape. Take care not to file the hole too large or the result will look untidy! The socket is raised above the PCB by stacking up a number of 28 pin DIL IC sockets, three were used on the prototype. If the result feels insecure, the sockets may be held together with a suitable adhesive.

Position the PCB and mark the positions of the four fixing holes and then measure the positions of the three LED holes. The first LED is 4mm down and 6mm to the left of the top right fixing hole (view from outside the box), the other two are spaced below at 9mm intervals. The fixing holes are 3mm in diameter whilst the LED holes are 5mm. Also drill suitable holes in the rear of the case for the DC input socket and the RS232 cable or connector. On the prototype a 3.5mm jack socket was used for power (since this matched the plug on the PSU), and the RS232 cable passed through a hole fitted with a grommet. Choose connectors that are not likely to come unplugged accidentally! The case may now be marked with rub-down transfers or similar if required.

Solder suitable lengths of wire to the PCB for the off-board connections and insert the LEDs through the holes in the PCB (do not solder yet). Mount the PCB in the case using M3 screws, nuts and spacers, then position the LEDs so that they slightly protrude through the holes and solder them into place. Complete the interwiring (see Fig 5) and assemble the case. If an additional smoothing capacitor was found necessary whilst testing, this may be mounted across the pins of the DC input socket, or on the rear of the PCB in parallel with C21.

In Use.

The control software is shown in Listing 3 and is suitable for an IBM PC or compatible machine running BASICA, G.W.BASIC or QBASIC. This software is about the minimum required to make sensible use of the programmer. The functioning of the software is described in the "How it Works - Software" section.

If BASICA or G.W.BASIC is being used, the program will run fairly slowly. This is a limitation of interpreted BASIC. QBASIC supplied with MS-DOS 5 is a much more advanced product and a good deal better in this respect.

Additional (faster) software is supplied on the disk available from the author, see Buylines.

An EPROM must not be inserted or removed if the 'Program' or 'Read' LED is lit, or if the programmer is configured for a different type of device. 24 pin EPROMs must be fitted in the lower pins (3-26) of the socket. In all cases pin 1 is upwards. Failure to observe the above may result in damage to the EPROM or (less likely) the programmer.

The programmer should be switched on and connected to COM1 (RS232 serial port 1) on the computer. Start the software and the 'Program' LED will light. Once successful communication has been established the program will request information about the type of EPROM and the programming method required. See the table below or consult the manufacturers data book.

Some 2764 and 27128 types require a complex arrangement of programming pulses however a single 1ms pulse will usually suffice. If in doubt or if problems are experienced use 40ms. Although some 2716 and 2732 devices will program successfully with a 1ms programming pulse, this is not recommended for final EPROMs, but may prove useful when testing software etc.

The use of the 'A' suffix on 12.5V 2764 and 27128 types appears to be less than standard, it is suggested that all 2764s and 27128s should be tried on 12.5V first, since 21V will destroy a 12.5V device.

Once these selections have been made the 'Program' LED will extinguish and the main menu will appear.

'Read' (menu option 1) reads the contents of an EPROM to a file. Note that the file format used is non-standard, however programs to convert to and from the Intel-HEX standard are available from the author (see buylines). The advantage of the file format is that it is easy to generate and edit manually.

'Program' (option 2) programs the EPROM from a file. The EPROM is not blank checked before programming or verified afterwards, these operations should be done from the main menu individually if required.

'Verify' (option 3) compares the contents of the EPROM with a file, and 'Blank Check' (option 4) does as it's name suggests! Both these options report the number of locations that failed.

EPROM Type Number	Programming Voltage	Supply Voltage	Programming Pulse Length
2716 & 27C16	25V	5V	40ms
2732 & 27C32	21V	5V	40ms
2764	21V	5V	1ms
27C64 & 2764A	12.5V	5V	1ms
27128	21V	5V	1ms
27C128 & 27128A	12.5V	5V	1ms
27256 & 27C256	12.5V	6V	1ms
27512 & 27C512	12.5V	6V	1ms

'Change Configuration' (option 5) re-starts the software so the EPROM type and programming method can be changed.

'Adjust Voltages' (option 6) allows the programming voltages to be checked and adjusted if required.

Happy programming!

LISTING 3

```
10  REM *** EPROM Programmer BASICA Control Software Version 1.00
20  REM *** Copyright (C) Paul Stenning and ETI, 1992.
30  REM
40  SCREEN 0 : CLS
50  PRINT "EPROM Programmer BASICA Control Software  -  (C) Paul Stenning & ETI, 1992"
60  PRINT : PRINT "Ensure EPROM Socket is Empty, then press any key..."
70  K$ = INKEY$ : IF K$ = "" THEN GOTO 70
80  IF K$ = CHR$(27) THEN PRINT : PRINT "Quit" : GOTO 10000
90  PRINT "Establishing Communication ";
100 OPEN "COM1:9600,N,8,1,CS200,CD0,DS0" FOR RANDOM AS #1 LEN = 1
110 PRINT ". ";
120 S$ = "30" : GOSUB 8000 : S$ = "60" : GOSUB 8000 : S$ = "70" : GOSUB 8000 : GOSUB 7000
130 IF F$ = "***" THEN PAUSE% = 1 ELSE PAUSE% = 0
140 PRINT ". ";
150 S$ = "05" : GOSUB 8000 : S$ = "15" : GOSUB 8000 : S$ = "60" : GOSUB 8000
160 IF PAUSE% = 0 THEN GOTO 190
170 TIME = TIMER + 0.1
180 IF TIMER < TIME GOTO 180
190 S$ = "70" : GOSUB 8000
200 GOSUB 7000
210 IF F$ <> "55" THEN GOTO 9000
220 PRINT ". ";
230 S$ = "0A" : GOSUB 8000 : S$ = "1A" : GOSUB 8000 : S$ = "60" : GOSUB 8000
240 IF PAUSE% = 0 THEN GOTO 270
250 TIME = TIMER + 0.1
260 IF TIMER < TIME GOTO 260
270 S$ = "70" : GOSUB 8000 : GOSUB 7000
280 IF F$ <> "AA" THEN GOTO 9000
290 PRINT ". ";
300 IF PAUSE% = 0 THEN PRINT "Ok." ELSE PRINT "No CTS Line -Software Delay Used."
310 PRINT
320 PRINT "   1 - 2716"
```

```
330 PRINT "  2 - 2732"
340 PRINT "  3 - 2764"
350 PRINT "  4 - 27128"
360 PRINT "  5 - 27256"
370 PRINT "  6 - 27512"
380 PRINT "Select EPROM Type: ";
390 K$ = INKEY$
400 IF K$ = CHR$(27) THEN PRINT "Quit" : GOTO 10000
410 IF K$ = "1" THEN TYPENAME$ = "2716" : TYPECODE$ = "2F" : MAXADDR = 2047 : GOTO 480
420 IF K$ = "2" THEN TYPENAME$ = "2732" : TYPECODE$ = "25" : MAXADDR = 4095 : GOTO 480
430 IF K$ = "3" THEN TYPENAME$ = "2764" : TYPECODE$ = "23" : MAXADDR = 8191 : GOTO 480
440 IF K$ = "4" THEN TYPENAME$ = "27128" : TYPECODE$ = "23" : MAXADDR = 16383 : GOTO 480
450 IF K$ = "5" THEN TYPENAME$ = "27256" : TYPECODE$ = "21" : MAXADDR = 32767 : GOTO 480
460 IF K$ = "6" THEN TYPENAME$ = "27512" : TYPECODE$ = "20" : MAXADDR = 65535 : GOTO 480
470 GOTO 390
480 PRINT TYPENAME$
490 PRINT
500 PRINT "  1 - 12.5 Volts"
510 PRINT "  2 - 21/25 Volts"
520 PRINT "Select EPROM Programming Voltage: ";
530 K$ = INKEY$
540 IF K$ = CHR$(27) THEN PRINT "Quit" : GOTO 10000
550 IF K$ = "1" THEN PROGVOLTNAME$ = "12.5 Volts" : STATUS% = 4 : GOTO 580
560 IF K$ = "2" THEN PROGVOLTNAME$ = "21/25 Volts" : STATUS% = 0 : GOTO 580
570 GOTO 530
580 PRINT PROGVOLTNAME$
590 PRINT
600 PRINT "  1 - 5 Volts"
610 PRINT "  2 - 6 Volts"
620 PRINT "Select EPROM Supply Voltage: ";
630 K$ = INKEY$
640 IF K$ = CHR$(27) THEN PRINT "Quit" : GOTO 10000
650 IF K$ = "1" THEN SUPPVOLTNAME$ = "5 Volts" : STATUS% = STATUS% + 8 : GOTO 680
660 IF K$ = "2" THEN SUPPVOLTNAME$ = "6 Volts" : STATUS% = STATUS% + 0 : GOTO 680
670 GOTO 630
680 PRINT SUPPVOLTNAME$
690 PRINT
700 PRINT "  1 - 1 MilliSecond"
710 PRINT "  2 - 40 MilliSeconds"
720 PRINT "Select EPROM Program Pulse Duration: ";
730 K$ = INKEY$
740 IF K$ = CHR$(27) THEN PRINT "Quit" : GOTO 10000
750 IF K$ = "1" THEN PROGPULSENAME$ = "1 MilliSecond" : STATUS% = STATUS% + 2 : GOTO
780
760 IF K$ = "2" THEN PROGPULSENAME$ = "40 MilliSeconds" : STATUS% = STATUS% + 0 : GOTO
780
770 GOTO 730
780 PRINT PROGPULSENAME$;
790 S$ = TYPECODE$ : GOSUB 8000
800 S$ = "3" + HEX$(STATUS% + 1) : GOSUB 8000
810 S$ = "00" : GOSUB 8000 : S$ = "10" : GOSUB 8000
820 S$ = "40" : GOSUB 8000 : S$ = "80" : GOSUB 8000

1000 REM *** Main Menu
1010 CLS
1020 PRINT "EPROM Programmer BASICA Control Software  -  (C) Paul Stenning & ETI, 1992"
1030 PRINT "Type "; TYPENAME$; "  Program "; PROGVOLTNAME$;
1040 PRINT "  Supply "; SUPPVOLTNAME$; "  Pulse "; PROGPULSENAME$
1050 PRINT : PRINT "MAIN MENU"
1060 PRINT "~~~~~~~~~"
1070 PRINT "  1 - Read"
1080 PRINT "  2 - Program"
1090 PRINT "  3 - Verify"
1100 PRINT "  4 - Blank Check"
1110 PRINT "  5 - Change Configuration"
1120 PRINT "  6 - Adjust Voltages"
1130 PRINT " ESC - Quit"
1140 PRINT : PRINT "Select Option Required: ";
1150 K$ = INKEY$
1160 IF K$ = "1" THEN PRINT "Read" : GOTO 2000
1170 IF K$ = "2" THEN PRINT "Program" : GOTO 3000
1180 IF K$ = "3" THEN PRINT "Verify" : GOTO 4000
1190 IF K$ = "4" THEN PRINT "Blank Check" : GOTO 5000
1200 IF K$ = "5" THEN RUN
1210 IF K$ = "6" THEN PRINT "Adjust Voltages" : GOTO 6000
1220 IF K$ = CHR$(27) THEN PRINT "Quit" : GOTO 10000
1230 GOTO 1150

2000 REM *** Read EPROM to File
2010 PRINT : PRINT "Insert EPROM to Read, then press any key (ESC to Abort)"
2020 K$ = INKEY$ : IF K$ = "" THEN GOTO 2020
2030 IF K$ = CHR$(27) THEN GOTO 1000
2040 PRINT
2050 INPUT "Output File Name "; FILE$
2060 OPEN FILE$ FOR OUTPUT AS #2
2070 PRINT
2080 PRINT #2, TYPENAME$
2090 S$ = "40" : GOSUB 8000
2100 TIME = TIMER + 0.25
2110 IF TIMER < TIME THEN GOTO 2110
2120 FOR ADDR = 0 TO (MAXADDR - 15) STEP 16
2130 IF INKEY$ = CHR$(27) THEN BEEP : PRINT : PRINT "ABORTED!" : GOTO 2260
2140 ADDR$ = HEX$(ADDR)
2150 IF LEN(ADDR$) < 4 THEN ADDR$ = "0" + ADDR$ : GOTO 2150
2160 PRINT #2, ADDR$; TAB(9);
2170 FOR COUNT% = 0 TO 15
2180 S$ = "70" : GOSUB 8000 : GOSUB 7000
2190 PRINT #2, F$; " ";
2200 S$ = "50" : GOSUB 8000
2210 LOCATE CSRLIN, 1
2220 PRINT "Reading Location"; ADDR + COUNT%; "of"; MAXADDR;
2230 NEXT
2240 PRINT #2,
2250 NEXT
2260 S$ = "40" : GOSUB 8000 : S$ = "80" : GOSUB 8000
2270 CLOSE #2
2280 PRINT : PRINT "Press any key to continue...";
2290 IF INKEY$ = "" THEN GOTO 2290
2300 GOTO 1000

3000 REM *** Program EPROM from File
3010 PRINT
3020 PRINT "Insert EPROM to Program, then press any key (ESC to Abort)"
3030 K$ = INKEY$ : IF K$ = "" THEN GOTO 3030
3040 IF K$ = CHR$(27) THEN GOTO 1000
3050 PRINT
3060 INPUT "Input File Name ";FILE$
3070 OPEN FILE$ FOR INPUT AS #2
3080 LINE INPUT #2, DAT$
3090 IF DAT$ <> TYPENAME$ THEN : BEEP : PRINT "FILE DOES NOT MATCH EPROM TYPE" :
GOTO 3340
3100 S$ = "3" + HEX$(STATUS%): GOSUB 8000
3110 S$ = "40" : GOSUB 8000
3120 TIME = TIMER + 0.25
3130 IF TIMER < TIME THEN GOTO 3130
3140 PRINT
3150 FOR ADDR = 0 TO (MAXADDR - 15) STEP 16
3160 IF INKEY$ = CHR$(27) THEN BEEP : PRINT : PRINT "ABORTED!" : GOTO 3340
3170 LINE INPUT #2, DAT$
3180 DAT = VAL("&h" + LEFT$(DAT$, 4))
3190 IF DAT < 0 THEN DAT = DAT + 65536
3200 IF DAT <> ADDR THEN BEEP : PRINT : PRINT "FILE ADDRESS ERROR" : GOTO 3340
3210 FOR COUNT% = 0 TO 15
3220 LOCATE CSRLIN, 1
3230 PRINT "Programming Location"; ADDR + COUNT%; "of"; MAXADDR;
3240 S$ = "0" + MID$(DAT$, 10 + COUNT% * 3, 1) : GOSUB 8000
3250 S$ = "1" + MID$(DAT$, 9 + COUNT% * 3, 1) : GOSUB 8000
3260 S$ = "60" : GOSUB 8000
3270 IF PAUSE% = 0 THEN GOTO 3300
3280 TIME = TIMER + 0.1
3290 IF TIMER < TIME THEN GOTO 3290
3300 S$ = "50" : GOSUB 8000
3310 NEXT
3320 NEXT
3330 PRINT
3340 S$ = "3" + HEX$(STATUS% + 1) : GOSUB 8000
3350 S$ = "40" : GOSUB 8000 : S$ = "80" : GOSUB 8000
3360 CLOSE #2
3370 PRINT : PRINT "Press any key to continue...";
3380 IF INKEY$ = "" THEN GOTO 3380
3390 GOTO 1000

4000 REM *** Verify EPROM with File
4010 PRINT : PRINT "Insert EPROM to Verify, then press any key (ESC to Abort)"
4020 K$ = INKEY$ : IF K$ = "" THEN GOTO 4020
4030 IF K$ = CHR$(27) THEN GOTO 1000
4040 PRINT : INPUT "Input File Name "; FILE$
4050 OPEN FILE$ FOR INPUT AS #2
4060 PRINT
4070 LINE INPUT #2, DAT$
4080 IF DAT$ <> TYPENAME$ THEN BEEP : PRINT "FILE DOES NOT MATCH EPROM TYPE" : GOTO
4300
4090 S$ = "40" : GOSUB 8000
4100 TIME = TIMER + 0.25
4110 IF TIMER < TIME THEN GOTO 4110
4120 FAIL = 0
4130 FOR ADDR = 0 TO (MAXADDR - 15) STEP 16
4140 IF INKEY$ = CHR$(27) THEN BEEP : PRINT : PRINT "ABORTED!" : GOTO 4300
```

```
4150 LINE INPUT #2, DAT$
4160 DAT = VAL("&h" + LEFT$(DAT$, 4))
4170 IF DAT < 0 THEN DAT = DAT + 65536
4180 IF DAT <> ADDR THEN BEEP : PRINT : PRINT "FILE ADDRESS ERROR" : GOTO 4300
4190 FOR COUNT% = 0 TO 15
4200 LOCATE CSRLIN, 1
4210 PRINT "Verifying Location"; ADDR + COUNT%; "of"; MAXADDR;
4220 S$ = "70" : GOSUB 8000 : GOSUB 7000
4230 IF F$ <> (MID$(DAT$, 9 + COUNT% * 3, 2)) THEN FAIL = FAIL + 1
4240 S$ = "50" : GOSUB 8000
4250 NEXT
4260 NEXT
4270 PRINT : PRINT
4280 IF FAIL = 0 THEN PRINT "Verified Ok"
4290 IF FAIL <> 0 THEN BEEP : PRINT "Verify Failed on"; FAIL; "Locations"
4300 S$ = "40" : GOSUB 8000 : S$ = "80" : GOSUB 8000
4310 CLOSE #2 : PRINT : PRINT "Press any key to continue...";
4320 IF INKEY$ = "" THEN GOTO 4320
4330 GOTO 1000

5000 REM *** Blank Check EPROM
5010 PRINT : PRINT "Insert EPROM to Blank Check, then press any key (ESC to Abort)"
5020 K$ = INKEY$ : IF K$ = "" THEN GOTO 5020
5030 IF K$ = CHR$(27) THEN GOTO 1000
5040 PRINT
5050 S$ = "40" : GOSUB 8000
5060 TIME = TIMER + 0.25
5070 IF TIMER < TIME THEN GOTO 5070
5080 FAIL = 0
5090 FOR ADDR = 0 TO MAXADDR
5100 LOCATE CSRLIN, 1
5110 PRINT "Checking Location"; ADDR; "of"; MAXADDR;
5120 S$ = "70" : GOSUB 8000
5130 GOSUB 7000
5140 IF F$ <> "FF" THEN FAIL = FAIL + 1
5150 S$ = "50" : GOSUB 8000
5160 IF INKEY$ = CHR$(27) THEN BEEP : PRINT : PRINT "ABORTED!" : GOTO 5210
5170 NEXT
5180 PRINT : PRINT
5190 IF FAIL = 0 THEN PRINT "Blank EPROM"
5200 IF FAIL <> 0 THEN BEEP : PRINT "EPROM NOT BLANK - Failed on"; FAIL; "Locations"
5210 S$ = "40" : GOSUB 8000 : S$ = "80" : GOSUB 8000
5220 PRINT : PRINT "Press any key to continue...";
5230 IF INKEY$ = "" THEN GOTO 5230
5240 GOTO 1000

6000 REM *** Adjust Programming Voltages
6010 PRINT : PRINT "Ensure EPROM Socket is Empty, then press any key (ESC to Abort)"
6020 K$ = INKEY$ : IF K$ = "" THEN GOTO 6020
6030 IF K$ = CHR$(27) THEN GOTO 1000
6040 PRINT
6050 S$ = "20" : GOSUB 8000 : S$ = "30" : GOSUB 8000
6060 PRINT "Connect Test Meter Between pins 28 (+ve) and 14 (-ve) of EPROM Socket"
6070 PRINT "Adjust RV1 for Reading of 6.1V  (+/- 0.1V), then press any key..."
6080 IF INKEY$ = "" THEN GOTO 6080
6090 PRINT
6100 S$ = "21" : GOSUB 8000 : S$ = "34" : GOSUB 8000
6110 PRINT "Connect Test Meter Between pins 1 (+ve) and 14 (-ve) of EPROM Socket"
6120 PRINT "Adjust RV2 for Reading of 12.6V (+/- 0.1V), then press any key..."
6130 IF INKEY$ = "" THEN GOTO 6130
6140 S$ = "30" : GOSUB 8000
6150 PRINT "Adjust RV3 for Reading of 21V  (+/- 0.25V), then press any key..."
6160 IF INKEY$ = "" THEN GOTO 6160
6170 S$ = "2F" : GOSUB 8000
6180 PRINT "Adjust RV4 for Reading of 25V  (+/- 0.25V), then press any key...";
6190 IF INKEY$ = "" THEN GOTO 6190
6200 S$ = TYPECODE$ : GOSUB 8000 : S$ = "3" + HEX$(STATUS% + 1) : GOSUB 8000
6210 S$ = "80" : GOSUB 8000
6220 GOTO 1000

7000 REM *** Fetch Byte from Programmer
7010 TIMEOUT = TIMER + 0.1
7020 IF EOF(1) AND TIMER < TIMEOUT THEN GOTO 7020
7030 IF TIMER >= TIMEOUT THEN F$ = "**" : RETURN
7040 F$ = HEX$(ASC(INPUT$(1, #1)))
7050 IF LEN(F$) < 2 THEN F$ = "0" + F$
7060 RETURN

8000 REM *** Send Byte to Programmer
8010 PRINT #1, CHR$(VAL("&h" + S$));
8020 RETURN

9000 REM *** Communication Error Message
9010 PRINT ". ERROR COMMUNICATING WITH PROGRAMMER"
9020 BEEP
9030 GOTO 10000

10000  REM *** End Program
10010  CLOSE
10020  PRINT : PRINT
10030  SYSTEM
```

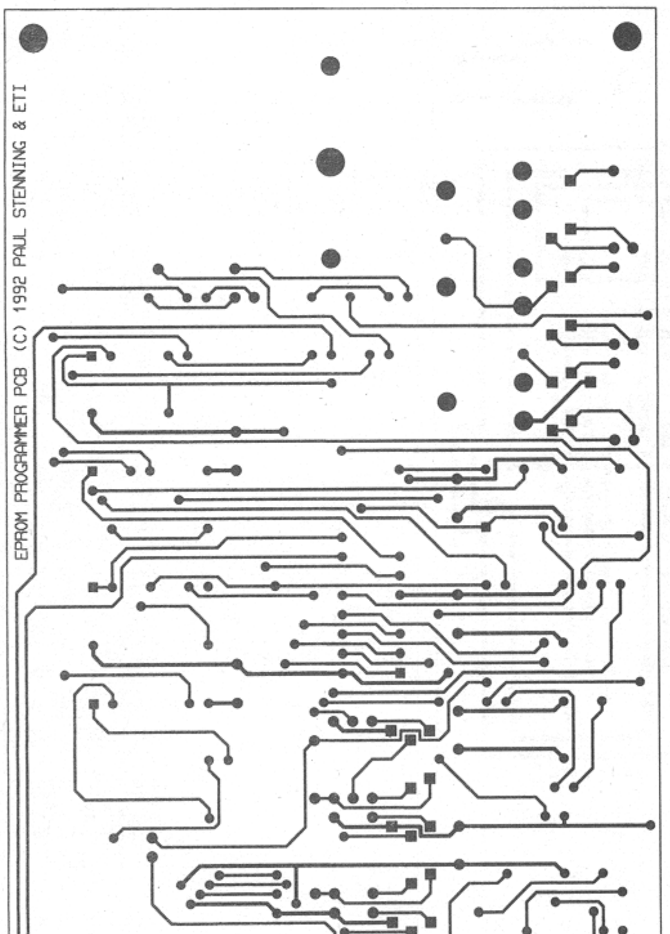
BUYLINES

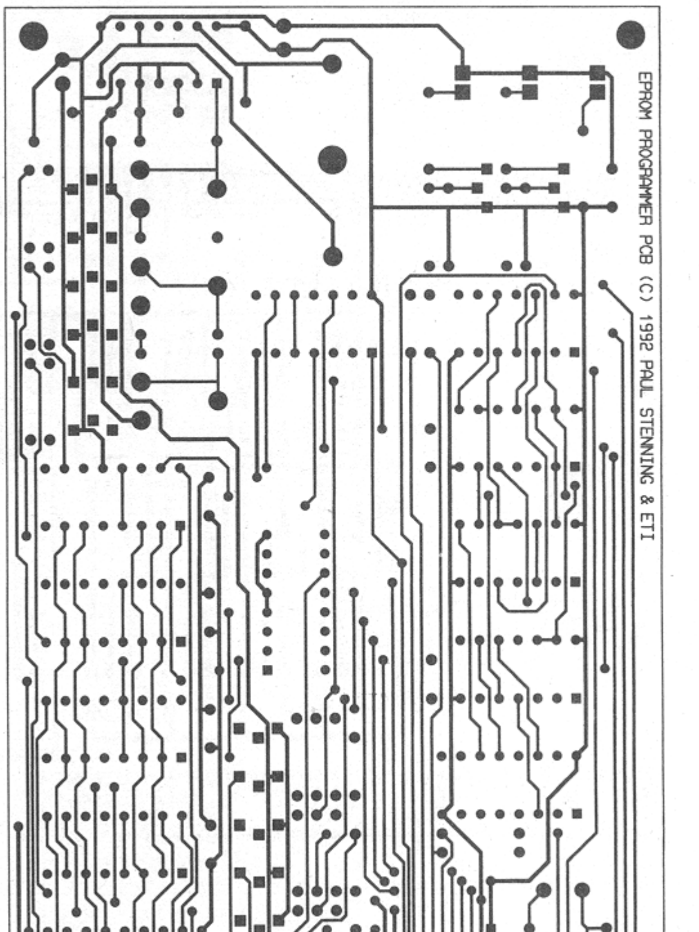
All components are available from Maplin, the majority can probably also be obtained from your usual supplier. Small 0.47R resistors do not appear to be readily available - use two 1R0 components in parallel. The PCB is available from the ETI PCB service. Before purchasing a power supply, check the latest bargain list from Greenweld (0703 236363), they often list suitable units for about £3.

The software listed in this article, together with a comprehensive menu driven control program and some useful bits and pieces (IBM PC or compatible only) is available from the author at the following address:

Paul Stenning, [redacted] Please send a blank PC formatted disk (3.5" or 5.25"), together with a cheque or postal order for £10, a return address label and adequate return postage (overseas 2 International Reply Coupons). If you do not have a disk send £12 and I will supply one (please specify size). B.A.E.C. members - see newsletter for a special offer!

The author would also be interested to hear from users of other computers, who have either written suitable control software or who are looking for some - he will attempt to put one in touch with the other! Please write with an SAE.





Top Side

Solder Side

EPROM Programmer