

# **Eprom Programmer Mark 1**

*Originally published by Paul Stenning in ETI, January and February 1993*

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 U21 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 (downloadable as a zip file from this website), containing these programs, as well as comprehensive menu driven control software and a few useful utilities.

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.

## **Circuit Operation**

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).

U2 and surrounding components generate the Baud Rate clock signal (CLOCK1) for U3, with the rate selectable by LK1. The CLOCK2 signal is used to produce the programming timing pulses, see

later. U3 is a CDP6402 (Maplin QQ04E) "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. U3 is reset by U4:A and associated components.

When serial data is received on pin 20 of U3, 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 U3 (SEND) is pulsed low, data on pins 26-33 (D0-D7) is transmitted in serial form on pin 25.

U1 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", U5. The STROBE signal is taken to one of the Enable lines on U5, 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 U5 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", U6-U9. The Latch Enable lines on these IC's are connected to 4 of the outputs of U5 (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 U6. Similarly sending 35h will store 5h (0101b) in U9

Latch U6 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 U7 holds the most significant nibble. U8 holds the EPROM type information (see later), and U9 holds 4 setup bits. Bit 1 of U9 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 U6 and U7 are taken to a tri-state buffer (U10), which in turn drives the EPROM data lines. U10 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 (U3), and the Y7 output from U5 is connected to pin 23 of U3 (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 U11 and U12. These are connected to lines Y4 and Y5 from U5, 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 U5 (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 52.1uS. This is fed to counter U13, which is normally held reset by the Set/Reset flip-flop built from U15:C and U15:D. 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 U14:A will be high and pins 12 and 13 of U14:B will be low. When the count reaches 19 ( $52.1\mu\text{S} \times 19 = 989.9\mu\text{S}$ ), all the inputs of U14:A will be high, it's output will therefore go low, changing the state of the flip-flop again, via U16:B, 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 U13 counts 768 CLOCK2 pulses ( $52.1\mu\text{S} \times 768 = 40.01\text{mS}$ ).

The PROG PULSE and STROBE signals are coupled via U18:A and U1 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 U3 by U16:A. 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 U16:A stopping the SEND pulse reaching U3. 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 U9 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      MODE LINE      SETUP  
TYPE  A  B  C  C- D  D-  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  
-----
```

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 U17:B pin 6 remains high, U16:C pin 8 follows PROG PULSE, as does U17:C 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 U15:B 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. U16:C pin 8 will remain low and U17:B pin 6 will follow PROG PULSE. U17:C 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 U15:B 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. U16:C pin 8 remains low and U17:B pin 6 remains high, therefore

U17:C pin 8 remains low.

PIN 22 - MODE A and MODE C- are both high, so U15:B 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. U17:D 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 U16:D pin 11 will remain low. U17:D 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 U15:B 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. U18:B pin 6 goes low when R-/W and ON-/OFF are both low, lighting D9 (Read) via Q10, whilst U18:C 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.

U21 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.

U20 supplies the power to the EPROM, and is enabled by the ON/OFF- line via Q17 and Q18. Q19 raises the GND pin of U20 by 1 Volt, giving 6 Volts. When 5 Volts is required Q19 is shorted out by Q20 controlled by U18:D, this happens when 5V/6V- or R/W- is low.

U19 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.

## Construction

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 X 88mm in size. Note that the holes in the PCB are not plated through. The PCB overlay is shown in figure ## Due to the complexity of the PCB, the construction should be carried out in the following order.

Firstly enlarge U20 and U21 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 U20, U21, 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 121 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 U3 and U19. 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 U3, U19 and SK1.

It is now possible to fit the remaining components in any convenient order. Temporarily solder the LED's 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. U20 and U21 should be mounted with M3 nuts and bolts, U21 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 U3 or U19 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 VCC 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 U1 and about -9V on pin 6 of U1.

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 1000uF 25V capacitor directly across the DC input.

You could now fit the remaining IC's, 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.

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!

Insert U3 (the 6402), connect the programmer to the computers RS232 serial port (see fig ##), 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.

Type "FF" (don't type the quotes, and follow it with <Enter>). Check the logic levels on pins 5 to 12 of U3, 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.

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 U6, 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 U7, then "2" and U8,

and finally "3" and U9.

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. U10 is now disabled so its 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
-----
```

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 brought 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
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
-----
```

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 U11 or U12, then trace the fault as necessary. Note that the program may run quite slowly. A compiled 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 U15 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 U15 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 U19. 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 X 150 X 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 LED's, 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 shallow 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 LED's 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 LED's so that they slightly protrude through the holes and solder them into place. Complete the interwiring (see Fig ##) 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.

## Software Description

The following information describes the operation of the software shown in Listing 1, Listing 2 and Listing 3 (these listings are included in the software download).

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> 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
```

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.

The programs given in Listing 4 and Listing 5 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.

## **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. 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 (downloadable from this website).

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 EPROM's 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.

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

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 EPROM's, 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 2764's and 27128's 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 given in Listings 4 and 5. 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.

"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!

# Parts List

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

27	R1,6-19,27-33,36-38,41,42	10K
1	R2	4M7
1	R3	2K2
1	R4	1K0
1	R5	47K
2	R20,21	22K
3	R22,23,40	330R
1	R24	0R47 (or 2 x 1R0 in parallel)
1	R25	1K2
1	R26	8K2
1	R34	5K6
1	R35	1K5
1	R39	470R
1	RN1	100K x 8 SIL resistor network
1	RV1	470R or 500R horizontal preset
3	RV2-4	4K7 or 5K0 horizontal preset

## CAPACITORS

8	C1-5,9,15,16	10u 16V radial
2	C6,7	22p 0.2" pitch ceramic
1	C8	2n2 0.2" pitch ceramic
1	C10	220p 0.2" pitch ceramic
1	C11	470n 0.2" pitch
1	C12	47u 35V radial
1	C13	47u 16V radial
2	C14,17	1u0 16V radial
9	C18-20,22-27	100n 0.2" pitch
1	C21	220u 25V radial

## INDUCTORS

1	L1	470uH 2.3A bobbin type
---	----	------------------------

## SEMICONDUCTORS

1	U1	MAX232
1	U2	74HC4060
1	U3	CDP6402 (Maplin QQ04E)
1	U4	74HC14
1	U5	74HC238
4	U6-9	74HC75
1	U10	74LS244

2	U11,13	74HC4040
1	U12	74HC4024
1	U14	74HC20
1	U15	74HC00
1	U16	74HC08
2	U17,18	74HC32
1	U19	TL497
2	U20,21	7805
6	Q1,3,5,7,15,17	ZTX750/751/752
10	Q2,4,6,8,13,14,16,18-20	BC547/548/549
4	Q9-12	BC557/558/559
5	D1-5	1N4148 silicon signal diode
2	D6,7	OA47 germanium signal diode
1	D8	Red LED
1	D9	Yellow LED
1	D10	Green LED
1	D11	1N4001

## MISCELLANEOUS

1	SK1	28 way ZIF socket
1	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 U21, PCB, Case type MB6, M3 hardware, Connectors for power and RS232, Veropins, Tinned copper wire (24SWG) or through-PCB pins.

Small 0.47R resistors do not appear to be readily available - use two 1R0 components in parallel.

The software listed in this article, together with a comprehensive menu driven control program (IBM PC or compatible only) is available. This program, together with the source code, is included in the software available for downloadable from this website. The QuickBASIC Toolbox, required for the menu program source code, is also available for download.

## Updates

The Harris CDP6402 may be replaced with a Harris HD6402.

Nico van Rooijen has developed Windows based control software for this unit. This is included in the software available for downloadable from this website.