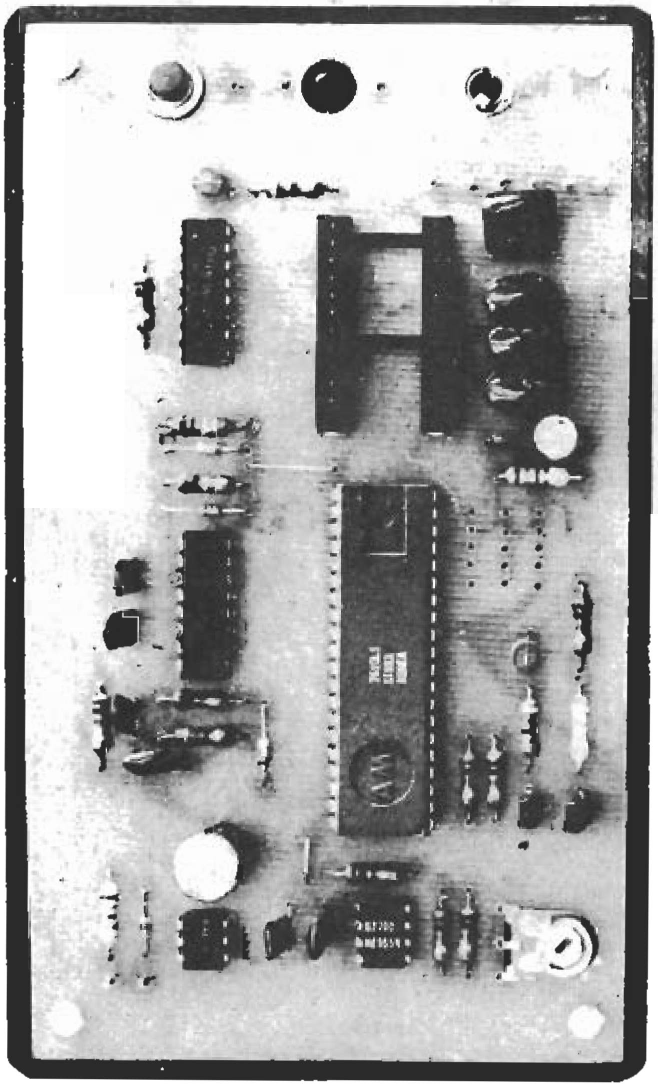
EPROM Programmer

This low-cost device will interface to just about any microcomputer, and requires only simple software to drive it. Based on a design by N.D. Hammond.



MANY OF OUR readers now have microprocessor systems, whether the large kind with keyboard, display and hunks of memory and I/O, or simply a small evaluation kit. Almost all of these systems accomodate a cassette recorder for storing your own programs, but the only drawback is that you have to load the contents of the cassette into RAM before you can use it. Wouldn't it be nice if you could have all those most used routines in ROM somewhere? Again most systems are designed to accomodate ROMs of some sort, frequently the popular 1K byte 2708 EPROM.

Now, you could go to the trouble and expense of having your ROMs programmed commercially, but what a drag waiting (and paying!). So why not build your own programmer, especially since this is made very easy by having the computer you already have do all the brainwork. You may find yourself frequently programming those not-so-expensive 2708s, and also quickly reprogramming them when such action is needed.

Here we present an EPROM programmer which is both simple and not too expensive. It communicates with your micro through a 20mA current loop which almost all systems have always had. It is based upon a design originally from N. D. Hammond.

The programmer is, in fact, slightly different from the original design submitted to us by Mr Hammond; we have replaced some TTL in his design with CMOS and added a data time-out synchronisation facility, on which more later.

DESIGN FEATURES

The objectives of the original design were simplicity of construction and operation, and low cost. Another requirement which must be met is simplicity and versatility of interfacing — one of our bigger headaches is the fact that everyone's system seems to be different.

This project meets these objectives very well. The interface to the user's computer is *serial*, i.e. through a 20 mA current loop. Most computers, except for some evaluation kits, have a suitable serial I/O port, so this is a pretty well universal interface. As a bonus, the UART and a couple of one-shots provide all the necessary timing signals, so the component count is low and cost is low.

A useful by-product of our switch to a completely CMOS design was a spare gate, which we put to good use in providing a 'synchronisation' facility. The idea is that if a supply glitch or noise causes the UART to miss a byte of data, so that the 2708 addressing is out of step with the desired addressing, a ¼ second pause at the end of each cycle will reset the 4040 to zero. This means that only that cycle will be affected and subsequent cycles will be correct, increasing the programmer's tolerance to glitches.

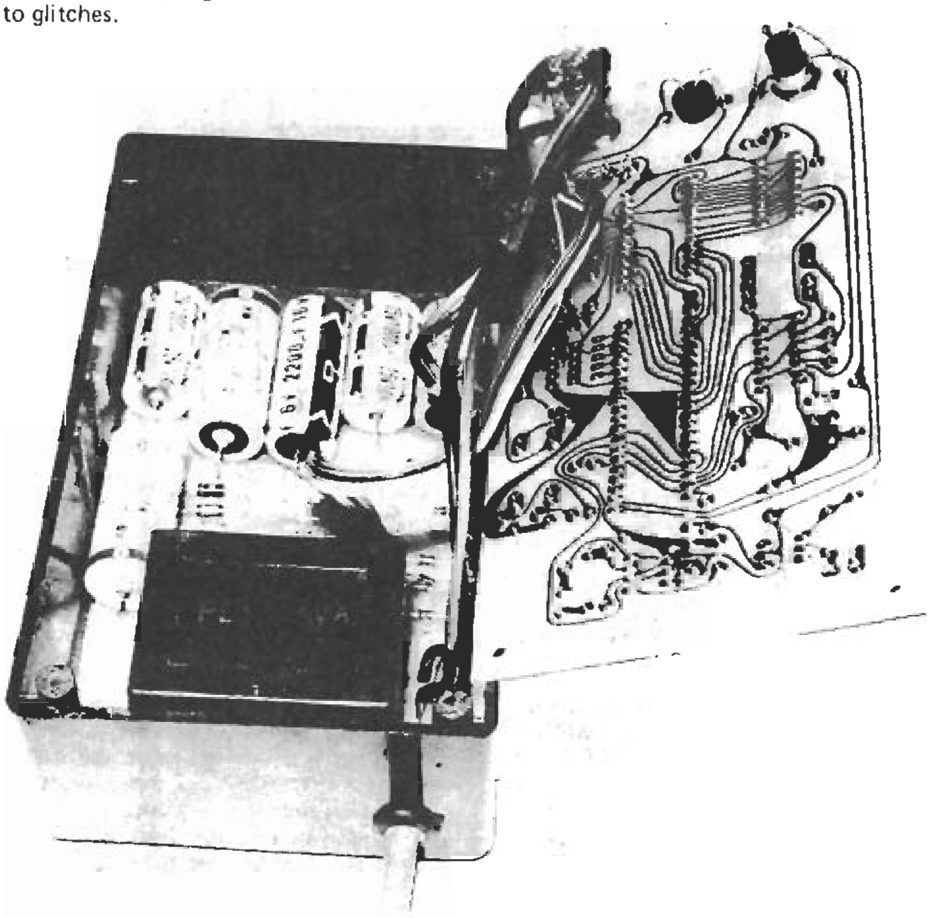
There is one slight penalty that has to be paid — at 300 baud, it will take about 70 minutes to output all 1024 addresses 125 times. This is by no means brilliantly fast compared to the theoretical minimum programming time of 104 seconds but it is a lot better than the several days that would be required by a commercial firm.

Mr Hammond originally supplied software for the 8080, but our tests of the circuit were done on a MEK6800D2, for which we have written a routine, reproduced here. Our routine incorporates a time delay of approximately ¼ second at the end of each run through the 2708 addresses, in order to take advantage of the time-out synchronisation feature. Mr Hammond's 8080 program does not include this facility, but it is easy to add a time delay loop which decrements (say) the BC pair using the DCX instruction.

ADJUSTMENT

Before adjusting the oscillator frequency first fit the links which set the start-stop bit arrangement of UART.

Now with power connected adjust RV1 until IC2 is operating at 4800 Hz.



PARTS LIST

ETI 638A

RESISTORS all ½W 5%
R1 . . . . . . 180R
R2, 3 . . . . . 10k
R4, 5 . . . . . 1k
R6 . . . . . . 10k
R7 . . . . . . 4M7
R8 . . . . . . 180k
R9 . . . . . . 100k
R10. . . . . . 470R
R11. . . . . . 10k
R12. . . . . . 1k
R13. . . . . . 10k
R14. . . . . . 33k
R15. . . . . . 10k
R16. . . . . . 47R
R17. . . . . . 180R

POTENTIOMETER
RV1 . . . . . 25k trim

CAPACITORS
C1. . . . . . . 8n2 polyester
C2–C4 . . . . 10n polyester
C5. . . . . . . 33n polyester
C6, 7 . . . . . 10n polyester
C8, 9 . . . . . 100n polyester
C10. . . . . . 100μ 25V electro
C11, 12. . . . 100n polyester
C13 . . . . . . 10μ 35V electro

SEMICONDUCTORS
IC1 . . . . . . 4N33 Opto coupler
IC2 . . . . . . 555 timer
IC3 . . . . . . MM5303 UART
IC4 . . . . . . 4049 Hex inverter
IC5 . . . . . . 4040 12 stage counter

Q1 . . . . . . 2N3638
Q2 . . . . . . 2N3904
Q3 . . . . . . 2N3905
Q4 . . . . . . 2N3904

D1–D4. . . . 1N914
LED1

MISCELLANEOUS
PC board ETI 638A
24 pin IC socket
Push button
Plastic box 158x96x50mm

ETI 638B

RESISTORS all ½W 5%
R1 . . . . . . 1k
R2, 3 . . . . . 120R
R4, 5 . . . . . 47R
R6 . . . . . . 470R
R7 . . . . . . 100R

CAPACITORS
C1. . . . . . . 470μ 50V electro
C2. . . . . . . 2200μ 16V electro
C3–C5 . . . . 1000μ 25V electro
C6. . . . . . . 470μ 50V electro

DIODES
D1–D6. . . . 1N4004
ZD1. . . . . . 27V 1W
ZD2. . . . . . 12V 400mW
ZD3. . . . . . 12V 1W
ZD4. . . . . . 5.1V 400mW
ZD5. . . . . . 5.1V 1W

MISCELLANEOUS
PC board ETI 638B
Transformer 120V–15Vct ½A
Switch DPDT toggle
3 wire line cord
Cable clamp

Kits and parts for this project are available from Northern Bear Electronics: see their ad in this issue for address.

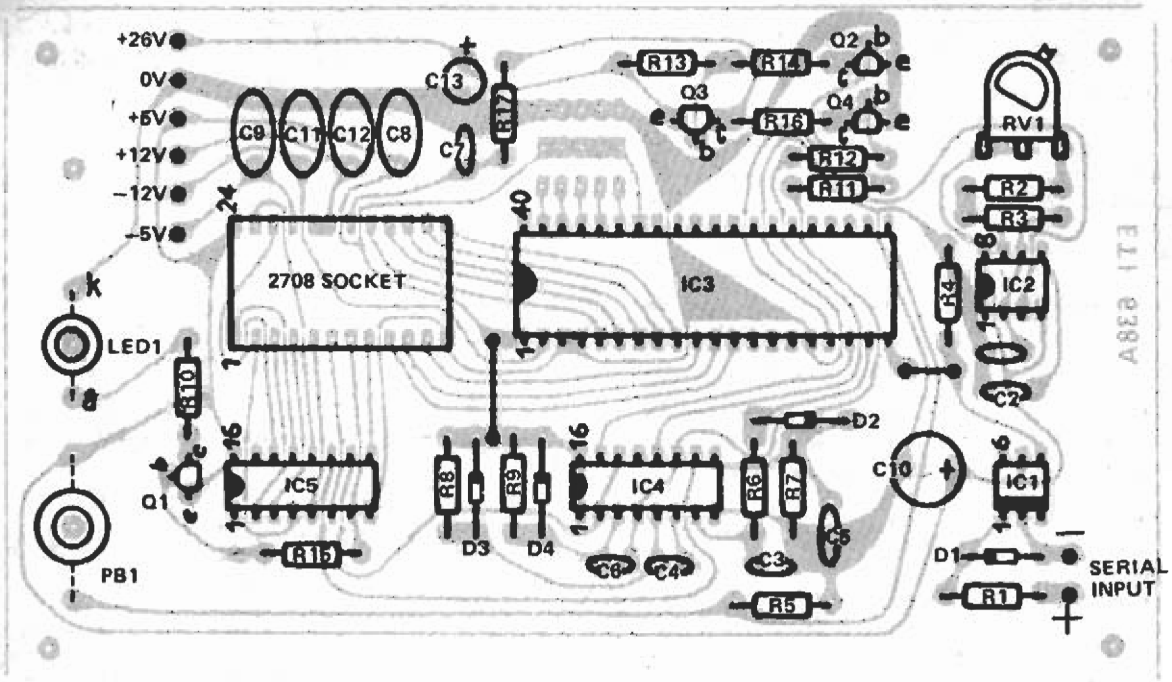


Fig. 1 The component overlay of the main board.

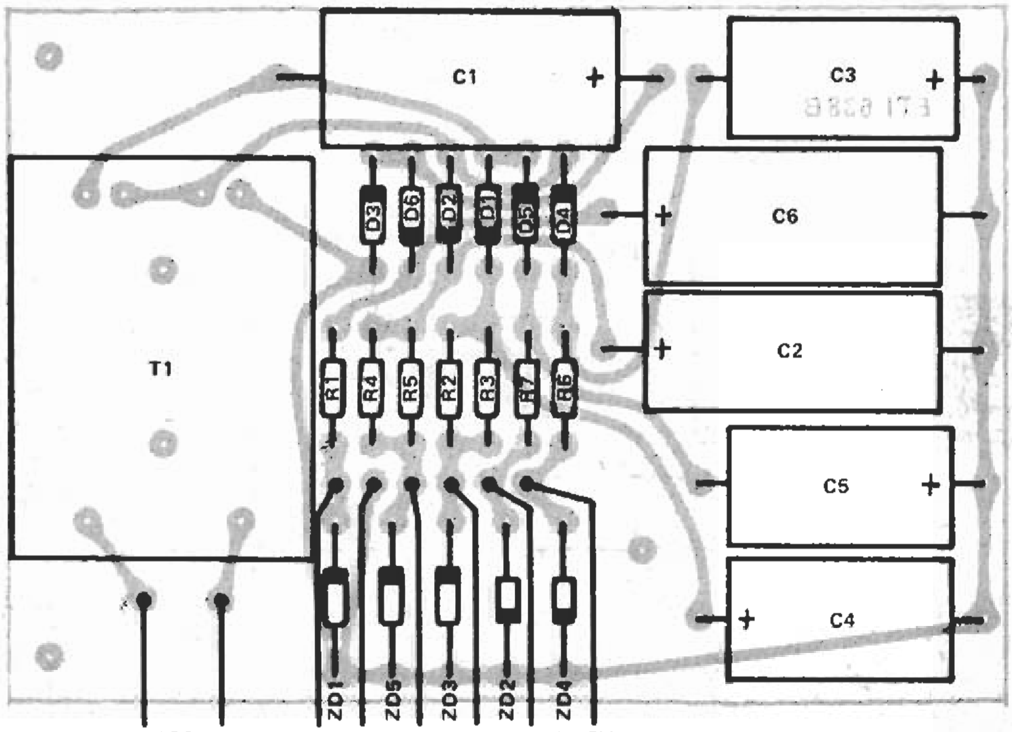


Fig. 2. The component overlay of the power supply.

CONSTRUCTION

We built our prototype into a plastic box with the power supply on one board in the box itself while the logic board was used in place of the lid.

These boards should be assembled according to the overlays provided. Normal handling procedures should be taken with the CMOS ICs and the UART. A good quality socket should be used for the EPROM as it will be used a lot. The pushbutton, LED and power switch are mounted on the logic board and connected from the rear.

With the power switch, due to the closeness of the capacitors on the lower board, the wires should be taken parallel to the pc board and the rear of the switch epoxied over to give protection. The connection between the power supply and logic board can be done with a piece of ribbon cable as the connections follow the same sequence.

We used pc pins for the data input points but a socket could be used if desired.

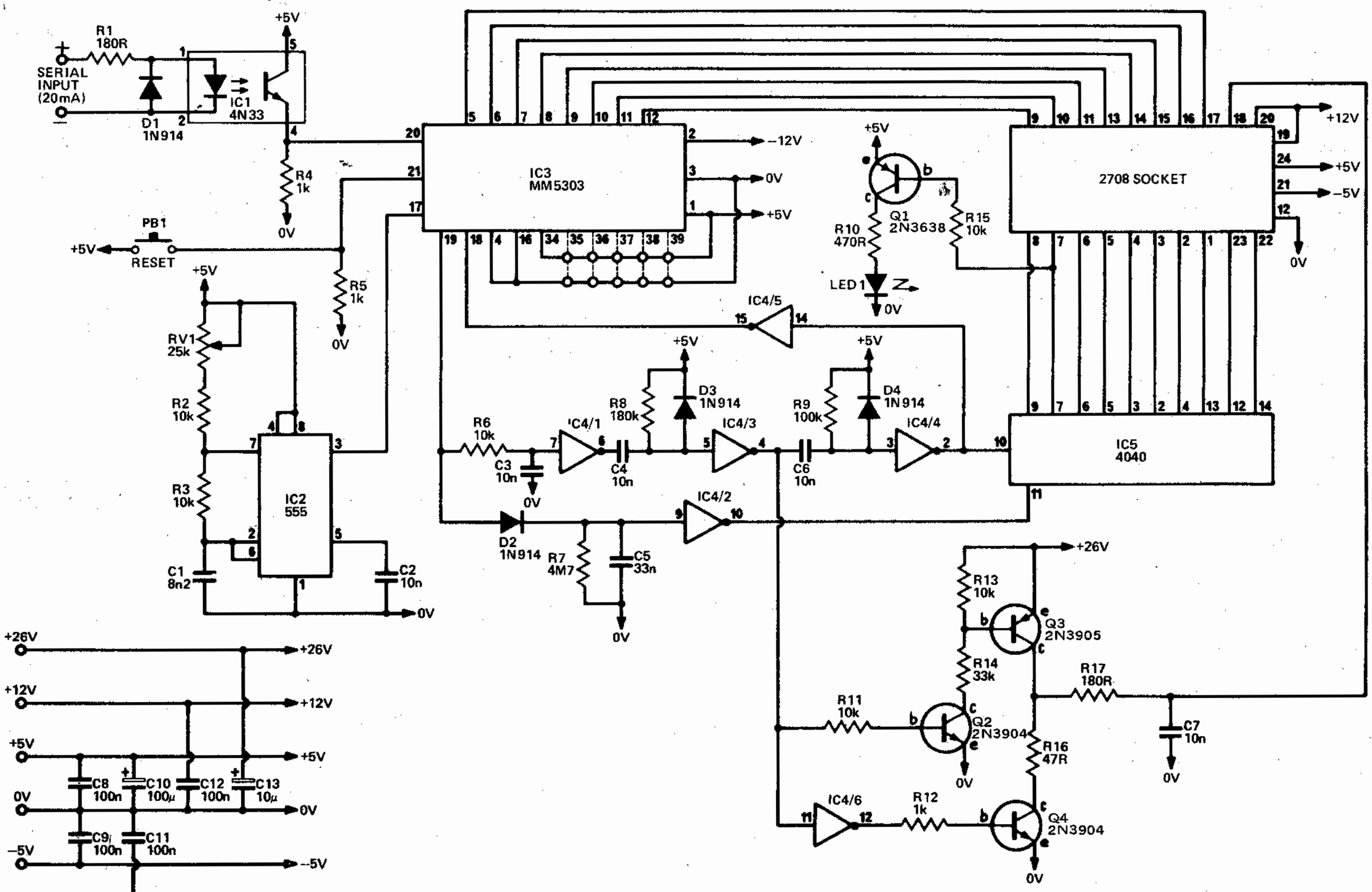


Fig. 3. The circuit diagram of the main circuit.

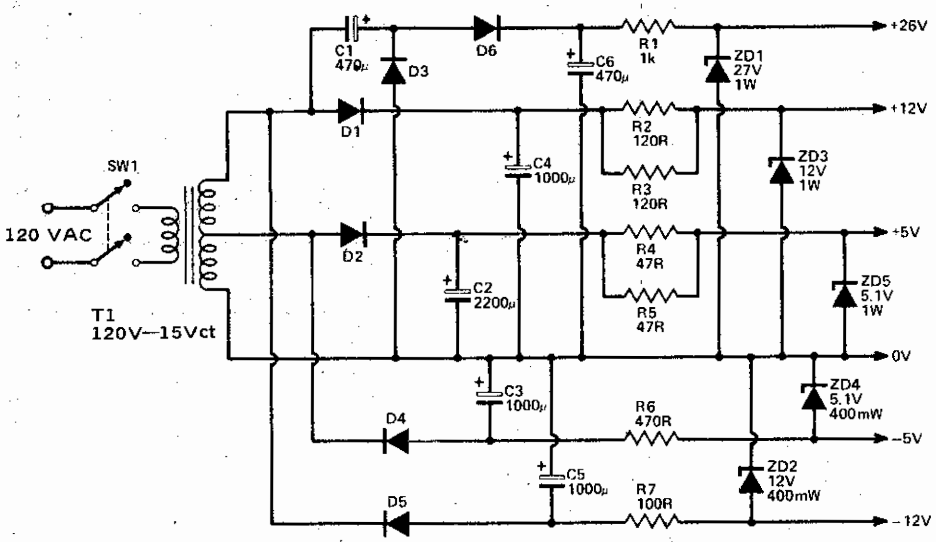


Fig. 4. The circuit diagram of the power supply.

HOW IT WORKS

A fully erased EPROM has every bit set to the "1" state. Programming sets selected bits to "0" To program a 2708 the selected address and corresponding data has to be presented to the EPROM and a 26V pulse applied to the program input pin. To make life more difficult each location has to be selected and programmed in sequence. Also a total time the program input has to be high for each location is 100 ms but the pulse used cannot be less than 100μs or longer than 1ms with about 1 ms recommended. This means that the IC has to be cycled through completely around 100 times for best results!

As we have a computer any way, (otherwise why the need for an EPROM!) we use it to provide the sequencing and timing needed.

The computer is programmed to copy data in its memory (1024 bytes) and sequentially transmit in serial form each byte 125 times. It also pauses for about ¼ second each 1024 bytes.

The serial information is transferred from the 20 mA loop into a 0-5V signal by the opto coupler, IC1, whose output is fed into the input of the UART IC3. This IC then converts this information into parallel form on pins 5-12 which is presented to the EPROM on its data lines. This IC needs a clock input at 16 times the Baud rate (4800Hz for 300 Baud) and IC2 is used for this.

The address lines for the EPROM are supplied by IC5 which is a 12 bit binary counter (we use only the first 10 bits) and this is reset to zero when no data is being received. On pin 19 of the UART we have an output which goes high when the serial data has been received and after this has been delayed by about 100 μs (R6/C3) the output of IC4/1 goes low. This triggers a 1 ms monostable (C4, R8, IC4/3) which drives the transistors Q2-Q4 to provide a 26V pulse to pin 18 of the 2708. At the end of this 1ms pulse a second mono is triggered (C6, R9, IC4/4), the UART is reset (pin 19 goes low again) and the address counter IC5 is incremented. The output (pin 19) of IC3 also charges C5 via D2 when it goes high. This causes the output of IC4/2 to go low allowing IC5 to be toggled (pin 11 of IC5 is the reset line). Provided the output of IC3 goes high regularly corresponding to data being received at 300 Baud C5 does not have time to discharge and the reset line remains low. If there is a pause at the end of a complete cycle the reset line will go high and will correct any error which may have been caused by a possible glitch.

The power indicator LED is driven by one of the outputs of IC5 and is turned on and off quickly indicating data is being received.

THE 2708

At this point we digress to describe the 2708 and the steps involved in using and programming it.

The device is a static 8192 bit EPROM organised as 1024 x 8. It is packaged in a 24 pin DIP with a quartz window which allows the data stored in the memory to be erased by exposure to ultra-violet light.

Reading the device is quite straightforward. The appropriate address is applied at the ten address pins, the chip select pin is taken low and after the appropriate access time (120ns from CS or 450ns from address select) the data is available at the eight output pins.

Fortunately, and in contrast to its predecessors, the 2708 is also simple to program. The chip select pin is taken to the 'write enable' level of +12 V and the applied address is cycled from 000H to 3FFH with the appropriate data applied at each address. After the data and address lines have settled at each address, a 26 V pulse of 0.1 ms to 1 ms duration is applied at the programming pin. The entire cycle of 1024 addresses is repeated until each address has received a mininum of 100 ms program pulse time.

Erasure is the simplest operation of all. The window is uncovered and the chip placed an inch or so away from an ultra violet tube. After half an hour or so, the memory is fully erased (to all '1's) and is ready for re-programming.

OPERATION

A fully erased EPROM has every bit set to the '1' state. Programming sets selected bits to '0'. It follows that a 2708 can be reprogrammed without erasing if there are no cases where a bit must be changed from '0' to '1', otherwise the device must be erased by exposure to ultra violet light. Any 'germicidal' UV tube is suitable for erasing.

The chip(s) which are to be erased should be placed about an inch or so from the tube and left for at least half an hour to ensure complete erasure.

To program the device, the pattern to be written should be available in RAM. The programmer is connected to the microprocessor's serial port which is configured for the appropriate signal format selected for the programmer (see table 3). The programmer is then reset to initialize the UART and ensure that the address counter starts at address zero. All that remains is for the microprocessor to output the contents of the selected RAM page, byte by byte at the port. As the programming pulse width is approximately 1ms, the whole 1024 bytes should be output at least 100 times. In practice, this should be increased by 25% or so to allow for the effects of component tolerances.

PROGRAMMER DESIGN

Use of the UART considerably simplifies the software requirements of the system which will drive the programmer, all that is necessary is a program which will output the required memory contents in order and repeat this for the required number of times.

TABLE 1. 6800 EPROM DRIVER FOR D2

6800 EPROM PROGRAMMER DRIVER FOR D2

```
               OUTCH       EQU    E37A
               PAGESTART   EQU    04
               NEXTPAGE    EQU    08
               ACIAS       EQU    8008
               ; INITIALISATION OF ACIA
0000  86 55                       LDA A   # %0101001
0002  B7 80  08                   STA A
               ; MAIN PROGRAM
0005  C6 7D                       LDA B     125
0007  CE 00  00   NEWCYCLE:  LDX PAGESTART
000A  A6 00       NEXTBYTE:  LDA A,    X
000C  BD E3  7A                   JSR  OUTCH
000F  08                          INX
0010  8C 04  00                   CPX  NEXTPAGE
0013  26 F5                       BNE  NEXTBYTE
0015  36                          PSH  A
0016  37                          PSH  B
0017  86 FF                       LDA A    $FF
0019  C6 FF                       LDA B    $FF
001B  5A               LOOP:      DEC  B
001C  26 FD                       BNE  LOOP
001E  4A                          DEC  A
001F  26 FA                       BNE  LOOP
0021  33                          PUL  B
0022  32                          PUL  A
0023  5A                          DEC  B
0024  26  E1                      BNE  NEWCYCLE
0026  3F                          SWI

For Test:
000A  86  XX      NEXTBYTE: LDA A      XX
outputs ASCII character XX
 or
000A  4C          NEXTBYTE: INC  A
000B  01                    NOP
outputs incrementing characters.
```

TABLE 3 SIGNAL FORMAT OPTIONS

OPTION		INPUT	UART PIN	LEVEL
No OF DATA BITS	8	NDB2	37	H
		NDB1	38	H
	7	NDB2	37	H
		NDB1	38	L
PARITY	EVEN	NPB	35	L
		POE	39	H
	ODD	NPB	35	L
		POE	39	L
	INHIBIT	NPB	35	H
		POE	39	X
No OF STOP BITS	1	NSB	36	L
	2	NSB	36	H

H=HIGH (+5V) L=LOW (0V) X=DON'T CARE

TABLE 2 – INTERFACE PROGRAM FOR 8080/Z80

```
                 ;***** INTERFACE PROGRAM FOR 2708 EPROM PROGRAMMER *****
                 ;
                 PAGESTART:  EQU   04H          ; HIGH ORDER BYTE OF RAM ADDRESS
                                                ;   TO BE LOADED IN EPROM ADDRESS
                                                ;   ZERO – LOW ORDER BYTE IS ZERO
                 NEXTPAGE:   EQU   08H          ; HIGH ORDER BYTE OF PAGESTART + 1024
                 CTRL:       EQU   0            ; ADDRESS OF I/O STATUS & CONTROL PORT
                 DATA:       EQU   1            ; ADDRESS OF I/O DATA PORT
                 ;
                 ; INITIALIZATION – NOTE: SYSTEM DEPENDENT. THIS SEGMENT WRITTEN
                 ; FOR AN INTEL 8251 SERIAL I/O PORT
                 ;
0000: 3E 4E                  MVI   A, 4EH       ; MODE INSTRUCTION. SELECT 1 STOP,
0002: D3 00                  OUT   CTRL         ;   8 DATA AND NO PARITY FORMAT
0004: 3E 11                  MVI   A, 11H       ; COMMAND INSTRUCTION. RESET 8251
0006: D3 00                  OUT   CTRL         ;   AND SET TX ENABLE
                 ;
                 ; MAIN PROGRAM
                 ;
0008: 06 7D                  MVI   B, 125       ; NO OF PROGRAMMER CYCLES TO B
000A: 26 04      NEWCYCLE:   MVI   H, PAGESTART ; HIGH ORDER ADDRESS OF BYTE 1 TO H
000C: 2E 00                  MVI   L, 0         ; LOW ORDER ADDRESS TO L
000E:            NEXTBYTE:
000E: DB 00      TESTPOINT:  IN    CTRL         ; READ I/O PORT STATUS
0010: E6 01                  ANI   01H          ; MASK ALL EXCEPT READY BIT
0012: CA 0E 00               JZ    TESTPORT     ; LOOP UNTIL READY BIT SET
0015: 7E                     MOV   A, M         ; MOVE SELECTED BYTE TO ACC
0016: D3 01                  OUT   DATA         ;   AND SEND TO PROGRAMMER
0018: 23                     INX   H            ; SELECT NEXT BYTE TO BE SENT
0019: 7C                     MOV   A, H         ; TEST CONTENTS OF H TO SEE WHETHER
001A: FE 08                  CPI   NEXTPAGE     ;   LAST BYTE HAS BEEN SENT
001C: C2 0E 00               JNZ   NEXTBYTE     ; IF NOT, REPEAT LOOP
001F: 05                     DCR   B            ; ELSE DECREMENT CYCLE COUNTER
0020: C2 0A 00               JNZ   NEWCYCLE     ; IF NOT FINISHED START NEW CYCLE
0023: 76                     HLT                ; ELSE HALT
```