Program a 2708 in two minutes flat!

EPROM Programmer suits the TRS-80, Sorcerer, &c.

If you have ever wanted to rewrite or extend the operating system of your microcomputer or if you're interested in dedicated microprocessor applications then this EPROM programmer is just the thing. It is an inexpensive unit that uses readily available ICs, interfaces directly to the expansion bus on the back of all the popular 8080/Z80 microcomputers and programs 2708s, 2716s, 2758s and 2732s.

by RON DE JONG

The ready availability of inexpensive EPROMs particularly the 2708 and 2716, now presents a vast number of possibilities for the computer enthusiast. Since EPROMs can be conveniently erased and reprogrammed, the operating system of a microcomputer could be modified or extended to include such features as renumbering routines in BASIC. The Compucolor microcomputer for example provides for 8k of additional ROM while the ROM-PACs used on the Sorcerers have provision for the use of EPROMs rather than ROMs.

Some more exciting possibilities are in the area of dedicated microprocessor applications. Just to mention a few — robots, intelligent video terminals, music synthesisers, train controllers, speech recognisers, and burglar alarms. It is a simple matter to prototype these sort of devices by first running the operating programs on such systems as the 6800 D2 kit, or by using the numerous assembler editor programs available with most computers.

The problem up until now though has been the lack of EPROM programmers suited for use with personal computers. With this in mind, we decided to design an EPROM programmer suitable for direct connection to the bus expansion ports of personal computers, in particular, the 8080 and Z80-based machines such as the Exidy Scorcerer and Tandy TRS-80.

Our EPROM programmer programs the popular 2708 as well as the more recent 2716, 2758 and 2732 EPROMs. It can read the contents of the EPROM to check that it has been erased prior to programming and check that the EPROM does in fact contain the correct data after programming. Using the machine language driver shown elsewhere in this article programming time is about two minutes for a 2708.

Before discussing the operation of the EPROM programmer, in detail, let's look at the operation and programming requirement of the 2708 and 2716.

The popular 2708 EPROM uses floating-gate avalanche mode MOS transistors as the storage cells. Stored charges on the floating gates are used to control the conduction of the MOS transistors, to determine whether they effectively store a "1" or a "0".

The floating gate's charge is produced by inducing a non-damaging avalanche breakdown in the drain-channel junction of the cell. High energy electrons from the avalanche breakdown are then injected into the floating gate, charging it negatively. Since the floating gate is surrounded by an extremely effective insulator, this charge will remain practically indefinitely, and hence the stored pattern will also remain.

To erase a programmed EPROM, the chip is irradiated with ultra-violet light. The resulting photons impart enough energy to the trapped electrons to allow

While the programmer will accept four EPROM types, the software featured in this article is specifically written to program the 2708. Small modifications to the program are required for the other EPROM types.

them to escape from the floating gate, leaving it uncharged.

An erased EPROM has all memory cells effectively containing 1's, so programming consists of inducing avalanche mode breakdowns in the appropriate cells to produce the required zeros. In principle, one programming pulse is required for each memory location. The appropriate address and data information must be applied to the address and data pins of the EPROM.

In practice, due to power dissipation limits, it is necessary to apply a short programming pulse between .1 and 1ms long to each memory location in sequence. Each complete sequence is called a program loop and N such loops are required, such that the total programming time for each location (N × pulse width) is at least 100ms.

Programming pulses are +26V in amplitude and during programming the CS/WE or chip-select pin must be at +12V. In addition three supplies are required to operate the 2708 during both programming and reading, namely +12V, +5 and -5V. In comparison the 2716 EPROM is considerably easier to program and it requires only the standard +5V supply. It is a 2k x 8 EPROM in which each location only has to be programmed once by a single 50ms TTL-level programming pulse with the chip select high and the Vpp supply at +25V.

The 2758 is a 1k version of the 2716 chip but with one half of the memory defective. Rather than throw these devices away, manufacturers have labelled them 2758A or 2758B depending on which half of the die is defective. These chips still have the advantage over 2708s however since they are single supply EPROMs, like the 2716 itself. Programming requirements are also the same except that the software has to "know" which half of the chip to program.

To see how we have satisfied all these requirements, refer now to the block diagram, Fig. 1 and Fig. 2 which shows the waveforms involved. The programmer occupies 4 consecutive address locations in I/O space. Depending on which of those four addresses is accessed by the microprocessor and whether a read or write operation is performed, one of the six outputs of the address decoder will briefly go low.

Two of the address decoder outputs go to the reset and clock inputs of a 12-bit binary counter. The counter outputs are connected to the address inputs of the EPROM, so that by simply accessing the appropriate addresses the counter can be made to reset or increment. Hence any of the 1024 locations in the 2708 EPROM can be accessed by appropriate reset and increment operations. This actually saves a lot of time because the memory locations of the EPROM will always be accessed in sequence during programming or reading.

Another two outputs from the address decoder drive the program-pulse

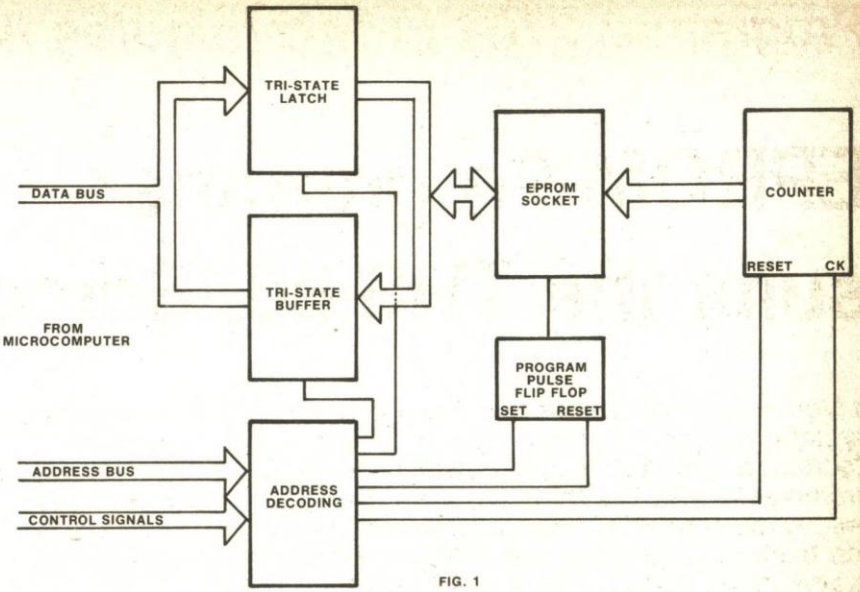


FIG. 1

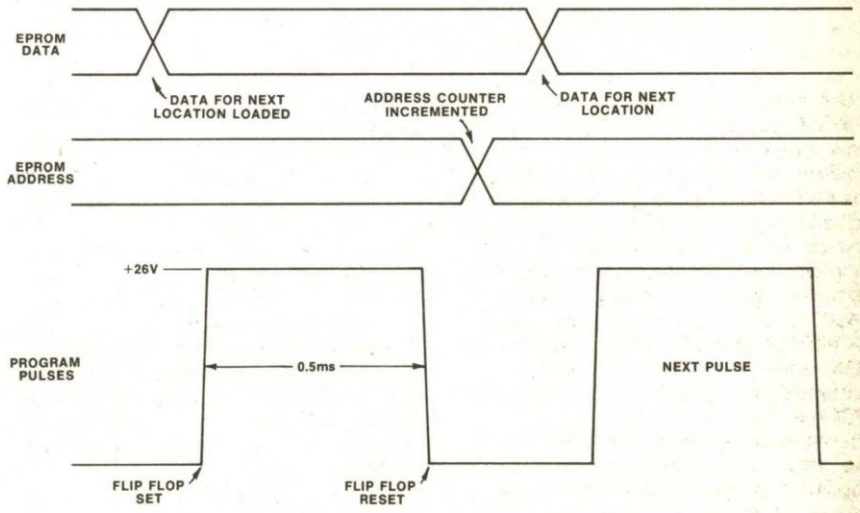


FIG. 2 SEQUENCE OF PROGRAMMING SIGNALS FOR 2708 EPROM

flipflop. One of the lines sets the flipflop and the other resets it. The two remaining lines from the decoder enable the Tristate latch and the Tristate buffer. When the microprocessor reads from a particular address location the Tristate buffer will be enabled and the data present at the output will be read, and when the micro writes to that same location the Tristate latch is enabled instead, latching the data.

Fig. 2 shows how these signals are used to program the 2708 EPROM. Firstly the EPROM programmer must be switched to the program mode so that the Tristate outputs of the EPROM are turned off and the Tristate outputs of latch are turned on. This means that any data written into the Tristate latch by the microprocessor will be present at the data inputs of the EPROM.

Each program loop starts with the address counter being cleared by accessing the appropriate address for each cycle in the loop and then the data for the particular memory location is loaded. A short time later the program-pulse flipflop is set and +26V is then applied to the program pin. After waiting 0.5ms, the microprocessor resets the flipflop which removes the +26V program pulse. The address counter is then incremented and new data loaded for the next location and then the cycle repeats itself.

To satisfy the requirement that the total program time for each location be at least 100ms, 200 program loops are required (200 x 0.5 = 100ms).

Using the software driver shown elsewhere in this article programming time for a 2708 is about two minutes.

In the case of the 2716 or 2758 only one program loop is required but each program pulse is at normal TTL levels and the pulse width for each location is 50ms, giving a total programming time of 100 seconds.

Looking now at the circuit diagram, we can see the various functional blocks in Fig. 1. The data bus, D0 to D7 and the

address bus A0 to A7 can be seen on the left of the diagram. Note that only address lines A0 to A7 are used because the EPROM programmer is connected as an I/0 port.

The control signals provided for on the circuit are IN, OUT, RD, WR, IOREQ, DBUSEN, and BDUSDIR. The first two, IN and OUT, are all that is required for use with the TRS-80. The other controls are required when interfacing to the Exidy Sorcerer.

The purpose of these various signals is as follows: WR will go low during either a memory or I/O write operation by the microprocessor and RD will go low during a memory or I/O read operation. The IOREQ signal goes low when an I/O operation rather than a memory operation is performed. The TANDY TRS-80 combines the RD and IOREQ and WR and IOREQ signals internally to generate the IN and OUT signals respectively. Hence if the TANDY is used, the IOREQ line on the programmer would simply be earthed.

The Sorcerer's signal DBUSEN is the Tristate enable for the data bus provided by the expansion port and it should be low if the data bus is to be enabled. DBUSDIR is the signal to the Sorcerer indicating the direction of the data bus, low for a read and high for write.

The first two address lines A0 and A1 plus the IN and OUT signals go to IC4 which is a dual 2-to-4 line decoder. The remaining six address lines are decoded by IC6 which is a triple 3-input NOR gate. The outputs of IC6a and IC6b will go high only when their inputs are all low. These two outputs are combined by IC7d which then generates a low enable for the rest of the address decoding circuit.

As a result, if all the address lines are connected directly to the inputs of IC6a and IC6b the EPROM programmer will be located at 00 in I/O space. Two inverters, IC5b and IC5c, can be used however to invert any of the six address lines so the EPROM programmer can be relocated quite easily. As it turns out neither the Sorcerer, Compucolor or Tandy machines has anything at 00 location, so you are quite free to locate the programmer there.

Now, the enable signal from IC6c is used to gate the IN and OUT signals via two NAND gates, IC7a and IC7b. These signals then enable either one of the two 2-to-4 decoders. Hence when the correct address is present and an I/O read or write operation is performed, one of the two decoders will be enabled, and depending on the lower two address bits, one of the four outputs of that particular decoder will go low.

To simplify our discussion of the circuit we will assume that the I/O address of the EPROM programmer starts at 0. So when a write to location 0 is performed, output 1Y0 of IC4 will go low, and when a read to location 2, for example, is performed then output 2Y2 will go low briefly. Two of the enable lines go to the Tristate latch and the buffer for reading and programming data. All the other decoded outputs, however, are used as signals themselves and nothing is actually read from or written to the computer's data bus.

The Tristate latch used in the circuit is IC1, a 74LS374. The latch signal is derived from output 1Y0 of the decoder, so when the microprocessor performs an I/O write to location 0 the brief enable signal from the decoder will latch the data present on the data bus into the latch. If the microprocessor performs an I/O read from the same location the 2Y0 output of the decoder would go low instead, enabling the Tristate outputs of IC2 which is an 81LS95 tristate buffer.

We estimate that the current cost of parts for this project is approximately

$75

including sales tax. This includes the cost of the flat cable and connectors.

The remaining outputs of the decoder drive the program-pulse flipflop and the address counter. The program pulse flipflop consists of IC8d, IC8c and IC7c which are 2-output NAND gates and IC5f which is an inverter. IC8d and IC8c are connected together as an RS flipflop, with pin 9 of IC8c as the R input and pin 12 of IC8d as the S input. When a brief low pulse is applied to the S input the output of IC8d will go high and the output of IC8c will go low. Similarly when a low pulse is applied to the R input, the output of IC8d will go low and IC8c will go high.

The SET input of the program pulse flipflop is obtained from output 2Y2 of the decoder so the program is initiated when the microprocessor writes to I/O location 2. The Reset input is obtained from output 2Y1 of the decoder via IC7c and IC5f which together function as a "power-on reset" for the flipflop. This to prevent the flipflop from being set when power is first applied – a situation which could damage the EPROM should the programmer be in the program mode.

Power on reset works as follows. When the power is first applied pin 10 of IC7c will be held low by the 10uF capacitor. The output of IC7c will then be high and the output of IC5f low causing the flipflop to be reset. The capacitor is charged up via a 2.2k resistor so that about 20ms later, pin 10 will be high and the combination of IC7c and IC5f merely pass the signal from the decoder, uninverted to the flipflop and normal operation can follow.

Outputs from the program-pulse flipflop drive a simple class-B output stage which generates the +26V programming pulses. When the flipflop is set the output of IC8d will be high and IC8c low. This causes transistor Q2 to be off and Q1 to be turned on which then turns Q3 on and pulls the output up to the +26V supply line. If the flipflop is reset though, Q1 and Q3 will be off while Q2 will be on.

Rise and fall times for the programming pulses must be within certain limits to ensure reliable programming. This is accomplished by using a .01uF capacitor on the output of the stage along with a 100 ohm resistor in series with the collector of Q3 and a 39 ohm resistor in series with the collector of Q2.

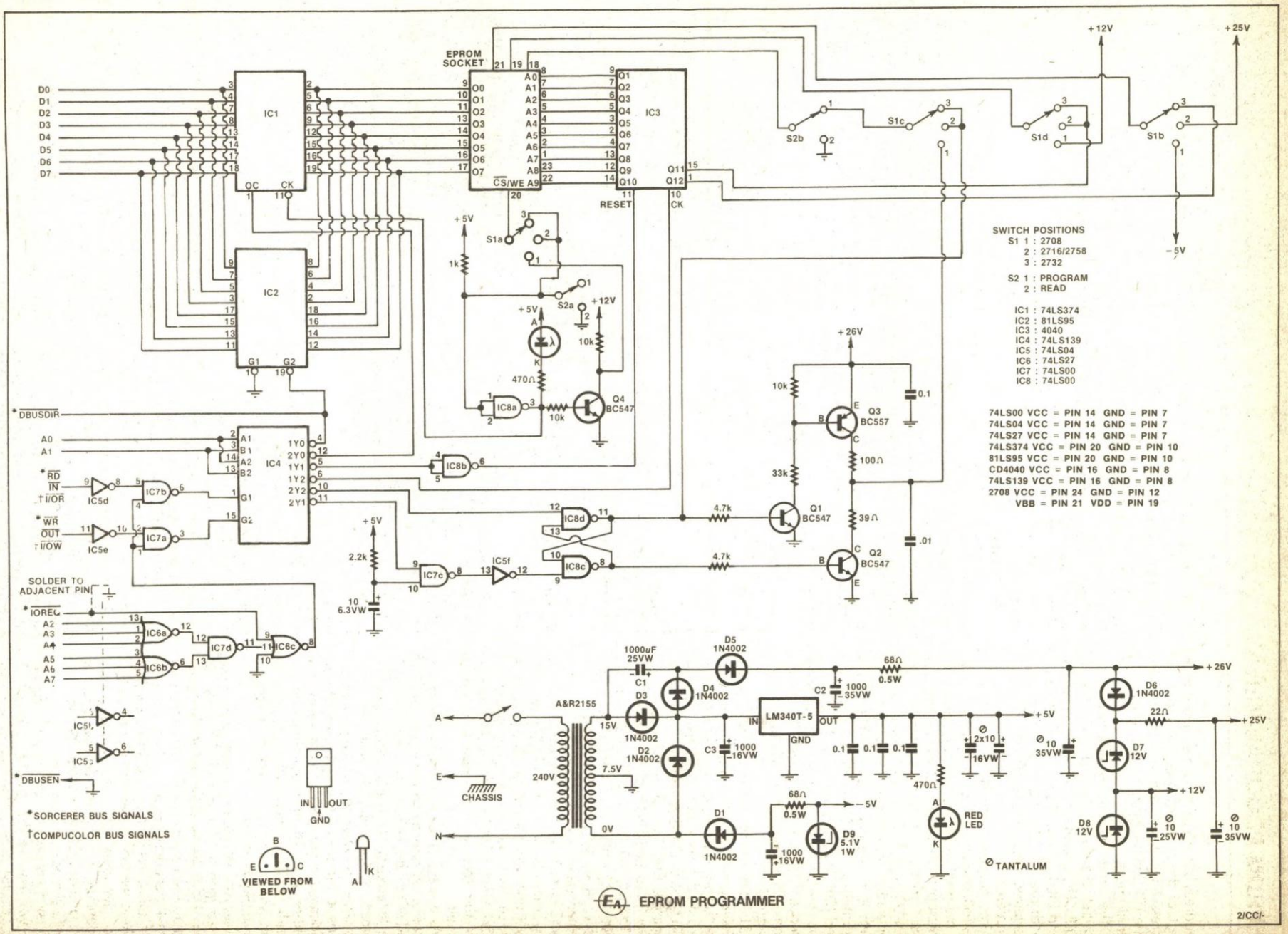
The address counter is IC3 which is a 4040 12-bit CMOS counter. Reset and clock inputs for the counter are obtained from the decoder outputs 1Y1 and 1Y2, respectively. Hence to reset the counter the micro would perform an I/O read from address 1, and to increment the address it would perform an I/O read from location 2. Note that IC8b inverts the signal to the rest input of the counter because the outputs of the decoder are active low while the counter resets on a high.

That completes our description of the basic data, address and programming circuitry. Data and address lines for the various EPROMs are in fact the same except for pins 18, 19, 20, 21. Depending on the type of EPROM being programmed, these pins will be switched to various supplies or signals by switch S1. This switch is a 4-pole 3-position rotary switch with each pole corresponding to one of the pins 18, 19, 20, 21. The three positions on the switch correspond to the three basic types of EPROM, viz 2708, 2716/2758 and 2732.

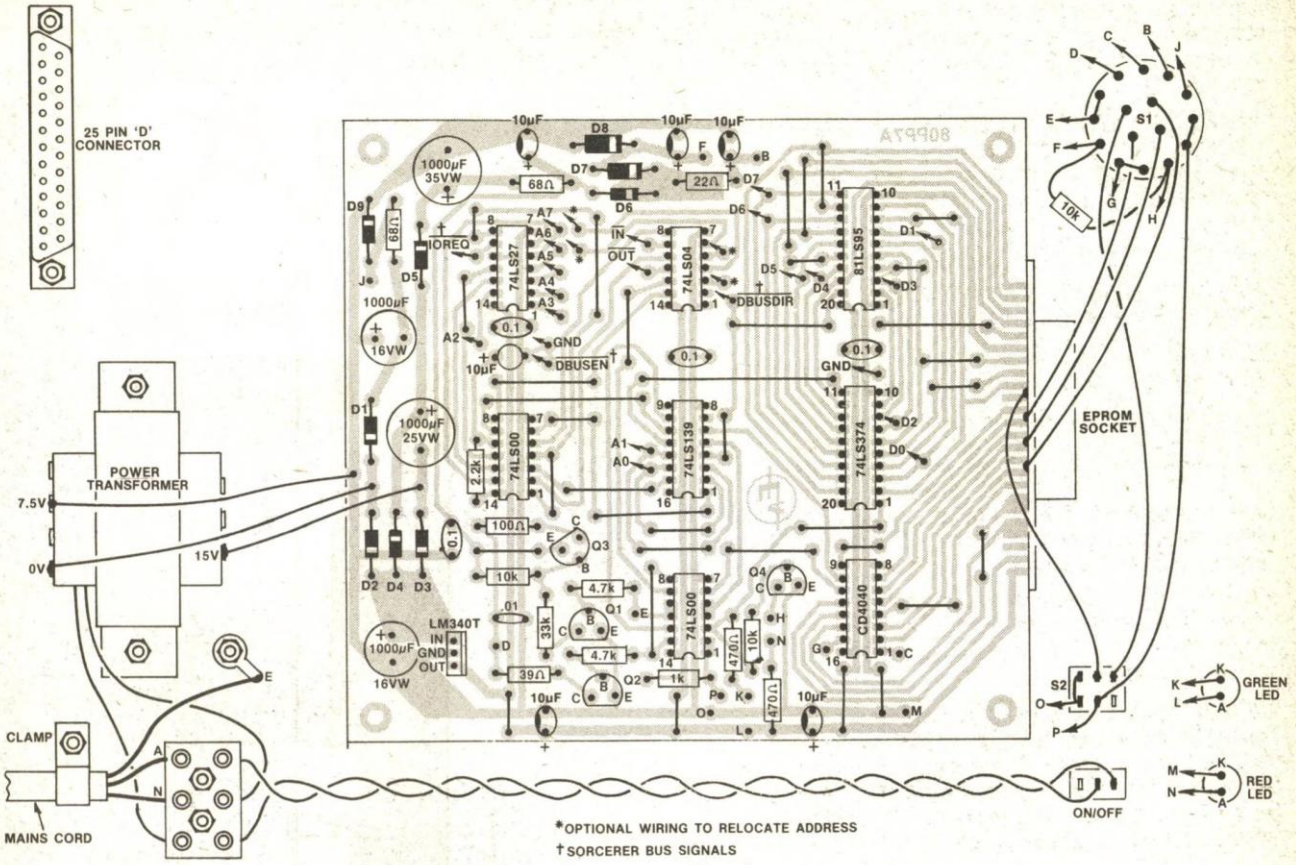
The only other switch to affect these four pins is S2. This is a DPDT toggle switch which sets the EPROM programmer into either the read or program modes. Without going into too much detail we will discuss only pins 18 and 20. Switch S2b switches pin 18 which is the program-pulse pin of the EPROM. In the read mode it switches pin 18 to ground, while in the program mode it switches pin 18 to S1c which then connects it either to the +26V or directly to the TTL level program pulses from the flipflop itself.

Pin 20 is the chip-select pin, CS/WE of the EPROM. In the read mode it will be low for all the EPROM types while in the program mode it should be at +12V for the 2708 and +5V for the 2716, 2758

The complete circuit, at right, can be built for around $75, including all hardware.


EPROM SOCKET
IC1
IC2
IC3
IC4
D0
D1
D2
D3
D4
D5
D6
D7
CS/WE
RESET
+12V
+25V
-5V
+5V
+26V
S1a
S1b
S1c
S1d
S2a
S2b
SWITCH POSITIONS
S1 1 : 2708
2 : 2716/2758
3 : 2732
S2 1 : PROGRAM
2 : READ
IC1 : 74LS374
IC2 : 81LS95
IC3 : 4040
IC4 : 74LS139
IC5 : 74LS04
IC6 : 74LS27
IC7 : 74LS00
IC8 : 74LS00
74LS00 VCC = PIN 14 GND = PIN 7
74LS04 VCC = PIN 14 GND = PIN 7
74LS27 VCC = PIN 14 GND = PIN 7
74LS374 VCC = PIN 20 GND = PIN 10
81LS95 VCC = PIN 20 GND = PIN 10
CD4040 VCC = PIN 16 GND = PIN 8
74LS139 VCC = PIN 16 GND = PIN 8
2708 VCC = PIN 24 GND = PIN 12
VBB = PIN 21 VDD = PIN 19
Q1 BC547
Q2 BC547
Q3 BC557
Q4 BC547
A&R2155
LM340T-5
240V
7.5V
0V
1N4002
RED LED
5.1V 1W
SOLDER TO ADJACENT PIN
*SORCERER BUS SIGNALS
†COMPUCOLOR BUS SIGNALS
VIEWED FROM BELOW
TANTALUM
EPROM PROGRAMMER
2/CC/-

Why not put all your frequently used routines in an EPROM for real convenience?



Note that a heatsink is required for the three-terminal regulator.

and 2732 EPROMs. This switching is accomplished by S1a and S2a. In the read mode, S2a is switched to ground. Hence for the 2716, 2758 and 2732 pin 20 will be low. S2a is also connected to the input of IC8a which inverts this signal, turning transistor Q4 on and hence also sending pin 18 to low for the 2708.

In the write mode the 1k resistor connected to S2a pulls pin 20 high (+5V) for the 2716, 2758, 2732 and at the same time pulling the input of IC8a high, which turns Q4 off. This results in +12V being applied to pin 20 if it is a 2708, which is precisely what is required. In addition a green LED is connected to the output of IC8d so that it turns on when the programmer is set to the program mode.

To prevent the Tristate latch IC1 from having its outputs enabled at the same time as the EPROM is enabled, the output of IC8a is connected to the output-control line of the latch, pin 1. Hence as soon as the programmer is switched to read, IC1 will be disabled and the EPROM enabled, while during programming the outputs IC1 are enabled and the ROMs disabled.

The power supply for the programmer is virtually the same as that used in a previous EPROM programmer project for 2650s and SC/MPs published February 1979 (File No. 2/CC/35). The transformer is an A&R2155 and it drives three seperate power supplies for -5V, +5V and +26V. The -5V supply is a simple half wave rectifier consisting of diode D1 and capacitor filter followed by a zener regulator using a 5.1V 1W zener.

A full-wave rectifier consisting of diodes D2 and D3 and a 1000uF filter capacitor C3, supplies an LM340T-5 three-terminal regulator. The regulator delivers a constant 5V supply with low ripple and excellent line and load regulation. The output of the regulator is decoupled by two 10uF tantalum capacitors and three 0.1uF capacitors. These are distributed around the board to reduce the effects of supply line inductance, which can be a problem due to the fast risetimes encountered with TTL.

About 30VDC is generated by a tripler circuit consisting of diodes D4, D5 and 1000uF capacitors C1, C2 together with the full-wave rectifier already described. The operation of the circuit can be understood by noting that when the 15V tap swings negative (about −10V) then the 1000uF capacitor is charged up to 20V by D4 which effectively clamps the positive side of C1 at +10V. Now when the 15V tap swings up to +10V D4 is reverse-biased and D5 is forward biased and C1's charge is dumped into C2.

The tripler is followed by a zener regulator consisting of a 68 ohm resistor, two 12V 2.5W zeners and a diode in series. Theoretically this would give a nominal output voltage of 24.6 but due to the positive thermal coefficient of the zeners this nominal voltage will exceed 25V. The voltage tolerance of the zeners also means that the actual output voltage can vary as much as ±1.4V hence this voltage should actually be checked and if it is not +26±1V then a higher or lower voltage zener should be substituted for D7.

The specified programming voltage for the 2716 and 2758 EPROMs is 25±1V which is 1V below the specified voltage range for the 2708s, viz 26±1V. Hence to keep this voltage well within specifications we have tapped off a seperate +25V supply via diode D6. The voltage at this point is 0.6V below the nominal +26V. A 22-ohm resistor has also been included in series with the +25V supply to limit the current should the EPROM selector be accidentally set to the 2716/2758 position when a 2708 is in the socket.

PARTS LIST

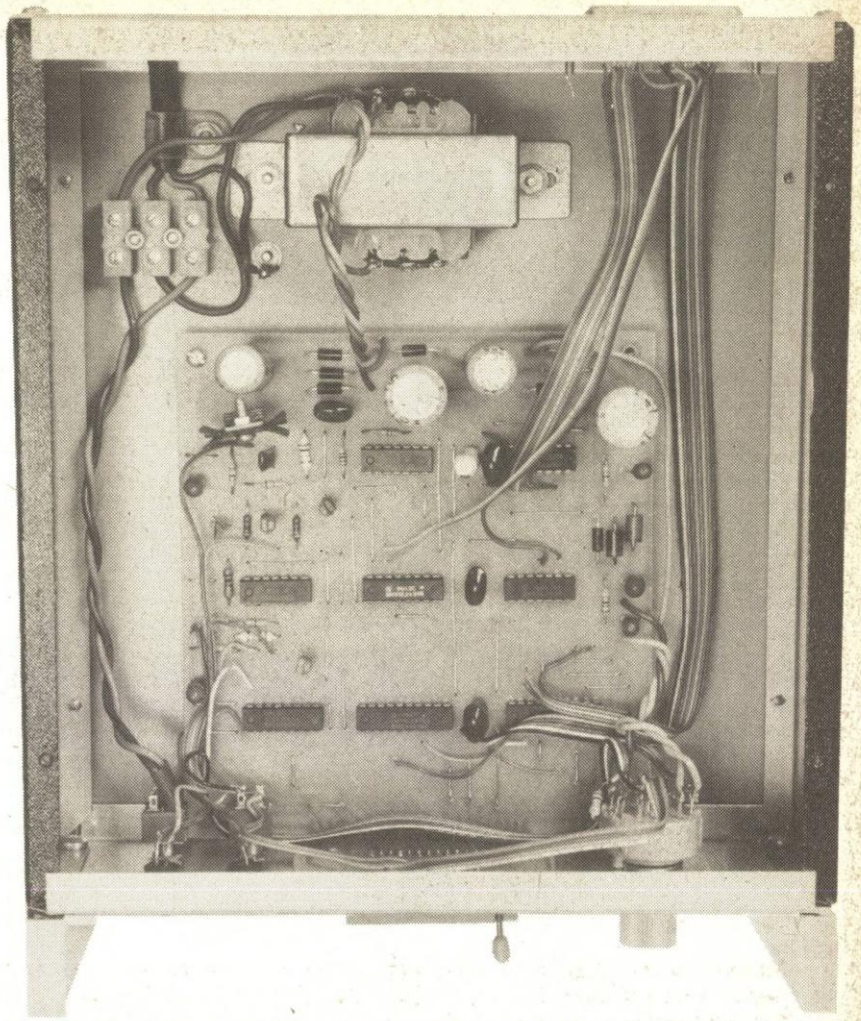
1 Horwood Instrument Case, 228 × 76 × 203mm or Pactec Instrument housing Model CH325 234 × 93 × 261mm (D × H × W)
1 A&R2155, Altronics 2155 or DSE2155 mains transformer
1 PC board coded 80pp7a, 140 × 127mm
1 PC board coded 80pp7b, 48 × 69mm
1 24-pin zero-insertion force socket
1 25-pin panel-mounting male D-connector and 25-pin female D-connector mass terminated with a suitable length of flat cable
1 20/40 way card edge connector with solder lugs
1 SPST miniature toggle switch
1 DPDT miniature toggle switch
1 3-position 4-pole rotary switch
1 mains cord and plug
1 TO-220 heatsink
1 mains cable clamp and rubber grommet
½ metre of rainbow cable
2 large LED bezels
1 large knob

SEMICONDUCTORS:
1 74LS374 octal Tristate latch
1 81LS95 octal Tristate buffer
1 74LS139 dual 2-to-4 decoder
1 74LS27 triple three-input NOR gates
1 4040 12-stage counter
2 74LS00 quad NAND gates
1 74LS04 hex buffer/inverter
1 LM340T-5 three terminal regulator
3 BC547 NPN transistors
1 BC557 PNP transistor
2 BZX70C12 zener diode
1 BZX70C13 zener diode
1 BZX70C5V1 zener diode
6 1N4002 diodes
1 large red LED
1 large green LED

CAPACITORS:
1 1000uF/35VW PC electrolytic
1 1000uF/25VW PC electrolytic
2 1000uF/16VW PC electrolytics
2 10uF/35VW tantalum electrolytics
3 10uF/25VW tantalum electrolytics
1 10uF/6.3VW PC electrolytic
4 0.1uF greencap (metallised polyester)
1 0.01uF greencap

RESISTORS (all ¼W 5%):
1 × 33k, 2 × 10k, 2 × 4.7k, 1 × 2.2k, 1 × 1k, 2 × 470 ohm, 1 × 100 ohm, 2 × 68 ohm ½W, 1 × 39 ohm, 1 × 22 ohm.

NOTE: Resistor wattage and capacitor voltage ratings are those used in our prototype. Components with higher ratings may be used provided they are physically compatible.



Terminate suitable lengths of rainbow cable to the PCB before installation in the case.

The +12V supply required by the 2708 is also tapped off the regulator by taking the voltage across zener diode D8. This is why two zener diodes were used in series rather than one single 24V zener, but it has the further advantage of reducing the dissipation by distributing the power dissipation in two 2.5W zeners rather than a larger and more expensive 24V zener.

CONSTRUCTION

That completes our discussion of the circuit. We can now discuss the construction of the programmer. The unit is assembled on two PC boards, 80pp7, measuring 140 x 127mm and 80pp7b, measuring 48 x 69mm. Most of the components are mounted on the larger PCB while the EPROM socket is mounted on the small PCB which is soldered at right-angles to the larger PCB.

Mount the links, ICs and other components on the main board first paying particular attention to the orientation of the ICs diodes and electrolytics. Next mount the EPROM socket on the socket PCB.

The EPROM socket should be a 24-pin zero-insertion force socket so as to avoid damage to the IC pins and ensure a long life for the socket. Two types of zero-insertion force sockets are available so the front panel artwork has been designed to accommodate either type. We understand that these sockets as well as most of the other parts for the programmer can be obtained from CQ Electronics, 95 Regent St, Sydney (or 30 Campbell St, Blacktown) and Radio Despatch Service, 869 George St, Sydney.

We housed our unit in a Horwood Instrument case measuring 228 x 76 x 230mm (D x H x W). The case has black Marviplate top, bottom and sides with aluminium front and back panels. Alternatively you can use a Pactec Instrument Housing model CH325 which is a particularly attractive unit though it is more expensive than the Horwood case.

Drill mounting holes for the major components and make a cutout in the back panel for a 25 pin male D-connector with solder lugs. This D-connector is required for the cable connection to the bus-expansion port on the back of the computer and it mates with a female D-connector which has been

Here is the full Software listing for the EPROM Programmer

mass-terminated with a length of rainbow cable. On the other end of the cable the individual wires have to be soldered to a card edge connector which then mates with the expansion port of the computer.

The connections which have to be made to the card edge connector will depend on which computer you are using. In the case of the Compucolor, Sorcerer and Tandy computers this information is given in the owner's manuals so we will not list them here. The only point to note is that a different size edge connector is required for each of them. The Compucolor and the Sorcerer both require 25/50 way edge connectors while the Tandy requires a 20/40 way edge connector.

Mass terminated 25-pin D-connectors are available from Radio Despatch Service, 869 George St, Sydney as well as the card edge connectors.

Now drill the holes in the front panel for the switches and LED bezels and make a cutout for the EPROM socket. Use the actual size artwork shown else where in this article to obtain drill centres etc. The artwork can also be used to make up a front panel from Scotchcal photosensitive aluminium or alternatively front panels can be purchased from Rod Irving Electronics or Radio Despatch.

The PCB can now be installed in the case using 10mm tapped spacers to support the board. Temporarily install the socket board with the zero insertion force socket into the front panel cutout and butt the main board against it. Then using a pencil, mark a line across the socket board, remove the two boards and using the line as a guide, solder the two boards together. Be careful to line up the connector strips on the boards and carefully solder two of the connector strips together. Check first to see that the boards will mount properly in the case then solder the remaining connector strips together.

With the main components mounted in the case, complete the wiring using the diagram shown elsewhere in this article. For a neat appearance use rainbow cable for the connections to the D-connector and the front panel switches.

CHECK VOLTAGES!

Rather then blow up an otherwise working EPROM your first check, after turning the programmer on, is to test that all the voltages on the power supply pins of the EPROM are okay. Some points to note when using the programmer is that the EPROMs should only be inserted into the programming socket when the power is off and the programmer has been switched to read mode. It is also good practice to check

```
10 REM
20 REM ELECTORNICS AUSTRALIA 2708 EPROM PROGRAMMER
30 REM WRITTEN BY RON DE JONG 22/5/80
40 REM FOR THE TRS-80 LEVEL II BASIC
50 REM
60 REM THE PROGRAM CONSISTS OF A NUMBER OF
70 REM INDIVIDUAL ROUTINES WHICH CAN BE CALLED UP
80 REM SEPARETLEY IF REQUIRED. THE ROUTINES ARE
90 REM LISTED BELOW:
100 REM 1. "2000" CHECKS TO SEE THAT THE EPROM
110 REM     HAS BEEN PROPERLY ERASED.
120 REM 2. "3000" READS DATA INTO MEMORY STARTING
130 REM     AT LOCATION 30001 FOR SUBSEQUENT LOADING
140 REM     INTO THE EPROM
150 REM 3. "8000" PROVIDES A HEX LISTING
160 REM     OF PREVIOUSLY ENTERED DATA.
170 REM 4. "4000" PROGRAMS DATA IN MEMORY INTO
180 REM     THE EPROM USING A MACHINE LANGUAGE DRIVER.
190 REM 5. "6000" IS A ROUTINE FOR LOADING DATA
200 REM     FROM AN EPROM INTO MEMORY FOR SUBSEQUENT
210 REM     PROGRAMMING INTO A BLANK EPROM. HENCE IF A
220 REM     BLANK RON IS LATER INSERTED AND RUN 4000
230 REM     EXECUTED THE FIRST EPROM WILL BE DUPLICATED
240 REM 6. "9000" IS AN EDITING ROUTINE WHICH
250 REM     ACCEPTS A HEX ADDRESS FOLLOWED BY A
260 REM     HEX DATA BYTE TO BE PLACED AT THAT LOCATION.
270 REM
1000 DEFINT A-Z
1010 GOSUB 2000
1020 GOSUB 3000
1030 GOSUB 8000
1035 GOSUB 9000
1040 GOSUB 4000
1050 Z=0:GOSUB 5000
1060 END
2000 INPUT "SWITCH TO READ MODE";A$
2010 J=INP(1):CR=0
2020 FOR I=1 TO 1024
2030 POKE 30000+I,255
2040 IF INP(0)<>255 THEN CR=1
2050 J=INP(2)
2060 NEXT I
2070 IF CR=1 THEN PRINT "EPROM NOT ERASED":END
2080 PRINT "EPRON ERASED"
2090 RETURN
3000 FOR I=1 TO 1024
3010 J=I-1:GOSUB 7050
3020 PRINT "0"+P$,
3030 A$="":INPUT A$
3040 IF LEN(A$)=0 THEN GOTO 3080
3050 GOSUB 7000
3060 K=J:GOSUB 7000
3070 POKE 30000+I,16*K+J
3080 NEXT I
3090 PRINT "DATA ENTRY COMPLETED"
3100 RETURN
4000 INPUT "SWITCH TO PROGRAM MODE";A$
4010 DATA 30,220,33,49,117,1,0,4,219,1,126,211
4020 DATA 0,211,2,62,54,61,32,253,211,1,35
4030 DATA 219,2,11,120,177,32,236,29,32,225,201
4040 FOR I=0 TO 33
4050 READ A:POKE I+32002,A
4060 NEXT I
```

EPROM Programmer listing contd:

```
4070 POKE 16526,2:POKE 16527,125
4080 J=USR(0)
4090 PRINT "EPROM PROGRAMMING COMPLETED"
4100 RETURN
5000 INPUT"SWITCH TO READ MODE";A$
5010 J=INP(1):I=1
5020 FOR Q=1 TO 9
5030 P$=RIGHT$(P$,2)
5040 FOR R=1 TO 15
5050 J=I-1:GOSUB 7050
5060 PRINT "0"+P$+"
5070 FOR T=1 TO 8
5080 IF Z=1 THEN J=PEEK(30000+I):ELSE J=INP(0)
5090 GOSUB 7050
5100 P$=RIGHT$(P$,2):B=PEEK(30000+1):PRINT P$;
5110 IF J=B THEN PRINT"    ";:ELSE PRINT"*   ";
5120 J=INP(2):I=I+1
5130 NEXT T
5140 PRINT
5150 IF I>1024 THEN RETURN
5160 NEXT R
5170 INPUT A$
5180 NEXT Q
5190 RETURN
6000 J=INP(1):INPUT "SWITCH TO READ MODE";A$
6010 FOR I=1 TO 1024:POKE 30000+I,INP(0)
6015 J=INP(2):NEXT I
6020 Z=1:GOSUB 5010
6030 RETURN
7000 IF B$<"A" THEN J=ASC(B$)-ASC("0"):RETURN
7010 J=ASC(B$)-ASC("A")+10
7020 RETURN
7050 K=INT(J/256)
7060 M=J-K*256
7070 L=INT(M/16)
7080 M=M-L*16
7090 N=K:GOSUB 7140
7100 P$=N$:N=L:GOSUB 7140
7110 P$=P$+N$:N=M:GOSUB 7140
7120 P$=P$+N$
7130 RETURN
7140 IF N>9 THEN N$CHR$(N-10+ASC("A")):RETURN
7150 N$=CHR$(N+ASC("0"))
7160 RETURN
8000 Z=1:GOSUB 5010
8010 RETURN
9000 PRINT "EDITING INPUT FORMAT AAAA DD"
9010 INPUT A$
9020 IF LEN(A$)=0 THEN RETURN
9030 GOSUB 7000
9040 K=J:GOSUB 7000
9050 K=16*K+J:GOSUB 7000
9060 K=16*K+J:GOSUB 7000
9070 K=16*K+J:GOSUB 7000
9080 A$=RIGHT$(A$,2):GOSUB 7000
9090 L=J:GOSUB 7000
9100 POKE 30001+K,16*L+J
9110 GOTO 9010
READY
```

that the right EPROM type has been selected before using the programmer.

Well that completes the hardware.

SOFTWARE DESCRIPTION

Now all that remains before we can start programming EPROMs is suitable software for the microcomputer. To this end we have provided a listing of a program (for 2708 EPROMs only) written in Tandy Level II BASIC. The timing for the program routine assumes a 2MHz CPU clock. It is possible to adapt this program for other microcomputers and different clock speeds. Also by changing the various constants in the program and adding extra routines, 2716/2758 and 2732 EPROMs can be programmed.

There are six separate routines which are called up in sequence by the main program during normal execution. These routines are as follows: "2000" checks that the EPROM has been erased; "3000" reads the data to be stored in the EPROM by first prompting with the hexadecimal address and then reading the hex data to be stored there – note that this data is stored in the computer's memory at this stage; "8000" lists the data which has been entered in the form of a hex dump with a hex address then eight hex data bytes per line. The listing is page by page so that each page of data can be individually studied and any errors noted.

UTILITY ROUTINES

If there are any errors these can be corrected in the next routine. This is an editing routine starting at "9000" which accepts input in the form of a four-digit hex address followed by a two-digit hex data byte which is to be stored at that location. Routine "4000" is then called and this loads and jumps to a machine language program that takes the data stored in the computer's memory and programs it into the EPROM. This routine is quite fast and as we have mentioned, programming time is about two minutes or less.

The programming routine is followed by a verification routine which provides a hex dump of the data contained in the EPROM and marks any data bytes which do not match the data stored in the computer's memory with an asterisk. (If there are any bad locations then the EPROM may be faulty.)

In addition to these, we have provided another routine at line 6000 which may be called up to read the data from an EPROM into the computer's memory and list it in the form of a hex dump. This is useful for duplication of EPROMs since if routine "4000" is then called this data can be programmed into any number of blank EPROMs which are subsequently loaded into the programmer.

Next month we shall feature a listing of this program for the Exidy Sorcerer.

(To be continued)