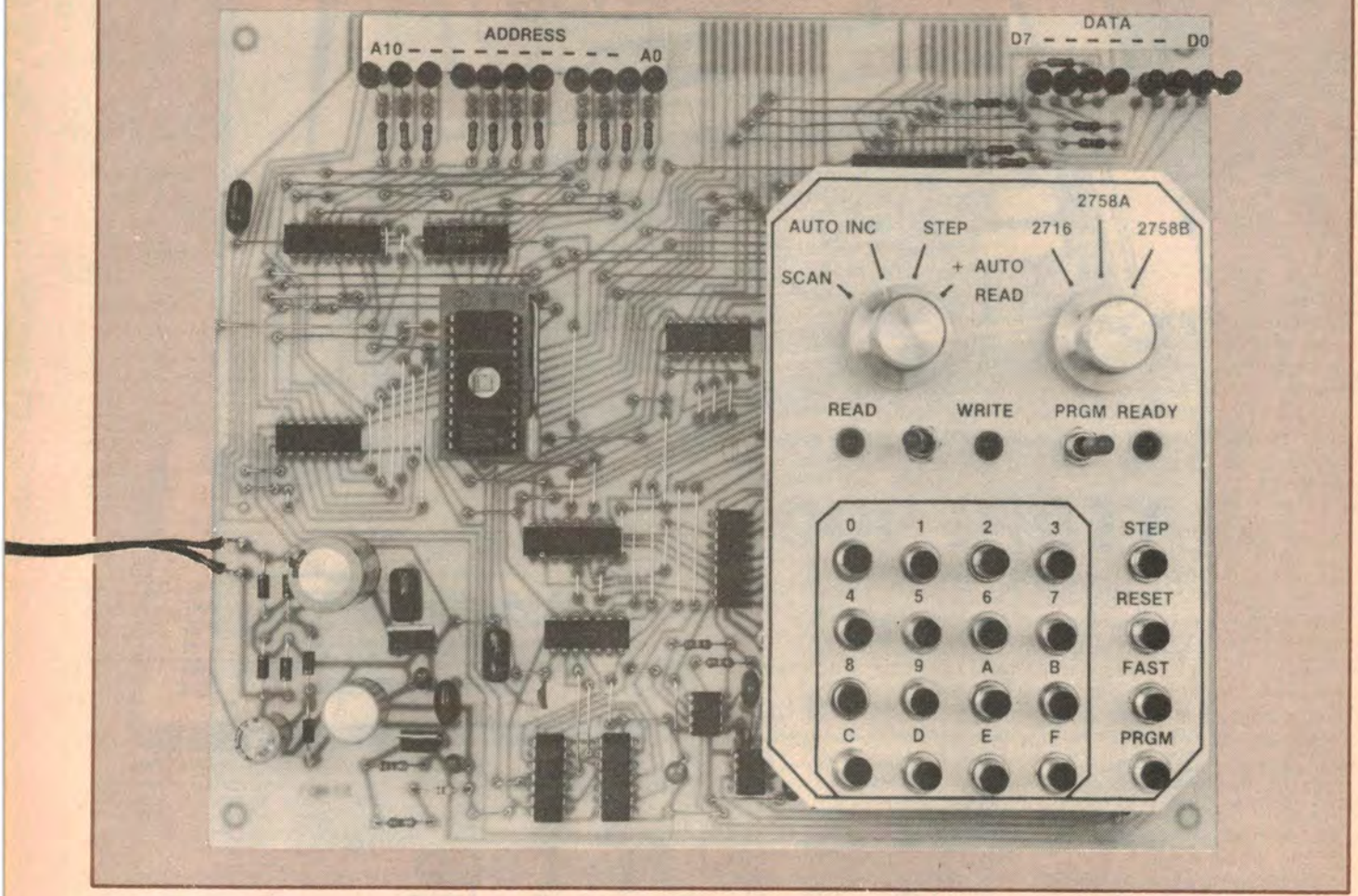
An easy-to-use EPROM Programmer



This simple and inexpensive EPROM programmer operates without the need for a personal computer or microprocessor system. Data entry is via a hexadecimal keypad and the address and data readout is indicated with LEDs in binary code. Programming is virtually as fast as the data can be entered and program verification is immediate. Suitable for 2716/2758 EPROMs, the programmer is construced on a single printed circuit board and has provision for future expansion.

by JOHN CLARKE

Electrically Programmable Read Only Memories (EPROMs) are ideally suited for development work, where the final contents of the ROM may have to be arrived at by trial and error. They have the advantage of being a permanent memory but they can be easily erased and new or revised data added.

Some of the many uses for an EPROM are to be found in the exciting area of dedicated control. Robotics, industrial controllers, train controllers, speech synthesisers, burglar alarms, light shows, display drivers and code converters benefit greatly in terms of versatility when using an EPROM. In fact, complete redesigns can be made simply with modifications to the EPROM program. In most cases no hardware changes are required. Of course, EPROMs are also used widely in small system computers and microprocessor applications as a permanent memory storage for dedicated programs and operating systems.

One of the major problems with using EPROMs has been with programming. Many of the commercial EPROM programmers available contain microprocessors. As such, they require programmed EPROMs to provide the software necessary to enable the microprocessor to function. For constructors, this presents a problem since an EPROM needs to be programmed before the programmer can be completed. A tricky situation to be sure.

An alternative is a programmer designed to be controlled by a personal computer (such as our EPROM

Programmer for the TRS-80, July/August, 1980) but these require special software and not everyone has access to a personal computer.

Up till now, EPROM programmers virtually demanded the assistance from a computer or microprocessor to perform the repetitive cycling required of the popular but now outdated 2708 EPROM. In fact, for this EPROM it is necessary to apply a short programming pulse between 0.1 and 1ms long to each memory location in sequence. Once all the memory addresses have been accessed and each pulse applied, this entire sequence has to be repeated again for many loops. The minimum number of loops required is 100 when the pulse width is 1ms, and 1000 loops with a 0.1ms pulse width.

Obviously this procedure would be very tedious if it were attempted manually, perhaps taking many days.

With the advent of the newer 2716 type of EPROM, programming has been made considerably easier. Any location can be programmed at any time either individually, sequentially or randomly. A single TTL 50ms pulse is all that is required to perform the complete programming of each location. It is not necessary to repeat the process as in the 2708 EPROM.

In short, to program the 2716 EPROM all that is required is to apply 25 volts to the Vpp pin of the IC, select the address, set up the data required, apply the 50ms pulse and the EPROM is programmed in that location. Programming manually is almost as fast as the data can be entered.

Obviously, with the ease that the 2716 EPROMs can be programmed, a simple circuit can be made to program them. We wanted to produce a simple and inexpensive programmer which requires no assistance from a personal computer, nor from a preprogrammed EPROM. In particular we wanted it to be capable of programming not only the 2716, but the 2758 EPROMs as well.

The 2758 is a 1K version of the 2716, a 2K EPROM, but with one half of the memory defective. Rather than throw these devices away, manufacturers label them 2758A or 2758B depending on which half of the die is defective. These have an advantage over the 2708 in that they have the easier programming method and are a single supply EPROM (the 2708 needs −5, +5 and +12 volts) like the 2716.

The unit is easy to construct, inexpensive and yet does not lack in features. It can read the contents of the EPROM to check that it has been erased before programming and check that the EPROM does in fact contain the correct data after programming. As well, there is a special fast scan which enables a certain address to be accessed quickly.

Extra features include automatic increment of the address when programming and an automatic check of the EPROM contents directly after programming each location.

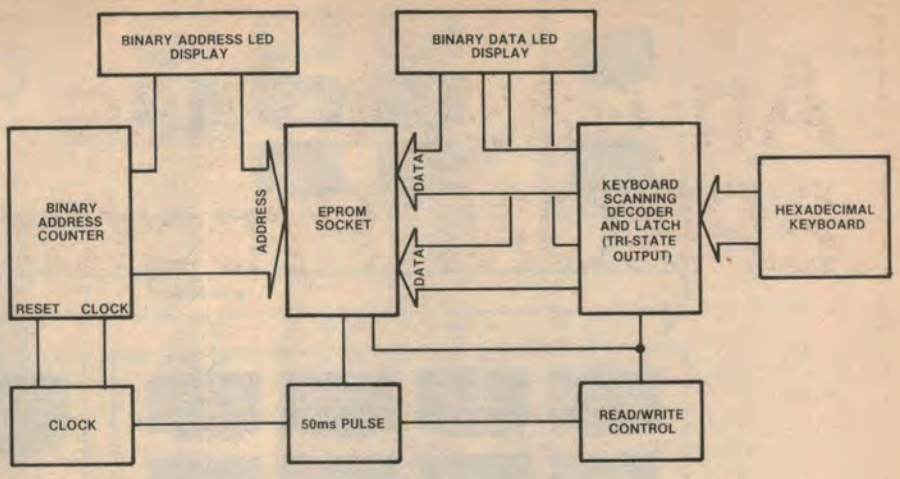


FIG. 1

Fig. 1: block diagram for the new EPROM Programmer. Data is entered via a keyboard and decoded to 8-bit binary data with the keyboard scanning decoder.

Although programming a single location can be achieved independently of other memory locations, erasure (with UV light) of the data affects the entire memory contents. If the data at one address was to be altered, the complete memory contents would need to be erased and re-entered with the revised data. The exception to this is when all the data required to be changed is from a "1" to a "0", in which case the 1 can be reprogrammed to 0.

In fact, this explains the concept of the EPROM: erasing brings all the bits high, "1", and programming can only send the bits low, "0".

HEXADECIMAL AND BINARY CODES

Many readers who have seen programming lists for EPROMs will have noticed that the address and data are generally shown in hexadecimal. The hexadecimal number is really just an easier and shorter means for expressing a 4-bit binary number which has 16 possible codes. Since our programmer only displays the address and data in binary, the code will have to be converted. Fig. 2 shows the conversion table for the hex and binary codes.

The LED readouts are arranged in blocks of four. This helps with decoding since each block is read as a hexa-

PARTS LIST

1 PCB, 230 × 188mm, coded 82ep1
1 Scotchcal label, 117 × 140mm
2 knobs
1 sheet of aluminium, 117 x 140mm
6 rubber feet
1 24-pin IC socket
20 pushbutton switches
1 SPDT switch
1 DPDT switch
1 double-pole, three position rotary switch
1 double-pole, four position rotary switch

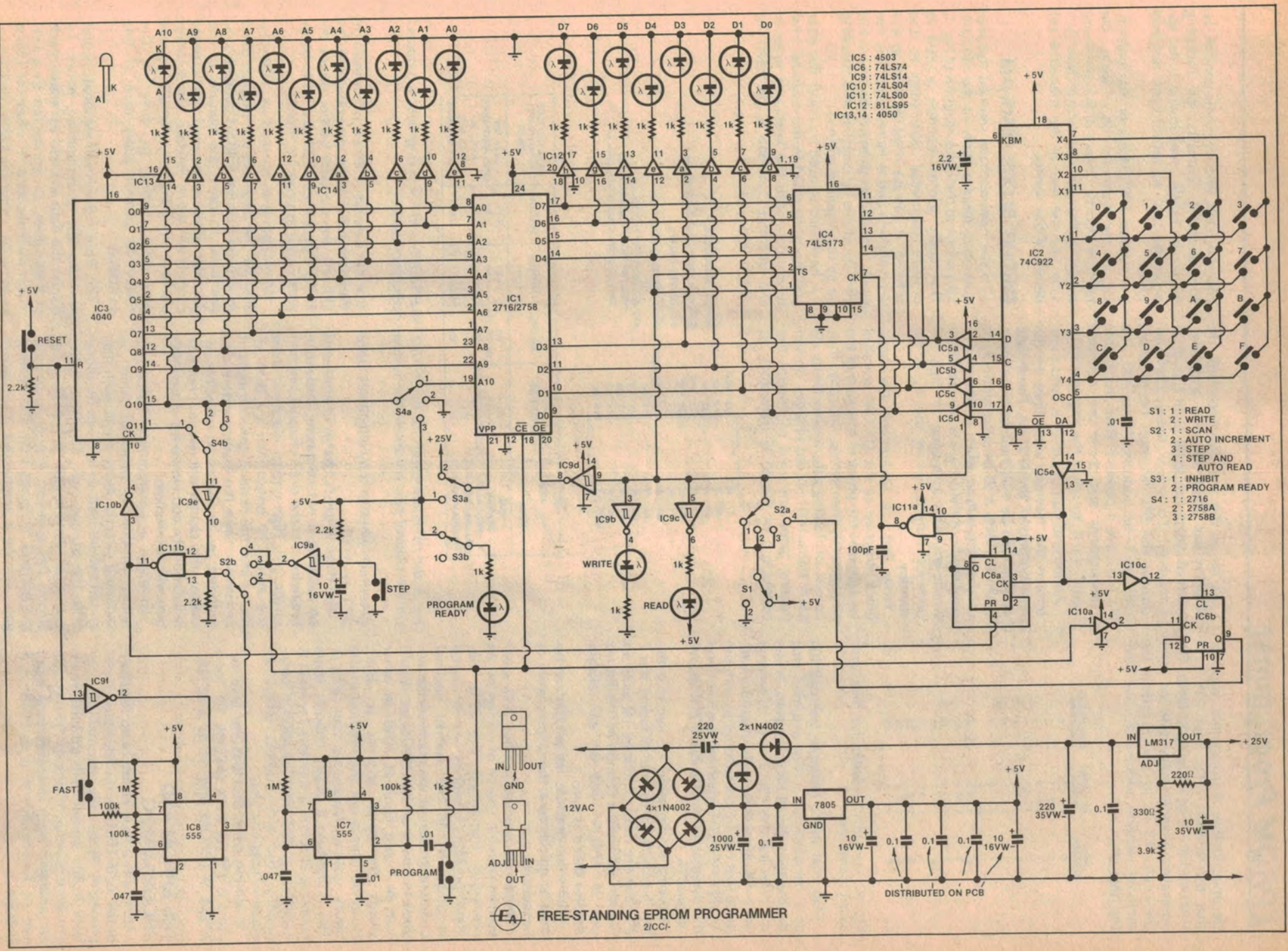
SEMICONDUCTORS
1 74C922 16-key encoder
1 4040 12-stage ripple-carry binary counter
1 4503 hex non-inverting Tri-state buffer
2 4050 hex non-inverting buffers
1 74LS173 Tri-state quad D register
1 74LS74 dual D flipflop
1 74LS14 hex Schmitt trigger
1 74LS04 hex inverter
1 74LS00 quad NAND gate
1 81LS95 Tri-state octal buffer
2 555 timers
1 7805 5-volt 1A regulator
1 LM317T adjustable three-terminal regulator
6 1N4002 1A silicon diodes
22 5mm diameter red LEDs

CAPACITORS
1 1000µF/25VW electrolytic
1 220µF/35VW electrolytic
1 220µF/25VW electrolytic
1 10µF/35VW electrolytic
3 10µF/16VW electrolytic
1 2.2µF 16VW electrolytic
5 0.1µF metallised polyester
2 0.047µF metallised polyester
3 0.01µF metallised polyester
1 100pF ceramic

RESISTORS (¼W, 5%)
2 × 1MΩ, 3 × 100kΩ, 1 × 3.9kΩ, 3 × 2.2kΩ, 23 × 1kΩ, 1 × 330Ω, 1 × 220Ω

MISCELLANEOUS
Tinned copper wire, solder, etc.

Note: Components specified are those used in the prototype. In general components with higher ratings may be used provided they are physically compatible.



FREE-STANDING EPROM PROGRAMMER

2/CC/-

decimal number. For example, suppose that the address LEDs read 101 1111 0101 and the data LEDs, 0011 0000. The equivalent hexadecimal would be, using the table, 5F5 30. In other words in the memory location address 5F5, we have the data 30.

HEX NUMBER	4-BIT BINARY WORD	EQUIVALENT DECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Fig. 2: conversion table for hex and binary numbers.

Note that, in the example, we read 101 as a hexadecimal 5. We assumed that the most significant bit was a zero. In fact we have left this LED out of the circuit since it is unnecessary. The maximum address that can be counted up to with the 2K EPROM is 2047 decimal. If we convert this decimal number to hexadecimal, we obtain 7FF.

Looking to the table we see that the hex numbers up to and including the 7 all have the most significant bit as 0. So we do not have a change in this bit for our 2K EPROM and we can always assume that it is zero.

With some practice, these LEDs can be read as quickly as if the display were in hexadecimal.

Before we proceed further, let's clear up one possible source of confusion. As previously mentioned, the 2716 has 2048 possible locations, or in digital language 2K. In the normal decimal counting system, if we counted the address locations of the EPROM we would start at 1 and finish at 2048. In other words the final address is 2048. However, this is not the method in the binary system. We always start from 0. This is because a binary counter starts counting at 0 and a memory location is available at this zero location. Consequently, the final address when we count in binary is 2047.

It follows from this that the final address is 7FF for a 2K EPROM, 3FF for a 2758A 1K EPROM, and 7FF for a 2758B.

THE EPROM AND PROGRAMMING

Let's take a look at the basic structure of the 2716 EPROM to understand these processes more fully. Internally it contains 16,384 storage cells (or bits) which can store either a "1" or a "0". These cells are arranged eight bits wide (one byte) and so functionally it is a 2048 × 8 EPROM.

As far as the user is concerned, the EPROM has 2048 effective addresses, each of which can store eight bits of information. To select these 2048 addresses the EPROM must be provided with an 11-bit address, A0 to A10. The 11 bits have a possible 2048 truth table combinations and these are decoded within the IC to select the required byte.

An internal output buffer is used to interface the memory data lines to the data outputs of the EPROM. These data outputs can be switched from an input when the data is to be programmed, or used as the output when in the normal read mode. A third state is also available which provides a floating or high impedance output, when the IC is on standby.

Now that we understand the basic block structure of the EPROM we shall venture into the physical structure of the storage cells themselves. Basically, they consist of floating-gate avalanche-mode MOS transistors. 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, a quartz window is provided which is mounted directly above the IC such that an ultra-violet light source can be shone onto the storage cells. The light at the particular wavelength of 2537 Angstroms has sufficient energy to release the trapped electrons from the floating gate, leaving it uncharged. The erased memory cells effectively all contain 1s. Programming consists of inducing avalanche mode breakdowns in the appropriate cells to provide the required zeros.

A suitable EPROM eraser was published in our February 1979 issue, file number 2/cc/36. Copies of this article are available through our Information Service. See the rear of the magazine for details of this service.

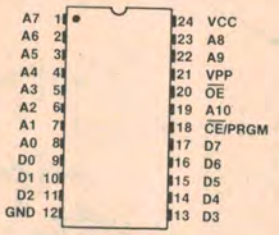
CIRCUIT BLOCK DIAGRAM

To see how we have satisfied the programming requirements of the EPROM, take a look at the block diagram of Fig. 1. The binary address counter is used to provide sequential addressing to the EPROM. It can be reset to start at the zero address and is incremented by the clock. This allows us to access any of the 2048 locations by resetting and counting up to the required address.

A keyboard is used to enter the data and this is decoded to the required 8-bit binary data with the keyboard scanner and decoder. Latching is provided so that the data just entered will remain on the data bus. When programming the EPROM, the decoder output drives the EPROM data lines which are set to be inputs (write or program mode).

When data is to be read out from the EPROM, the data lines are outputs (read mode). So that the EPROM data lines do not have to drive the outputs of the decoder when in the read mode, these outputs are set to Tri-state. The Tri-state refers to a floating output which requires very little current to drive it. The read/write control takes care of this switching.

The 50ms pulse is self explanatory and provides the programming pulse. To display the data and address states, each of these lines is connected to a LED which will indicate a "1" when the LED is lit and a "0" when the LED is unlit.



A0-A10 : ADDRESSES
$\overline{CE}$/PRGM : CHIP ENABLE/PROGRAM
$\overline{OE}$: OUTPUT ENABLE
D0-D7 : DATA OUTPUTS
A10: ADDRESS FOR 2716 (2K × 8)
HELD AT VIL FOR 2758A (1K × 8)
VIH FOR 2758–S1865 (2758B) (1K × 8)

PINS / MODE	$\overline{CE}$/PRGM (18)	$\overline{OE}$ (20)	VPP (21)	VCC (24)	DATA OUTPUTS (9-11) (13-17) D0-D7
READ	VIL	VIL	+5V	+5V	DATA OUT
STANDBY	VIH	VIL OR VIH	+5V	+5V	HIGH IMPEDANCE
PROGRAM	PULSED VIL TO VIH	VIH	+25V ±1V	+5V	DATA IN
PROGRAM VERIFY	VIL	VIL	+25V ±1V	+5V	DATA OUT
PROGRAM INHIBIT	VIL	VIH	+25V ±1V	+5V	HIGH IMPEDANCE

Fig. 3: the pin connections for the 2716/2758 EPROMs and the voltage levels required on the control pins to obtain the various operation modes.

THE CIRCUIT

Fig. 3 shows the pin connections for the 2716/2758 EPROMs and the various voltage levels required on the control pins to obtain the operation modes. In our programmer, we have used all modes of the EPROM with the exception of the standby mode. To see how we have met the specifications for programming, program verify and read, we shall now discuss the circuit.

Heart of the circuit is IC1, the EPROM, or more correctly the EPROM socket. Most of the address lines are directly connected to the 4040, IC3, a 12-bit binary counter. The exception is A10 which is connected to Q10 of the counter only when switch S4a is in position 1. This position is suitable for the 2716 EPROM. The other two positions of the switch connect A10 to either the positive or ground rail, to provide for both versions of the 2758 EPROM.

IC3 is controlled by two inputs, the reset and clock. Resetting is effected with the reset switch by pulling pin 11 high. This brings all the outputs of the counter low. The reset is normally held low with the 2.2kΩ resistor to prevent false resetting. The clock input is gated with IC11b, a NAND gate, and inverted with IC10b. The reason for this inversion shall become clear later. Clock signals, selected with S2b, are sent to pin 13 of IC11b. This signal can pass to the output of IC11b only when pin 12 is high.

The pin 12 input of IC11b is normally held high with the inverter IC9e. The input to IC9e, pin 11, is normally held low with Q11 (pin 1) of IC3, when S4b is in position 1. When the switch is in positions 2 and 3, Q10 connects to IC9e. When IC3 finishes the count of 3FF hex or 1023 decimal, Q10 goes high and indicates the end of memory for the 2758A EPROM. The inverter, IC9e, will go low preventing any further clock pulses to IC3. The address outputs will now remain in this state, all other outputs low and Q10 high until the counter is reset.

Similarly, when S4b is in position 1, the clock will be stopped when Q11 goes high at the finish of the 2047th count or 7FF hex.

Data entry to the EPROM is a little more complex. A 4 × 4 matrix keypad is used to enter data and IC2 is used to scan and encode the keypad. The capacitor connected at the oscillator input, pin 5 of IC2, sets the rate at which the keypad is scanned. When a key is pressed, IC2 determines which contacts are closed and presents the 4-bit binary code at the data outputs, pins 14 to 17.

This data is buffered with the Tri-state buffers, IC5a to IC5d. After a key debounce period, set by the capacitor connected to KBM, pin 6, the Data Available output, DA, pin 12 (buffered by IC5e), goes high.

IC6a, a D flipflop, is connected to divide by two and is clocked by the output of IC5e. At each positive-going clock signal the $\overline{Q}$ output changes state from a low to a high or from a high to a low. Assuming that initially $\overline{Q}$ is clocked high when DA goes high, this holds pin 9 of IC11a high. Since IC5e, pin 13, is also connected to IC11a, pin 10, both inputs will be high and the output of IC11a will go low.

As soon as the pressed key of the keyboard is released, the DA goes low and the output, pin 8, of IC11a goes high, clocking the latch, IC4. The input to the latch is transferred to the output when the clock goes high and so the data entered on the keyboard is transferred to the output of the latch and remains there until reclocked.

Upon pressing a key on the keypad again, the DA goes high, clocks $\overline{Q}$ low and this low prevents the output of IC11a from going low for the second time. Subsequently the data just entered remains at the outputs of IC5a to IC5d. The 100pF capacitor at the output of IC11a prevents any glitches causing false latching in IC4.

Now we have the 8-bit data word at the data lines of IC1. The first time the keypad is pressed, the most significant 4-bit word is set, D7 to D4, and the second time the keypad is pressed the least significant 4-bit word is loaded to the data bus, D3 to D0.

IC12 is an octal Tri-state buffer which is used to buffer each of the data lines. The outputs of these buffers drive the data LEDs via 1kΩ resistors. Similarly, the address LEDs are driven by hex buffers IC13 and 14, connected to the address lines.

So far we have discussed the address and data inputs to IC1 and the readout LEDs. This constitutes the major section of the circuitry. The remainder of the circuit consists of the various clocking options available, the programming controls and the power supply.

We shall now describe the clocking modes available by selecting one of the four positions on switch S2b.

Position 1 on S2b allows the astable clock (IC8) to provide a scan through all the memory addresses. The frequency of this clock is around 26Hz and so about 82 seconds are needed to count through all the memory addresses of a 2K EPROM. A fast control, which connects a 100kΩ resistor across the 1MΩ resistor tied from pin 7 to the positive rail, increases the clock rate by about five times and considerably shortens the time to scan the memory locations. When all the memory locations have been accessed, the scanning will stop.

The reset, pin 4 of IC8 (a 555 timer), is connected via IC9f to the reset switch of IC3. Resetting IC8 when IC3 is reset ensures that a full clock cycle is provided for the first memory location before the next memory location is accessed.

Digressing for a moment, we need to discuss the other parts of the circuit before returning to the clocking modes.

IC9b and IC9c are used to drive the write and read LEDs respectively via 1kΩ current limiting resistors. The inputs to these inverters are both connected to the Tri-state controls of IC4 and IC5. Another inverter, IC9d, also has its input connected to the Tri-state line and the inverted output is connected to $\overline{OE}$ of IC1.

When the Tri-state line is high, the read LED is lit, IC1 is in the read mode and the data lines of IC4 and IC5 are in Tri-state. When the Tri-state line is low, the write LED is lit and IC1 is in the program mode. The outputs of IC4 and IC5 are then in the normal output mode.

The Tri-state line is controlled with S2a. When the switch is in positions 1 to 3, S1 determines whether this line is high or low by switching the line to +5 volts or to ground. When S2a is in position 4, the control is effected by the Q output of IC6b. Now let us explain how the programming pulse and the step waveforms are produced.

S3a is used to provide either 5 volts or 25 volts to the Vpp input of IC1. S3b drives the PRGM Ready LED when the voltage at Vpp is 25 volts.

IC7 is used to provide the 50ms programming pulse. Initially, the 0.01μF capacitor connected to pin 2 of the 555 is charged positive on both sides. The 1KΩ and 100kΩ resistors hold both sides of the capacitor high. When the PRGM switch is closed, the 1kΩ side of the capacitor is brought to ground.

Now both sides of the capacitor are effectively at ground potential and pin 2 is brought low, triggering the 555. The capacitor begins to charge through the 100kΩ resistor and the low at pin 2 is gradually brought high. When the PRGM switch is released, the capacitor is discharged.

When the 555 is triggered, the output, pin 3, goes high for the time period of 1.1 × 1MΩ × 0.047μF. The programming pulse is therefore about 52ms, which is well within the range of 45 to 55ms (50± 5ms) necessary for correct programming of the EPROM.

The Step switch is used to provide a clock signal. IC9a is a Schmitt trigger and

We estimate that the current cost of components for this project is

$40

This includes sales tax but does not include the plugpack, ZIF socket, or EPROM.

the input, pin 1, is held high with the 2.2kΩ resistor. Closing of the Step switch immediately brings pin 1 low and the output of the Schmitt goes high. The 10μF capacitor provides debouncing for the switch.

A Schmitt trigger is used to ensure a clean square output waveform in spite of the slow rising of the capacitor voltage when the switch is released.

Refer now to the waveforms of Fig. 4. When S2b is in position 3, the address is incremented at each closing of the Step switch. The falling edge of the clock (Step) clocks the address counter. The program pulse will program the EPROM providing the $\overline{OE}$ is high (write) and Vpp is at 25 volts.

Comparing these waveforms to the table of Fig. 3 shows that the programming conditions are satisfied. When the $\overline{OE}$ is low the programming can be verified and when Vpp is at 5 volts we are in the normal read mode.

The Step + Auto Read mode is a little more complicated and occurs with S2 in position 4. S2b still allows control of the address counter with the Step switch and this is seen as the first waveform under the Step + Auto Read heading of Fig. 4. S2a connects the Q output of IC6b to the Tri-state line.

Initially, the Q output is low and so $\overline{OE}$ of IC1 is high. After the programming pulse, the falling edge (a rising edge after the inverter IC10a) clocks the data at D (+5 volts) to the Q output. The $\overline{OE}$ goes low and we are now in the read mode, reading the just programmed data. When the keypad is pressed, to enter the data for the next location, the DA output of IC2 clears the Q output of IC6b and we are in the write mode again.

Clearly, this mode is quite useful. The just programmed data can be immediately verified before the next location is programmed.

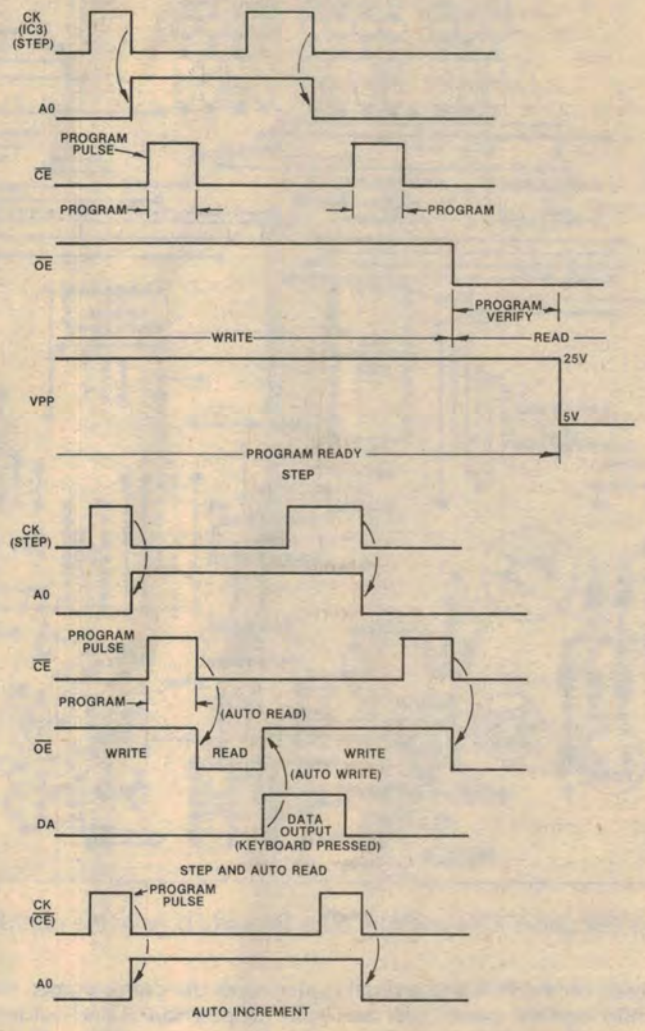


Fig. 4: this diagram shows the waveforms for the various programming modes.

Position 2 of S2b connects the output of IC7, pin 3, to control the clocking of the binary counter, IC3. This is shown in Fig. 4 under the heading of Auto Inc. On the negative edge of the programming pulse directly after programming, the address is advanced by one memory location. To ensure that this is so, we have included the inverter IC10b. In this mode the data can be entered, the programming pulse applied, and the data for the next location entered without regard for incrementing the address, as this is done automatically.

One final point. The output of IC11b is directly connected to the preset pin, pin 4, of IC6a. In doing this, it is ensured that after each clocking of the address, IC4 is ready to latch the first data entered on the keyboard.

This at first may appear unnecessary, since IC4 will latch at every second entry of 4-bit data. However, say we want to enter data which has both items of 4-bit data the same. For example, if we want to enter FF, the first depression of the keyboard, will enter FF. Remember that the first data entry always enters the same data for both 4 bits. The second depression enters the least significant 4-bit data.

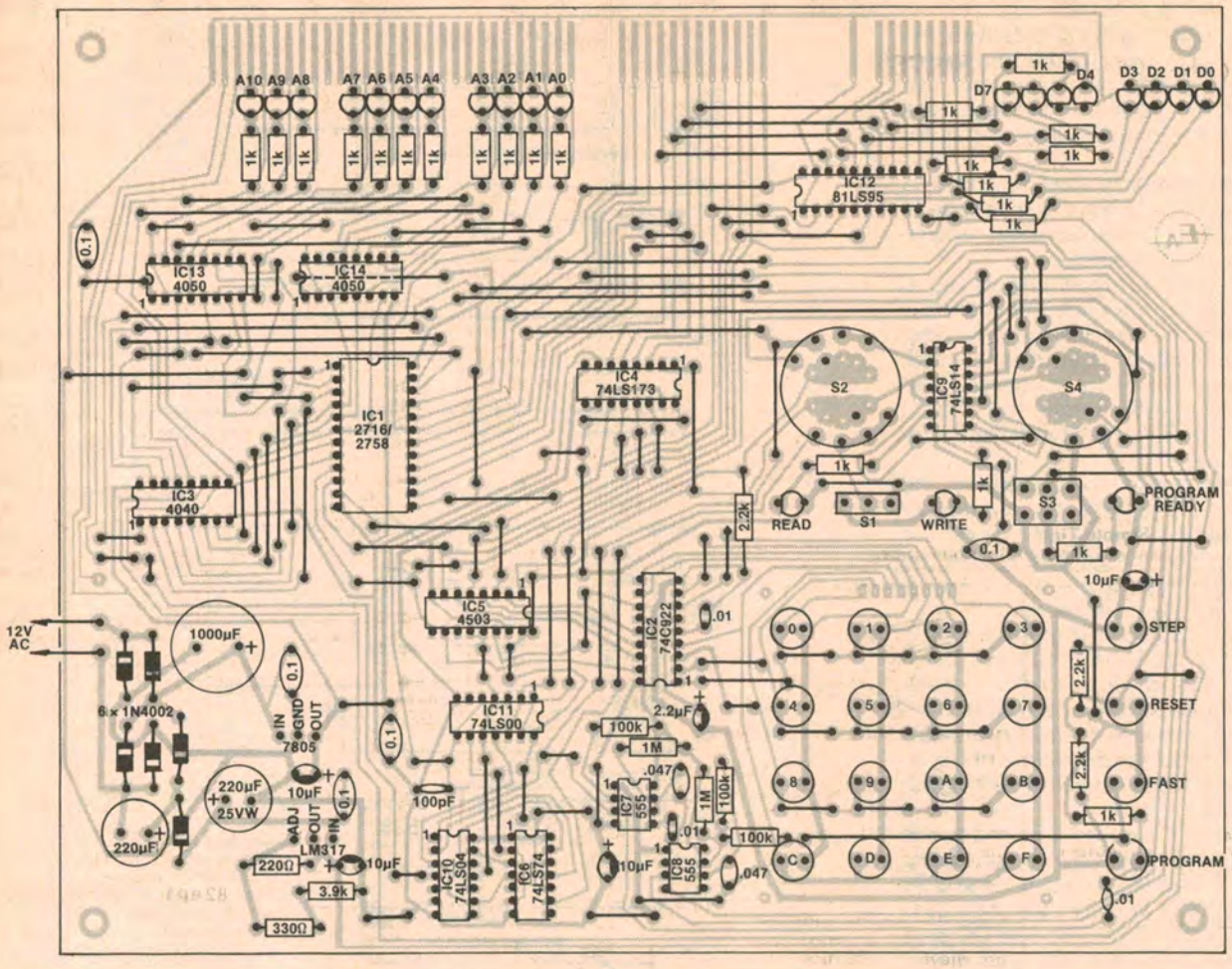
However, what if you, the operator, only pressed the keypad once to enter FF? The program switch would be pressed and the next data entered. At the change in address, the latch is ready to latch the first data. If we did not have this feature, at the next address change the whole data entering would be out of sequence.

Power for the circuit is derived from a 12 volt AC, 500mA plugpack which feeds a full-wave bridge rectifier, and a 1000μF filter capacitor. A three-terminal regulator regulates the voltage to five volts. The 0.1μF capacitor at the input and the 10μF capacitor at the output provide stability and ripple rejection respectively. The remainder of the capacitors connected to the output of this regulator provide decoupling of the supply around the circuit.

A voltage doubler is used to provide about 30 volts at the input to the LM317 adjustable three-terminal regulator. The 3.9kΩ and 330Ω resistors in series to the ADJ terminal and the 220Ω resistor from this terminal to the output provide a well-regulated supply of 25 volts. The 10μF capacitor at the output of the regulator improves the transient response of the regulator.

CONSTRUCTION

All the circuitry for the EPROM programmer is accommodated on a single-sided printed circuit board (PCB), measuring 230 × 188mm and coded 82ep1. All the switches and LEDs are



Note the wire link under IC14 and that LEDs D4 and D5 have the opposite orientation to D6 and D7.

directly mounted on the PCB and a small (140 × 100mm) control panel can be made to fit over the switches. A Scotchcal front panel label has been produced and the overall arrangement of the PCB and label can be seen in the photograph.

The PCB is designed to accept the Digitran keypad as well as separate pushbutton switches for data entry. The keypad will fit directly on to the PCB if the four corner pads are drilled and the eight pins from the keypad fitted into the matching pads on the PCB. Separate pushbuttons are used for the Step, Reset, Fast and PRGM controls. We used a Zero Insertion Force (ZIF) socket for the EPROM, although an ordinary 24-pin socket could also be used.

Note that a pad is left spare on the central left hand side of the PCB and a change of linking in this area can be used for an expansion board which may be published at a later date. The bus brought out on the top of the PCB is also for future expansion.

Start construction by mounting all the links, diodes and resistors. Be sure that the diodes are oriented correctly. Use the overlay diagram to help you in the placing of the components. Next the ICs can be positioned and soldered. Solder the supply pins of the CMOS ICs first with the barrel of the soldering iron earthed and connected to the negative rail of the PCB. Solder the pins quickly to prevent heat from damaging the IC.

CMOS ICs can be recognised by their 4000 or 74C00 series type numbers.

The capacitors and three terminal regulators can now be mounted as well as the LEDs. Try to keep the address and data LEDs all at the same height, as they look better that way. Be careful when soldering the data LEDs since two of them are oriented differently to the others. Note that the electrolytic capacitors must be inserted the correct way around as they are polarised components.

The switches can now be mounted on the PCB. We found it easier to only solder one of the legs of the pushbutton switches first, keeping the lug of the switch almost flush with the base of the PCB. The second lug of the switch can be soldered after the small front panel is fitted.

The rotary switches will need the eyelets cut off the lugs as well as all the unused lugs completely removed (by cutting with side cutters). Before cutting the lugs check the switch operation to make sure that you remove the correct lugs. Use a multimeter on one of the "ohms" ranges and go through each switch movement. When soldering each switch, have the body flush with the PCB. Note that several holes are available for the centre wiper lugs of each switch. This should allow all types of rotary switch to be fitted.

Finally the Read/Write and PRGM switches can be mounted, flush to the PCB. The LEDs for the indications on the front panel are best soldered low on the PCB on one leg only, but do not cut the leads yet.

Fashion the front panel from a piece of sheet aluminium and apply the Scotchcal label to it. Spray the label with a hard setting lacquer to protect the label from being scratched. Drill holes for the switches and LEDs. If the Digitran keypad is to be used, this can be positioned on the front panel so as to cover the locations of the 16 switches. A slot will need to be made for the eight pins for the keypad and stiff wires brought to the PCB.

Place the front panel over the switches

and tighten up the switch nuts on the panel. The LEDs can be lined up with the front panel by melting the solder of the soldered leg and positioning the LED. The remainder of soldering can now be completed. Solder the unsoldered lugs of the switches first and then reheat the other lugs to remove any mechanical stress that may have been introduced when the front panel was positioned.

Rubber feet should be placed on the corners of the PCB to support it and protect the surface on which it is used.

The A10 to A0 and D7 to D0 Scotchcal labels can be applied to the PCB above the LEDs and the construction is complete. Connect the 12-volt plugpack to the PCB and the unit is now ready to be tested.

TESTING AND OPERATION

Without the EPROM inserted, apply power. Check the voltages at the supply pins of all the ICs, to ensure that they are at ground and +5 volts. Check also that pin 21 of the IC1 socket can be switched from +5 volts to +25 volts with the "PRGM Ready" switch. When the 25 volts is applied, the LED should light. Check also that pin 20 of IC1 is at 0 volts when the Read/Write switch is in the Read position and that the Read LED lights. Alternatively, in the Write position, the Write LED should light and pin 20 should be at +5 volts.

The 5-volt regulator will run quite hot in normal use. However, the temperature at which it runs is well within the specifications of the device.

The +25 volt supply should be between 24 and 26 volts. Anything other than within this range is unacceptable. If the voltage is just outside these limits, then the voltage should be trimmed by adjusting the value of the 330Ω resistor connected to the ADJ terminal of the LM317. Increasing this value will increase the voltage and decreasing the value will decrease the voltage.

Now the operation of the remainder of the controls can be checked. Turn the switch to Scan and the address LEDs will increment: A0 will flash at the highest frequency, A1 will be half this frequency and so on. Pressing the Fast switch will increase the Scan rate.

The counting can be reset with the Reset switch and the counter will stop at 7FF if the EPROM select switch is in the 2716 and 2758B position and 3FF when in the 2758A position. In the case of the 2758B position, the A10 LED will always be lit.

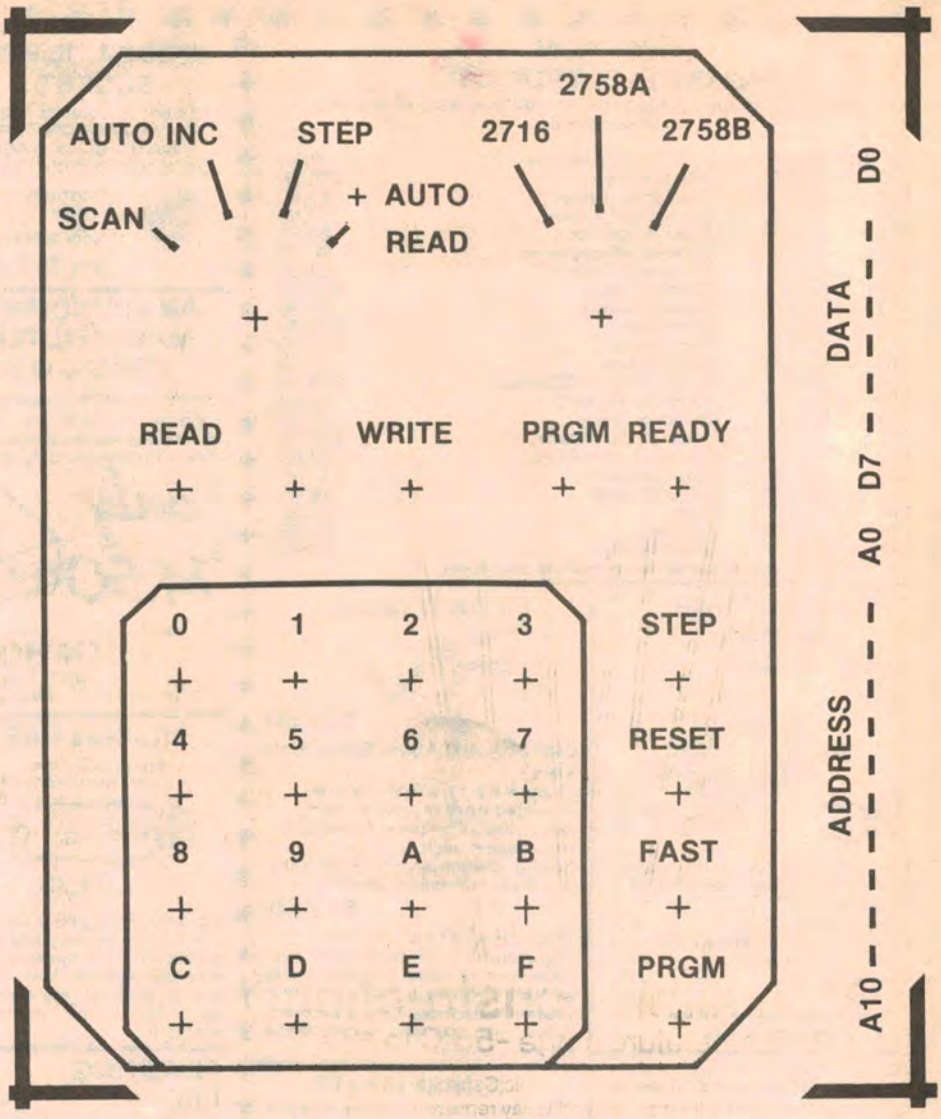
When the addresses are being scanned it is possible to observe the data LEDs, with the Read/Write switch in the Read position. It is easy in this Scan mode to see if the EPROM is erased since all the data LEDs should be permanently lit. For a more detailed examination of the address data in each address, turn the Mode switch to Step. The counter can be Reset and the addresses incremented one by one.

With the Read/Write switch in the Write position, check that the correct LEDs light with the hex keypad. Note that at the first depression of a key, both 4-bit data LEDs will show that code for the key pressed. On the second depression, the least significant 4-bit LEDs will show the code for the second key pressed, and the most significant 4-bit data LEDs will remain lit with the previously entered first key entry.

The Auto Inc and Step + Auto Read positions are used for programming. In the Auto Inc position, the address will increment at each depression of the PRGM switch. In the Step + Auto Read position, control of the Read/Write LEDs is automatic. Upon entering data, the Write LED will be lit and data can be programmed into the EPROM. As soon as the PRGM pulse is completed, the Read LED will light and the programmed location will be verified since we are reading the programmed data.

Now the erased EPROM can be inserted into the socket. Reset the address and with the mode switch in the Step position read the contents of the first address of the EPROM. Enter the required data with the keypad, apply the PRGM pulse, switch to Read and check that the location was programmed. Now that you have programmed the first location, the Step switch can be used to increment the address and the next location programmed. Of course, any of the three modes can be used for programming.

One final point: whenever the programmer is first switched on, always make sure that the PRGM Ready switch is off before power is applied. This ensures that no false programming occurs; it is always a good idea to keep this switch off until programming is required.



Actual size artwork for the front panel. The PCB artwork has been omitted because it is too big to satisfactorily fit on a single page.