VERSATILE EPROM EMULATOR

If imitation is the sincerest form of flattery, your EPROMs are going to have a lot to blush about. Design by Mike Bedford.

Microprocessor systems may be divided into two main categories. The first group, which will be familiar to all home computer enthusiasts, is usually referred to as a personal computer and is comparatively highly priced.

ch systems contain a large amount of memory, mostly RAM and a wide variety of I/O connected to devices such as keyboards, VDUs etc, making them very versatile pieces of equipment which may be programmed to carry out an almost infinite variety of different tasks. The second group may be described as minimal microprocessor systems and are used for control applications. Even domestic appliances now include such systems as their cost compares favourably with that of dedicated digital electronics. Such a system is designed to do one specific task and for this reason has less memory than systems in the first group, most of this memory being ROM or EPROM, and the I/O is not designed to interface with normal computer ripherals.

This brings us to the question of how software is developed for such dedicated control systems. To put it simply, this may be carried out on the system itself or on a separate development system. If the control computer itself is to be used it will have to be given some facilities additional to those required to carry out its final task. This is obviously out of the question in the commercial world where the extra cost would be prohibitive. In the amateur world, however, this approach has generally been used, the board having a monitor EPROM and interface to a keypad and LED display.

If software could be developed on a separate computer the availability of editors, assemblers, compilers and hardware such as displays and mass storage devices would simplify the process very much. However, unless special hardware is available, the development cycle will then consist of:— modify software — program EPROM — test on target system — modify software etc. etc. The fact that this process involves programming EPROMs slows it down very much.

This article describes the construction of a piece of hardware which allows software to be developed on a separate computer without having to program EPROMs until the program is perfected. An EPROM emulator is basically a dual port memory card, ie, a RAM board which may be accessed from either computer. The method of operation is to produce software on the development system and download the object code to the emulator, after which the control computer may access the card as if it were its own memory. To ease interfacing to the target system the emulator is fitted with a length of ribbon cable and a DIL header which may be plugged directly into an EPROM socket.

System Philosophy

The most convenient way to add an EPROM emulator to a home computer is to interface it directly onto the bus so that it may be accessed as memory. However, this is also the most system dependent way of adding the hardware, which means that it will only be usable with one type of computer or, at best, only with computers using one family of processors.

This emulator has been designed so that it may be interfaced in quite a number of different ways and the user may pick the method which is most suited to his particular computer.

a) The board has been artworked to the Microtan standard so that users of this computer may plug it directly into the mother board and access it directly as system memory.

b) Since the TANBUS signals are fairly standard among 6502, 6800 and 6809 systems, owners of other computers using these processors may interface the emulator card onto the bus so long as they sort out the physical aspects of this (ie, making sure the edge connectors match).

c) For those users with computers utilising different processors (including the large number of Z80 systems) or those with a memory map, which is already full, the emulator may be interfaced via a parallel port. Although this is a very versatile method there are certain disadvantages: a small amount of downloading/uploading software is required on the computer, and 23 bits of parallel I/O are needed for the interface.

d) The most versatile method of all is also the most complicated, and for this reason will be dealt with in a separate article. This is to add some local intelligence in the form of a simple processor board with an RS232 interface to the emulator. The system would then be able to communicate with any computer having a standard RS232 serial interface using standard system routines on that computer. In fact, the universal EPROM programmer card described in the August and September '83 and January '84 issues of ETI may also be added to the system, giving a three card intelligent EPROM programmer/emulator which may be interfaced to virtually any computer and which would provide very comprehensive firmware development facilities.

The card described in this article is even more versatile than the foregoing paragraph would suggest. So far we have only considered the card as an EPROM emulator. The board is, of course, essentially an 8k RAM card and thus may be used as a memory extension without any reference

to emulation. In these days of 16k and 48k computers an extra 8k may not seem a very big leap forward, but the basic Microtan has only 1k of RAM and even if the TANEX card is being used this board would double the amount of available RAM. In addition the emulator contains low power CMOS memory and battery back-up facilities, enabling data to be retained on power down. Even in a system with a full memory map it may be used as external non-volatile memory, accessing it via a PIA.

Design Process

This article not only describes the construction of a piece of equipment which readers may assemble for their home computers, it also describes the various options which have been used in the design and how the circuit as it now appears was arrived at.

Choice Of Ram: We have already noted that an EPROM emulator is essentially a random access memory (RAM) card. The most fundamental design consideration therefore is what type of RAM ICs to use. Since it was felt that the memory on the card whould be non-volatile, being backed up by a battery when the computer is switched off, the choice is limited to CMOS static RAMS, all other devices having too high a current consumption to be powered from a battery for very long.

The next question is the total size of the memory to be used and how this total should be made up in terms of individual chips. The choice is essentially between 2K byte devices and 8K byte devices. The 8K devices have only recently been introduced and as a result are still very expensive. For this reason alone, rather then any technical consideration, these memories were rejected in favour of 2K RAMs. So how many devices should be used on the emulator? Four can easily fit onto an 8" x 4½" card without making it double sided. At least 8 chips could be put on a double sided card, but it was thought that on grounds of economy the board should be made single sided if at all possible. Four ICs will in fact give 8K bytes of memory in total which means that all EPROMs up to and including the 2764 (or 3564) may be emulated. At the moment most amateur computer equipment uses 2716s, 2732s or occasionally 2764s, so this seems quite an adequate solution. If, at a later date, a need arises to emulate larger EPROMs, two such boards could be used together with a small amount of additional logic.

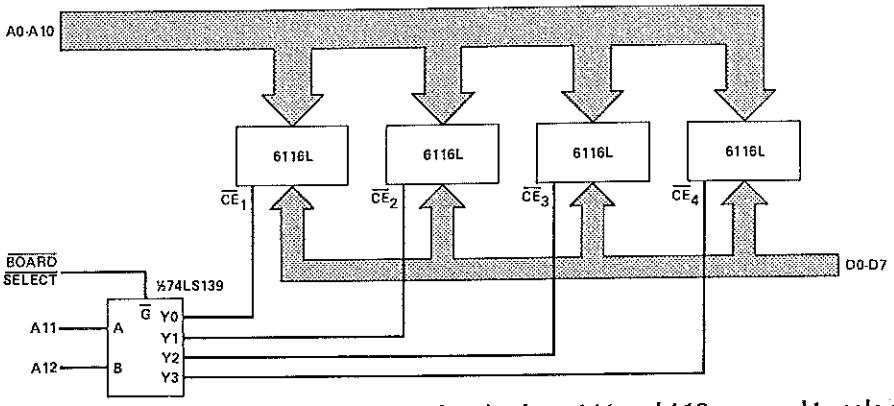


Fig. 1 The RAM interface arrangements, showing how A11 and A12 are used to select between memory chips by means of a 2-to-4 line decoder.

The final question, then, is exactly which 2K byte CMOS static RAMs to use. Not all CMOS static RAMs have a low enough standby current for battery backed-up applications, and the choice eventually narrowed to the 6116L or the 5516. The 5516 has a number of advantages over the 6116L, having a standby current of 1.0μA (depending on temperature) compared to 4.0μA typically for the 6116L, and two CE inputs, one specifically intended for a power-down signal, hence simplifying battery backed-up operation. Further, standby mode is only guaranteed on the 6116L if all the inputs are held to within 0.2V of 0V or VCC (except for CE which must be high) which means that 21 pull-up or pull-down resistors are required.

In spite of these considerations, the 6116L was eventually chosen on the grounds of its lower cost. Assuming a 100 mAh capacity battery is used, a data retention time of quite a few thousand hours will still be achieved. The circuit will be slightly more complicated but it was felt that a cost reduction in the region of £10 was of prime importance. It was also realised that, since 6116Ls have a standard JEDEC pin out, designing the circuit around these devices would mean that users not wanting battery back-up can choose a number of other, less expensive, 2K x 8 RAMs.

Intefacing The Ram: In order to address 8K bytes of memory, 13 address bits (A0-A12) are required (8196 (8K) = 2^{13}). Of these 13 bits it is evident that 11 will connect directly to the 6116L devices whilst the remaining two will be required to select which of the four RAM chips to address. This implies an arrangement like that shown in Fig. 1, where the ½ 74LS139 is a 2 to 4-line decoder, a device which inputs a 2 bit binary value and gives a logic zero at one of the outputs depending on the input value. The eight data bits obviously connect directly to all the RAM chips.

If this were a standard 8K memory card for an 8 bit microprocessor system, the board select signal would take a logic low value for one combination of the remaining address bits (A13-A15) hence locating the board in one of the 8, 8K blocks available within the 64K addressing space. This would ideally require a 3 to 8 line decoder, but since there is already a 74LS139 in the circuit only half of which is used, it seems more appropriate to use the other half and then use A15 or an inverted A15 to select it.

Since this is not a simple 8K RAM card but an EPROM emulator, there is an alternative board select condition, namely, when the EPROM socket is being read. To generate the final board select signal therefore, the two active low signals are ORed. This is shown in Fig. 2. It should be noted that, in the final version, the link arrangement associated with A15 has been slightly changed to avoid the possibility of 2 TTL loads being

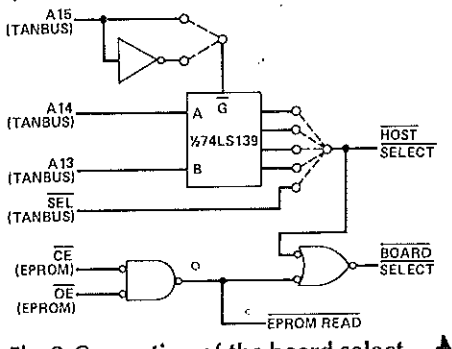


Fig. 2 Generation of the board select and host select signals.

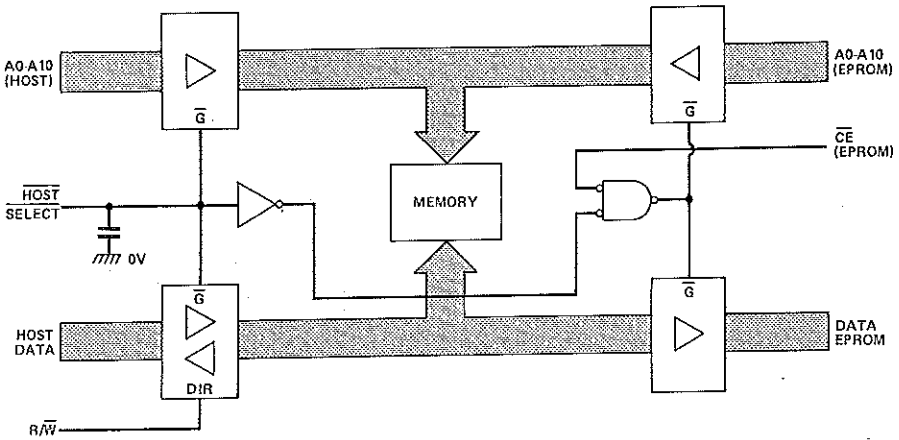


Fig. 3 The use of buffers to prevent the host and target systems being simultaneously connected to the RAM chips.

applied to this signal. Note also that an external board enable signal from the edge connector is link selectable. This was added to reduce by 2 the number of PIA lines required when interfacing the card in his way.

So far we have discussed how the 8 data bits and 11 of the address bits connect to the 6116Ls without considering whether these are the EPROM address and data busses or the corresponding host computer signals. Both sets of signals must be connected to the RAMs, but if this is done directly it would short the bus of the host computer to the bus of the target system, a condition which would prevent either system functioning.

The answer is to use buffers, only one of which may be enabled at any one time. It was decided that the buffers to the target system should normally be enabled but whenever the development computer required access to the memory it would take priority, enabling its buffers and disabling those to the target system. Address buses are always unidirectional, and as such a 74LS244 buffer may be used. Data buses can be either uni-directional or bi-directional depending on whether there is write access to the memory. The data bus to the host must be bi-directional so a 74LS245 is used, whereas the data bus to the EPROM socket only requires read access and a 74LS244 is sufficient.

During initial testing it was discovered that the target system port occasionally suffered from read errors. This was caused by false $\overline{CE}$ signals generated in the host during the first half of the processor cycle in which addresses are not valid. Since the duration of such signals is very short, the addition of a capacitor effectively overcomes this problem. Figure 3 illustrates this aspect of the circuit.

One final point on the interfacing of the RAMs to the address and data buses. You might expect A0 on the host and target systems to be connected to A0 on the RAMs, A1 to A1, etc. This is not the case in this circuit. From an electronic point of view there is no reason why it should not have been interfaced this way, and if it were not for the fact that the author also produced the PCB artwork this is the way the circuit would have been designed.

It was designed in the manner presented so as to simplify the artwork and keep the number of wire links down to a minimum. This might seem a strange decision to make but as far as the outside world is concerned, the address pin labelling on the RAM chips is quite arbitrary. It makes no difference what order they are connected in — each address bit combination still addresses a unique location within the IC. A similar argument may be applied to the data pins on the ICs. It should be noted that this method should not be used when interfacing EPROMs as these will have been programmed assuming the correct signal order and hence compatibility must be maintained.

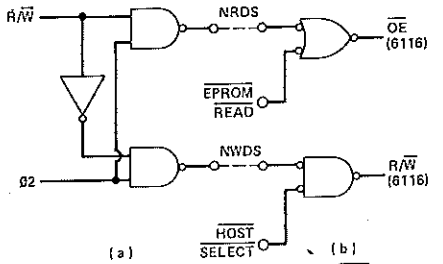


Fig. 4 a & b Generation of the $\overline{OE}$ and R/$\overline{W}$ signals.

RAM Control Signals: We have already dealt with A0-A10, D0-D7 and $\overline{CE}$ on the 6116Ls; this leaves the $\overline{OE}$ and R/$\overline{W}$ still to be connected. These signals are equivalent to the Intel NRDS and NWDS respectively and Fig. 4a shows the standard method of generating them from 6502 signals. Since the EPROM port of this card has no write access to the RAM, the generated NWDS does not require ORing with a similar signal associated with the EPROM port. However, when the RAM is enabled by a read from the EPROM port but a R/W is generated by a write to some other memory on the host system, the signal must be gated with HOST SELECT in order to prevent a false write. The $\overline{OE}$, on the other hand, does require ORing with a corresponding signal on the target system.

The additional two gates required are shown in Fig. 4b which extends Fig. 4a. It should be noted that $\overline{OE}$ and R/$\overline{W}$ can both be active when a write to the card is being carried out from the host system and the target system is attempting a read. This is not a problem since the data sheet for the 6116L makes it clear that, under these circumstances, the write takes priority over the read. This is perfectly acceptable since the host is to have priority over the target port.

Supply And Power-Down Circuitry: To ensure that there is data retention when the main computer supply is switched off a battery supply is required. Since the 6116L only requires 2.0V in its standby mode,

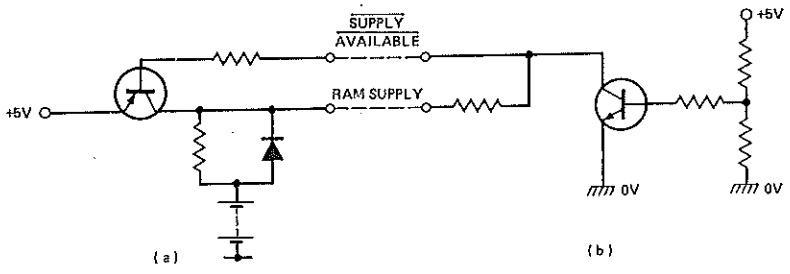
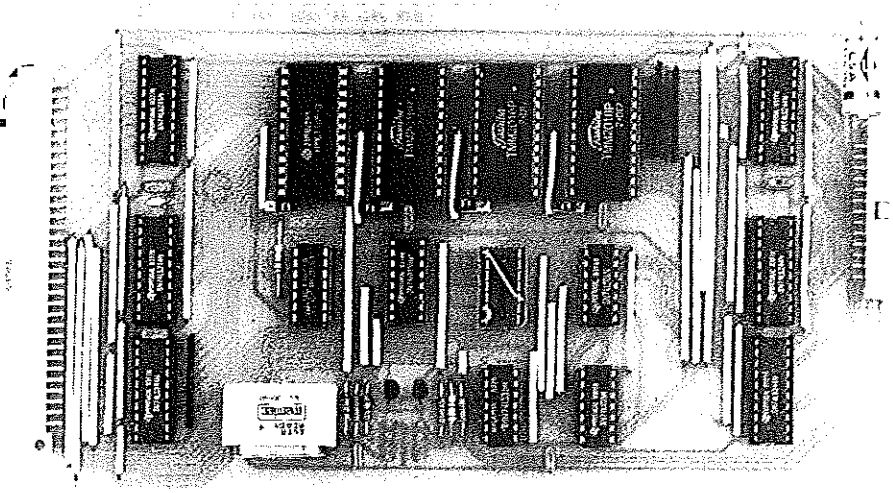


Fig. 5 a & b The battery trickle charger and the power-down circuitry.



the readily available PCB mounting 3.6V 100mAh battery is a perfect choice. Figure 5a shows a circuit in which the battery is trickle charged via the current limiting resistor when the main supply is present, supplies current to the RAMs via the diode when the main supply is not present, and is prevented from discharging through the power supply under these conditions by the transistor which will be turned off. The resistor value is selected to give a charging current of 1.0mA (the current stated for this battery in the data sheet). From Ohm's Law, this will be V/I where I is this charging current and V is the potential difference between the battery and the main supply (5V-3.6V) or in other words 1.5k ohms. Although there will be a potential drop of typically 0.7V across the diode, there will still be 3.6V-0.7V=2.9V available to the RAMs, which is within the specification for these devices.

It now remains to decide how to generate the power available signal. For the purpuses of supply isolation the requirements are not too stringent — all that is required is for it to go sufficiently high to turn on the transistor when the supply voltage is higher than the battery voltage. There is another use for this signal however, to write protect the memory on power down. Since the major part of any computer system is made up of TTL devices and these are only guaranteed to function correctly at supply voltages of 4.5V and above, it is quite feasible that random signals on the bus will cause un-intentional writes to takes place on power-down, hence corrupting the data in the RAM.

Considerable time was spent to find some way of accurately detecting a voltage level of about 4.75V to generate the supply available signal, but any such method would involve the constructor in some quite precise setting up which would obviously be undesirable. The method eventually used does not require any setting up, and although it does not succeed in accurately detecting 4.75V experiment shows that it works. The level detector is simply a potential divider and transistor so arranged that, when the supply voltage is greater than about 4.2V, a potential of greater than 0.7V is present at the transistor base which turns it on and hence gives a logic low signal. This arrangement is shown in Fig. 5b. If an attempt is made to detect something much closer to 4.75V, the resistor tolerances might cause the transistor not to turn fully on at 5.0V.

The need for write protection of the RAMs has already been mentioned. This is done by gating the four chip enable ($\overline{CE}$) signals in Fig. 1 with the supply available signal in such a way that they can't go low when the power isn't present. Obviously, the gates used need to be active even when the main supply is not present, so they must consume little power and work on a low supply voltage. This demands a CMOS device. Figure 6 shows this gating arrangement which is used to modify the circuit given in Fig. 2.

If the circuit portions illustrated in Figs 1-6 are connected together the result will be the complete circuit diagram shown in Fig 7. There will be a few changes from the circuits already given due to the following:—

1) A few extra gates have been added as buffers to ensure that no more than 1 TTL load is presented to any bussed signal.
2) To minimise the number of IC packages required, two gates have sometimes been used to replace a single gate of a different type. For example, an AND gate followed by an inverter has been used as a NAND gate in two places.
3) Gates have sometimes been drawn in negative logic notation to clarify their function. In the final circuit diagram, however, these have been translated to their mo conventional forms.
4) The 6116L RAMs are only in their low power standby mode when all their inputs are within 0.2V of either 0V or VCC. The resistors in the SIL packages, ie RP1, RP2 and RP3, have been added to ensure this.
5) In accordance with normal dig tal practice, a number of decoupl ing and reservoir capacitors have been connected across the supplies.
6) Since a number of less expensive but higher power RAMs are pin compatible with the 6116L devices, and since not all users would require all four RAMs to be non-volatile, links have been added to allow the user to select either the main supply or the battery supply to each of the RAMs.

Next month: Construction and us

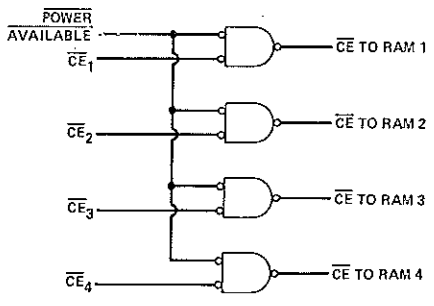


Fig. 6 A modification to the circuit shown in Fig. 2 which provides write protection for the RAMs on power-down.

HOW IT WORKS

Since a lengthy description of the design process has already been given, this section really serves only to give an overall picture of the circuit, outlining which components are associated with each particular task.

The memory is made up of four 2K x 8 RAMs, these being IC8, IC9, IC10 and IC11. The RAMs are connected to internal data and address busses which are isolated from the host and target system busses by various tri-state buffers. IC5 buffers the host data bus, IC12 ad IC14 the host address bvus, IC6 the target data bus and IC13 and IC15 the target address bus. The circuitry comprising IC1c and IC2d controls the buffer enabling and ensures that both sets cannot be enabled at the same time and that the host takes priority. IC3 and most of the remainder of IC1 and IC2 are associated with generating the RAM CE, OE and R/W signals by a combination of control signals from both ports. The RAM CE is split into four separate signals for the four RAMs by IC5b, and IC7 ensures that these signals can't be active under power-down conditions. This circuitry requires a signal indicating that the appropriate portion of the host memory map has been addressed, and this signal is generated by IC5a, IC1d, LK1 and LK2. The remainder of the circuit is associated with the battery supply and power-down circuitry.

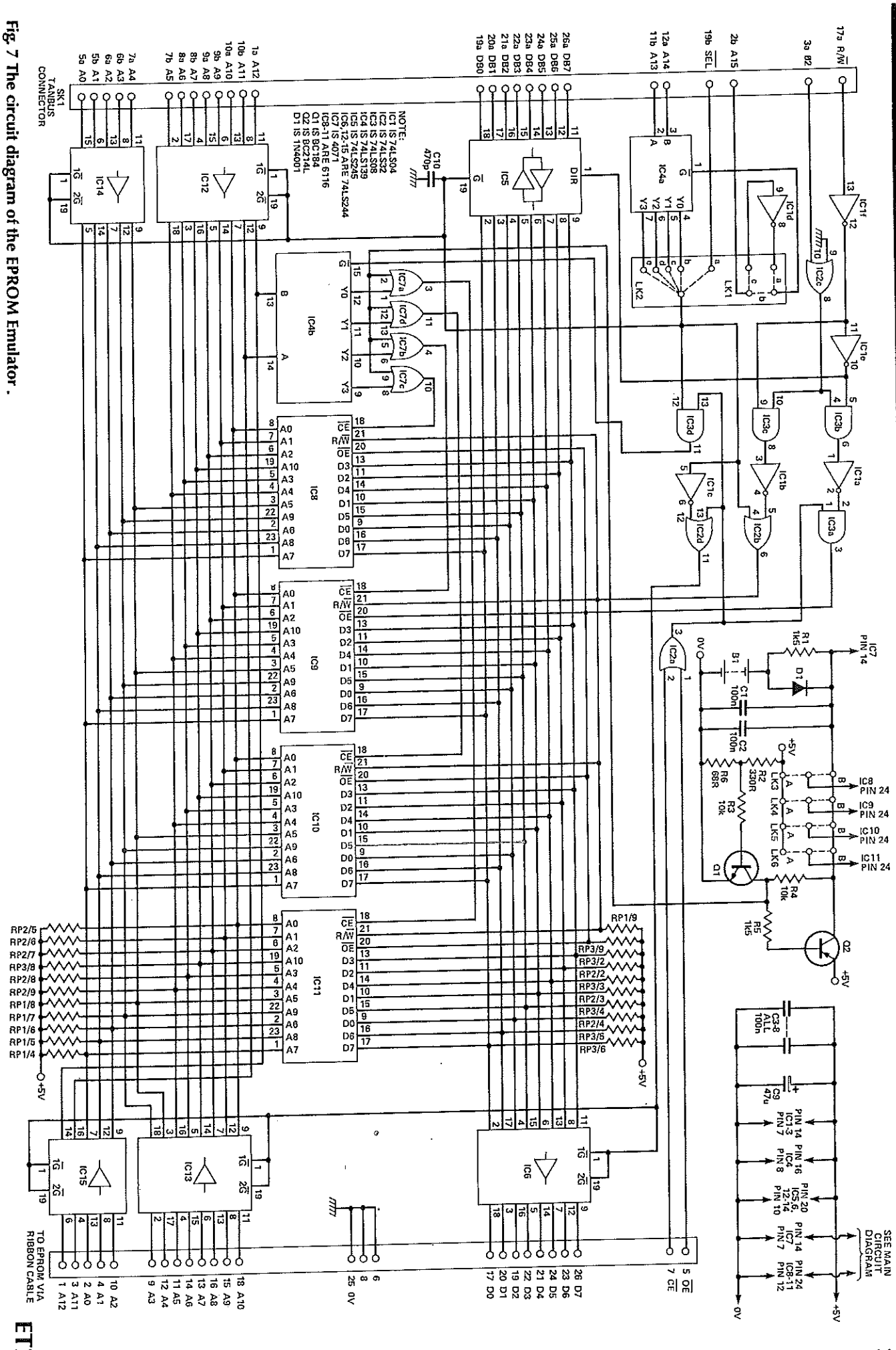
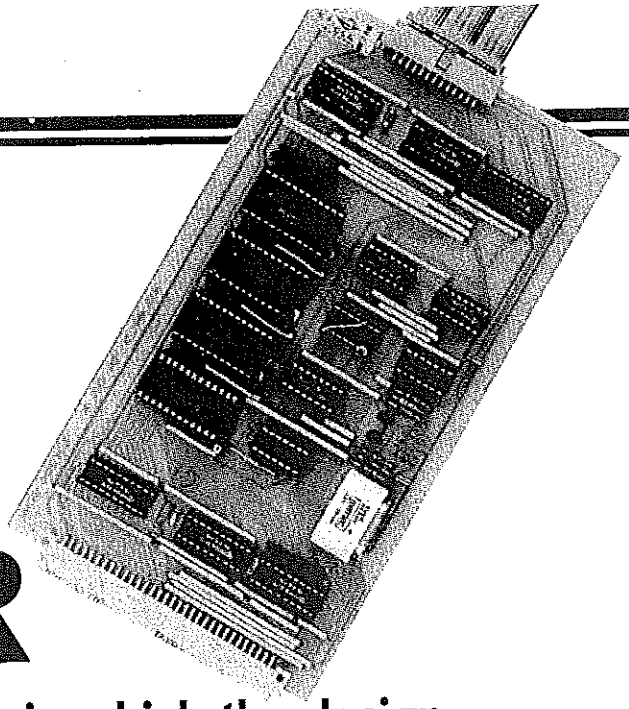


Fig. 7 The circuit diagram of the EPROM Emulator.

VERSATILE EPROM EMULATOR



Following on from last month's article in which the design process was examined in some detail, Mike Bedford describes the construction of the board and offers some advice on interfacing it to your microcomputer.

Construction is quite straightforward and, with the exception of link selection, the board requires no setting up. The circuit has been artworked as a single sided board to keep its cost to a minimum. The inevitable result of this, in a circuit of even moderate complexity, is that there are a number of wire links on the board. It is suggested that these are fitted first.

Sockets for the ICs are not absolutely essential, but since their omission would not greatly reduce the component cost it is suggested that they are used, especially for the 6116L devices. The proper precautions (ie, not touching the pins) should be taken when handling the CMOS devices. These are the 6116Ls and the 4071.

No assumption should be made regarding the state of charge of the PCB battery as supplied. It should not be placed pins-down on a conducting surface, but neither should the board be expected to function correctly in battery back-up mode until it has been powered up for a number of hours, hence allowing the battery to charge.

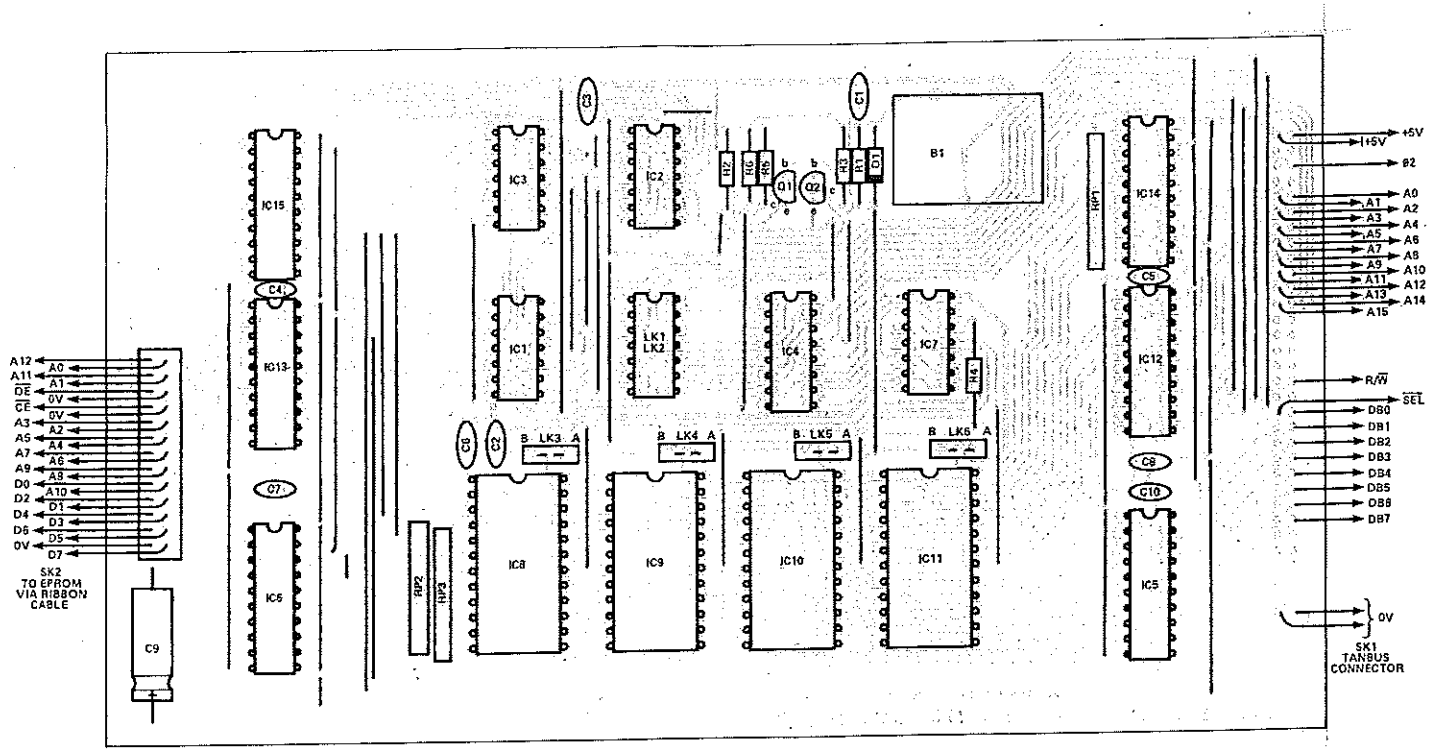


Fig. 8 Overlay diagram of the PCB.

For users not wishing to fully populate the board, IC11 should be fitted to give a 2K board whereas IC11 and IC10 should be fitted to give a 4K configuration. When emulating EPROMs smaller than 2764s, unless tied low by use of pull down resistors, the unused address lines (A11 and A12 on the 2716 and A12 on the 2732) will float high, causing the emulated RAM to occur at the top of the 8K space on the host system. So long as this is realised it presents no problems unless, of course, the board is only partially populated, in which case non existent memory will be accessed. To avoid this problem any such unused address lines should be connected to 0V.

Links LK3 — LK6 should be fitted to select whether or not battery backed up operation is required on each of the RAMs. LK3 affects IC9, LK4 — IC10, LK5 — IC11 and LK6 — IC12. In each case, if the board is held component side upmost with the TANBUS edge connector at the right of the board, connecting the centre to the left pin of the link will select battery backed-up operation whereas connecting the centre to the right pin will select the system 5V supply. If the non-battery backed up option is selected for a particular position, a less expensive 6116 or 2016 device may be substituted for the 6116L.

Links LK1 and LK2 are fitted onto a 14 pin DIL header which then plugs into a DIL socket on the board. Figure 9 may be used to select these links to give the required positioning of the board within the host address space.

After building up the board, the final aspect of construction is the making up of a cable to connect the emulator to the EPROM socket on the target system. This will consist of a 26-way female IDC connector, a length of ribbon cable and either a 24 pin or 28 pin DIL header. Separate leads will be required for 2716/2732 and for 2764 devices unless there is room on the target board to enable only the lower 24 pins of a 28 pin DIL header to be plugged into the EPROM socket, in which case the same lead may be used for all these EPROMs. Table 1 shows the details required to make up the cables.

A length of ribbon cable can cause read errors from the target system due to noise pick-up resulting from the capacitance between adjacent conductors. This is a particular problem since the signals present on EPROM sockets are not usually intended to drive lengths of ribbon cable. The problem is reduced by keeping the cable length to a minimum, and 12" should be considered an absolute maximum. Initial experiments also showed that a good

IDC connector pin no	EPROM signal	2716 2732 pin no	2764 pin no
1	A12 (Only 2764)	—	2
2	A0	8	10
3	A11 (Not 2716)	21	23
4	A1	7	9
5	OE	20	22
6	0V	—	—
7	CE	18	20
8	0V	—	—
9	A3	5	7
10	A2	6	8
11	A5	3	5
12	A4	4	6
13	A7	1	3
14	A6	2	4
15	A9	22	24
16	A8	23	25
17	D0	11	
18	A10	19	21
19	D2	11	13
20	D1	10	12
21	D4	14	16
22	D3	13	15
23	D6	16	18
24	D5	15	17
25	0V	12	14
26	D7	17	19

Table 1 Details of the connections required between the emulator and the EPROM socket.

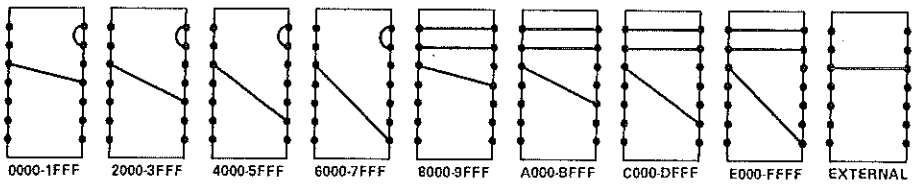


Fig. 9 Link arrangements to give various positions of the board within the host system address space.

PARTS LIST

RESISTORS (All ¼W 5%)
R1, R5 — 1k5
R2 — 330R
R3,R4 — 10k
R6 — 68R
RP1-RP3 — 47k, SIL, 8 commoned

CAPACITORS
C1-8 — 10n Ceramic
C9 — 47u 16V axial electrolytic
C10 — 470p ceramic

SEMICONDUCTORS
IC1 — 74LS04
IC2 — 74LS32
IC3 — 74LS08
IC4 — 74LS139
IC5 — 74LS245
IC6,12,13,14,15 — 74LS244
IC7 — 4071
IC8,9,10,11 — 6116LP-3 (OR 6116P, 6116P, 2016 — see text)
Q1 — BC184
Q2 — BC214L
D1 — 1N4001 or similar

MISCELLANEOUS
SK1 — 2x32 way, A+B DIN Euro Connector, male angled pins
SK2 — 26 way low profile male PCB mounting connector
LK1 & LK2 — 14 pin DIL header in 14 pin DIL socket
LK3,4,5,6 — 0.1" pitch, 3 way Molex connectors with 2 way link
B1 — 3.6V 100mAh, PCB mounting nicad battery

PCB; sockets for ICs; female IDC connector to fit SK2; ribbon cable; 24 and/or 28 pin DIL headers.

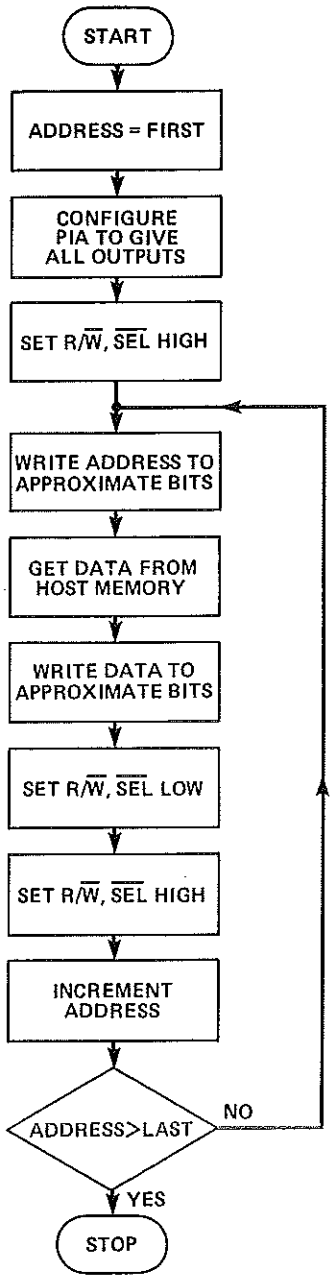


Fig. 10 Flow diagram showing how data should be written to the emulator from the host system via a parallel port.

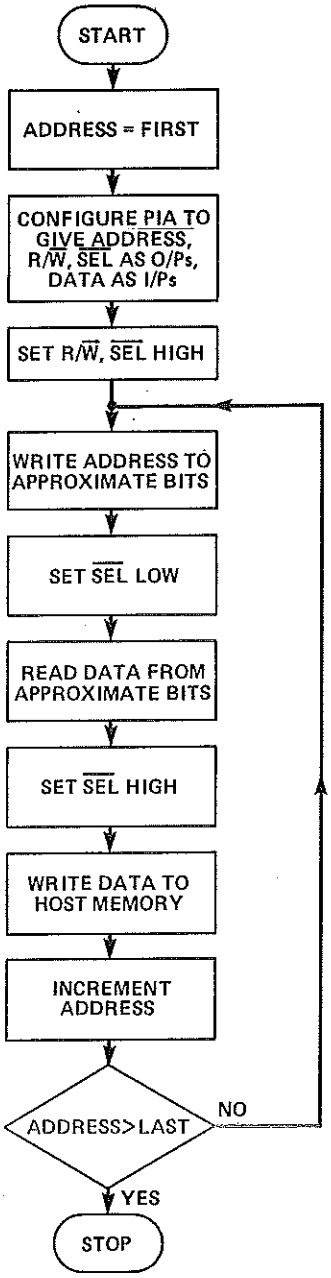


Fig. 11 Flow diagram showing how data should be read from the emulator by the host system via a parallel port.

earth connection is essential between the emulator and the target system; the single ribbon cable conductor could well be inadequate and a separate, thicker wire might be required. It was also found that the emulator can be sensitive to the path taken by the cable. In particular, care should be taken to ensure that it is not stretched tightly across the target board or interference may result.

Using The Emulator

Before making use of the emulator, the user must first check that it is compatible with the target system. This is determined by viewing the access times required by the target system in view of the fact that, if 150nS RAMs are used on the emulator card, the OE or CE to data valid time (whichever is later) will be about 210 ns. This means that the emulator will generally work with target systems having processor frequencies up to about 2.0 MHz. In fact, the prototype has been used consistently at 1.7MHz and with faster RAMs at 2.2MHz.

Another thing to remember about the emulator is that only one of the ports may have access to the memory at any one time. This card has been designed so that the port to the target system is the one which is normally enabled, but whenever the development system requires access it takes priority, hence denying access to the target system. The host computer will thus always be able to read or write to the RAM. The fact that the target port will sometimes be denied access to the emulator is not a big problem because whenever the development computer writes to the emulator it will generally be to change the program and, accordingly, it will usually be required to do a target system reset.

The method of driving the emulator depends very much on which method of interfacing is used. If the board is interfaced directly onto the system bus of a 6502 or 6809 based computer, there is really nothing to be said. It is simply a matter of writing the data to the emulator in just the same way as to any other memory on the computer. On the other hand, the hardware for implementing a stand alone emulator with an RS232 interface has not yet been described, so the method of using the emulator in this configuration will be left to a future article. The only interfacing method which therefore requires any amount of instructions is via a parallel port. Since a variety of different machines employing various PIAs, PIOs & VIAs could be used it seems pointless to give a BASIC or assembler program for one particular hardware. One flow diagram is for transferring data from the computer to the emulator and another for carrying out the reverse process. In either case the following signals require connecting to PIA pins: A0-A12, D0-D7, F/W and SEL. In addition, $\varphi 2$ should be connected to +5V and link 2 should be selected to the 'd' position.

BUYLINES

All of the semiconductors (including the various memory options), the passive components, the SIL resistors, etc, are available from our regular advertisers. The only item likely to cause any problems is the PCB mounting nicad; we obtained ours from a trade source, but if you are unable to find an identical item there is a Maplin equivalent which has slightly different pin spacing. The PCB is available from our PCB service, see page 54.