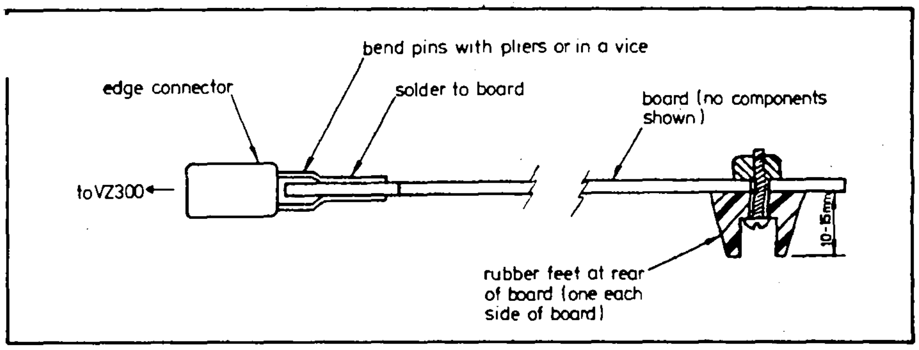
ETI-1611 EPROM programmer

This month part 2 continues with construction, testing and software for the programmer.

Herman Nacinovich

COULD YOU USE a low cost EPROM programmer that will program EPROMs in the popular 2764 to 27256 series? How about one that will also copy from one EPROM to another in seconds? And one which is fully software programmable to cater for EPROMs from different manufacturers, with different programming voltages? How about an EPROM programmer which can double as a memory expansion for a VZ3000 computer when it is not being used for programming EPROMs? Or, one which can load your favourite BASIC program directly from an EPROM into your VZ300 computer in a matter of seconds? This is it!



bend pins with pliers or in a vice
edge connector
solder to board
board (no components shown)
to VZ300
rubber feet at rear of board (one each side of board)
10-5mm

The bond plugs directly into the VZ300 expansion stat. The foot at right releives mechanical stress on the connector at left.

Construction

Construction of this project is simplified by the fact that everything goes on a single board and there is no messing about with wires, switches and a box to put everything into. However the usual, if not more, care, should be exercised to ensure that everything is put in the right way around, particularly the IC's, diodes, transistors and electrolytic capacitors. The board uses double sided construction and boards as supplied by kit suppliers will (hopefully) have plated through holes. Although the number of parts on the board is not too great, there are lots of tracks on the board and many of these are very closely spaced. It is strongly recommended that the greatest care be taken in the first instance when examining the board to ensure that there are no bridges or breaks in the board pattern. Never assume that any board, whether you make it yourself or get it in a kit, is free of faults.

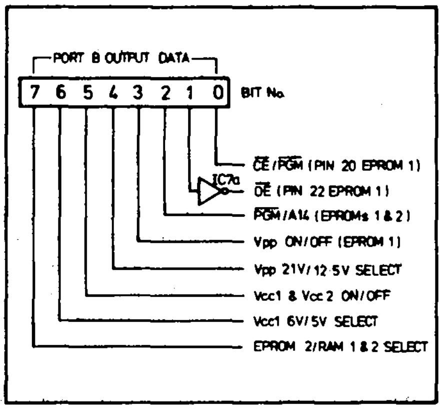
It is also important to be careful, when winding the ferrite core transformer, to ensure that the ends of the windings are connected the right way around. If not, the inverter won't work and you might find transistor Q1 getting very hot. The particular ferrite core and former recommended here are made by Neosid, the core and former being very kindly supplied for the prototype by Neosid Limited Australia in Lilyfield, NSW. It is a very easy transformer to wind and the former is of moulded construction with integral pin terminals, virtually guaranteeing success providing that reasonable care is taken in putting the transformer together. One point to watch, however, is that the pin terminals are fairly small and close together. This is no real problem but a steady hand and a pair of long-nosed pliers with very thin, pointed ends do help when trying to twist the wire ends around the pins. When that is done, solder the wire ends to the pins and then fit the transformer onto the board.

A little hint: It is very difficult to scrape the enamel off enamelled winding wire as required when making solder connections. I have found that by burning the wire end with a lighted match and then rubbing lightly with steel-wool, the enamel comes off very easily.

The board is designed so that it can be plugged into a 44-way edge connector, if desired, with a 44-wire ribbon cable going to a second edge connector which plugs into the memory expansion slot of a VZ300. Alternatively, an edge connector of the type with rearwardly extending solder pins can be soldered directly to the tracks on the edge of the board. This avoids the need for a ribbon cable and was the method chosen for the prototype.

One drawback, however, is that you may not be able to plug in a printer or disc drive at the same time because the memory expansion and I/O slots in the back of a VBZ300 are a little too close together. If you anticipate that you will need to plug in a printer or disc drive at the same time as the EPROM programmer then I would recommend that you use a pair of 44-way edge connectors joined by a ribbon cable.

Apart from the ZIF sockets for EPROMs 1 & 2, IC sockets were not used in the prototype. One reason for this was that the prototype board, though of double sided construction, did not have plated through holes and that would have made soldering IC sockets to the board a little tricky (though not impossible). In the case of the ZIF sockets, I could not solder the pins to both sides of the board as required by the double sided board construction. However, I solved the problem by drilling the IC pin holes on the board a little oversize and linking both sides of each solder



Port B connections

pad, where required, with very fine wire. I would expect boards supplied by kit suppliers to have plated through holes so that, if you choose to use IC sockets, you will find soldering them no more difficult than you would if the board were single sided. One note of advice, however, whether you use IC sockets or not: once you solder an IC or an IC socket, the tracks running underneath the IC or IC socket, on the component side of the board, will no longer be accessible. So, double check for breaks in, or shorts between, tracks underneath each IC before soldering.

On the board there are a number of through-the-board links. Assuming plated-through board construction, these links will be automatically formed in the board as supplied so that there will be need to solder anything to them.

You will notice that there are a couple of trimpots on the board. It may happen that the trimpots supplied with a kit may not quite fit the holes on the board since trimpots come in different sizes with different pin spacings. If you find this to be the case then simply bend the leads (very carefulloy) so that they will fit. Be particularly careful if the trimpots supplied have a ceramic rather than plastic base because the ceramic base is extremely brittle and therefore easily broken.

One last hint: I had a little problem with the locking lever on the ZIF socket being awkward to get at. I solved the problem by bending the lever upwards about a quarter way from its end with a pair of pliers.

Addendum

When you have completed soldering in the components on the board as per the payout given last month, you will have to solder additional resistors R19 and R20 to the reverse side of the board. Solder resistor R19 to pins 24 and 29 of IC4 (Z80 APIO) and resistor R20 directly to pins 1 and 7 of IC7 (74LS05). note: These resistors are not shown in the parts layout published last month.

Testing

Before plugging the board into your VZ300, make one final check over the entire board with a magnifying glass to ensure that there are no breaks in any of the tracks and that there are no solder bridges between tracks. It would also do no harm to check that all the components, particularly IC's diodes, transistors and electrolytics have been soldered in the right way around.

When satisfied that all is OK, plug the board into the VZ300. Do not plug in any EPROM yet. Switch on your monitor and allow it to warm up. Then switch on the VZ300 and observe the display on the screen. If it is the normal display that you get after switch-on then all, so far, is OK. If, however, you get garbage on the screen, or nothing at all, then switch off immediately. In this case there will almost certainly be a fault on the board, either a faulty component or a short circuit between tracks, or a broken track or you may have forgotten to solder one of the components.

Assuming that the display is OK, connect the negative lead of a multimeter (set to read 25 V or more) to a convenient point on the board at 0 V and the positive lead to either end of inductor L1. You should get a reading of around 21 V plus or minus 3 V. If you don't then switch off immediately. Check whether transistor Q1 is hot or cold. If it is hot then most likely one of the windings of transformer T1 has its ends wrongly connected or transposed. If Q1 is cold then possibly the transformer windings are connected to the wrong pins or there is an open circuit somewhere, depriving Q1 of base or collector current. In any case, check the circuit around Q1 and T1 before switching on power again.

Assuming, again, that so far everything is OK the LED should be alright. now, you can check a few voltages on the pins of the EPROM sockets. If these pins are not readily accessible by your multimeter probes, you could plug in a conventional 28-pin IC socket into each ZIF socket. Most ordinary IC sockets have the pin connectors exposed, making access with a probe easy.

At this stage, with the VZ300 switched on, you should be able to measure 5V (plus or minus 0.25V) at pins 1,27 and 28 of both EPROM 1 and EPROM 2 sockets.

Now key in the following: OUT203,7:OUT203,15:OUT201,7:OUT201,143. The LED should light up and you should get the following:

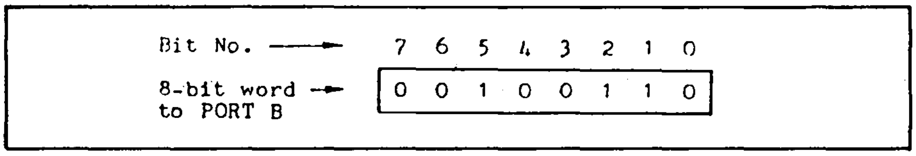
EPROM 1 socket: pin 1: 21 V plus or minus 3V, pin 27: 5 V, pin 20:4 V, pin 22: 0V, pin 28: 6V plus or minus 0.25 V. Adjust RV2 for 21 V at pin 1. Now key in: OUT202,40. You should now get the following voltages at the pins of EPROM 1: pin 1: 12.5 V plus or minus 2 V, pins 20,27: 0V, pin 22: 5 V, pin 28, 5 V. Adjust RV1 for 12.5 V at pin 1.

Now key in: OUT 202,2. The LED should now go out and you should get a reading of 0V at each of pins 1, 20,22,27 and 28.

This completes the preliminary testing of the EPROM programmer and also demonstrates how the various operating voltages and control functions for the EPROMs are software controlled.

The ultimate testing is carried out by loading a suitable program into the VZ300 and running that program with an EPROM plugged in. To save you, dear reader, the rather time consuming task of writing a program, such a program has been developed for you. Unfortunately, this has turned out to be a somewhat lengthy one and space limitations prevent publication here. However, the program is available on EPROM from the author at the address given at the end of this article. Loading the program into the VZ300 involves merely plugging the EPROM into the EPROM 2 socket on the programmer board and keying in a few short instructions in BASIC. The actual loading takes only a few seconds, compared with loading the same program from tape, which would take many minutes. This program will enable you to manually enter machine code into memory, make corrections, if necessary, and then copy into EPROM. The program will also let you check if an EPROM is fully erased and you can copy from one EPROM to another. EPROMs from 2764 to 27256 are catered for.

I would like to express my particular appreciation of Neosid Australia for their assistance. I had a lot of difficulty trying to find a suitable ferrite core from other sources for this project. Neosid came to



How the bits are arranged on Part B

the rescue with just the right core for the job.

SOFTWARE DESCRIPTION

This programmer board is configured to appear partly as an external memory and partly as an I/O (INPUT/OUTPUT) device to the Z80 microprocessor of a VZ300 computer. There are, therefore, four primary instructions in BASIC which are needed to communicat with and control the programmer. These are:

POKE (write into memory)
OUT (write into I/O)
PEEK (read from memory)
INP (read from I/O)

The VZ300 computer has 16K of internal ROM (Read Only Memory), occupying addresses 0000H (Hex) to 3FFFH and 16K of user accessible RAM (Random Access Memory) occuypying addresses from 7800H to B7FFH. The space occupying addresses B800H to FFFFH (a little over 16K) is vacant in the VZ300 and is available for memory expansion. In addition, the Z80 microprocessor in the VZ300 is capable of addressing up to 256 bytes in I/O space, independently of memory, by an I/O instruction.

The part of the programmer board which appears as external memory comprises up to 4K of optional RAM (RAMs 1 & 2) and up to 32K of optional ROM (EPROM 2). These share the same address space but only one of these can be selected at any one time. Whichever is selected depends on the status of one of the bits of an 8-bit control word contained in an I/O instruction.

In addition, address latches IC1 and IC2 are enabled by any memory read or write instruction to the same address space as occupied by the on-board RAM and ROM. Being 'write only', these latches do not pose any danger of bus conflict but it will be noted that any address stored in the outputs of these latches will be changed to a new address whenever a read or write instruction is sent to either of the external RAMs or ROM. The latched outputs are unaffected by any instruction to an address below B800H. When an address is latched into IC's 1 & 2, this becomes the address for EPROM 1 (the EPROM which is to be programmed).

IC4 provides the I/O interface between the I/O part of the EPROM programmer and the VZ300's internal Z80 microprocessor. As already noted, the Z80 is capable of addressing up to 256 bytes in I/O space (from 00H to FFH). A simple address decoder (IC5a) enables IC4 for I/O instructions to addresses C0H to FFH (192 decimal to 255 decimal). This leaves addresses 00H to VFH availble for other devices (disc, printer, etc) which may be plugged into the I/O expansion slot, next to the memory expansion slot, of a VZ300. IC4 has two 8-bit I/O ports (PORT A and PORT B) which are programmable as either input ports, output ports or as (PORT A only) a bi-directional data transfer between the Z80 microprocessor of a VZ300 and the data pins of EPROM 1. PORT B is used for generating various operating voltages and control signals for EPROMs 1 and 2 and is therefore programmed, in use, as an outport port. Programming of these ports consists in sending the following instructions (in BASIC) after power is switched on and prior to using the EPROM programmer:

OUT 203,7; OUT 203,15: OUT 201,7: OUT 201,143.

Note that the order in which these instructions are sent is important. These instructions are necessary to initialise ports A and B. Once the ports are initialised, data may be written into, or read from either port by appropriately addressed OUT and INP instructions, as follows:

INP 202 — read data from PORT B
OUT 200,A — write data (A) to PORT A
INP (200) — read data from PORT A
OUT 202,B — write data (B) to PORT B

Although PORT B data can be read by an IMP instruction, this instruction is not used for PORT B as it is an output port only in this application.

When either PORT A or PORT B is configured as an output port, data, in the form of an 8-bit word addressed to that port by an OUT instruction, will be latched in an internal register for that port in IC4. At the same time, the data will appear at the I/O pins associated with that port and remain there until a new instruction is addressed to that port.

Each bit of an 8-bit word written into PORT B determines a particular operating voltage or control function associated with the operation of the EPROM programmer. The respective bit allocations are shown in the accompanying diagram.

By way of example: Suppose that we want to set up the following conditions: enable RAMs 1 & 2, set Vcc1 to 5V, switch ON power (Vcc1 & Vcc2) to EPROMS 1 & 2, set Vpp to 12.5 V, switch Vpp OFF, set PGM HIGH, bring OE (EPROM 1) LOW, and bring CE (EPROM 1) LOW.

In this case, the required word, in binary form, which we would write into PORT B would look as follows:

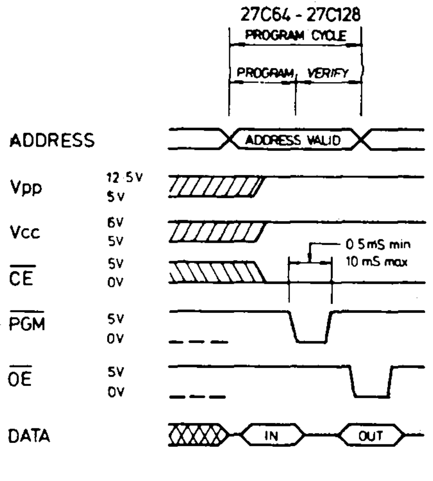
This word corresponds to 26 (hex) or 38 (decimal). The PORT B address is CA (hex) or 202 (decimal). Therefore, to set up PORT B as above, we simply execute the following instruction (in BASIC):

OUT202,38

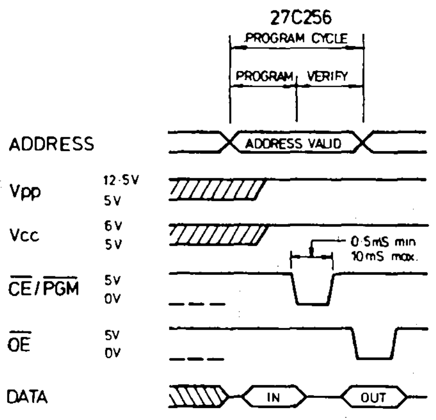
Bit 1, PORT B, determines whether PORT A is an output or an input. When bit 1 is LOW (logic 0), ASTB (pin 16, IC4) is also LOW and PORT A is an output. At the same time OE (pin 22, EPROM 1) is HIGH (logic 1) and any EPROM plugged into the EPROM 1 socket will have its output buffers disabled. That is, data can be written into EPROM 1 via PORT A, as will be the case during an EPROM programming cycle.

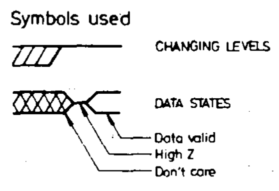
When bit 1, PORT B, is HIGH, ASTB goes HIGH and OE goes LOW. Data can now be read from EPROM 1 via PORT A.

An EPROM programming cycle is an operation in which a specified programming voltage (Vpp) and a programming



Waveforms. At top for 2764/128 type devices, at bottom for 256.





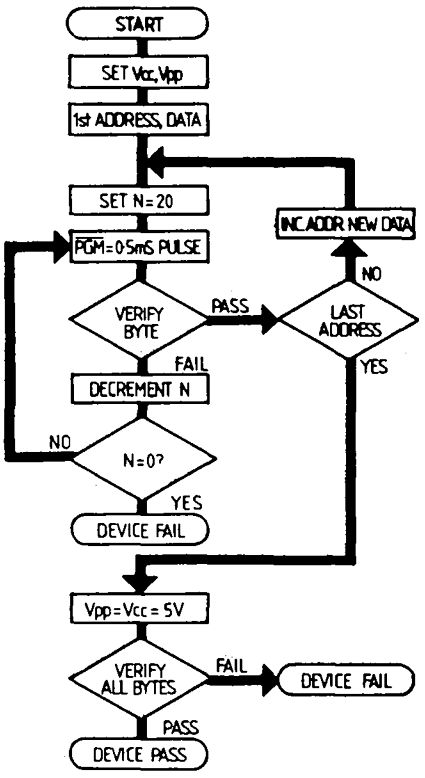
pulse (PGM) of specified duration, together with a desired address and data are applied to the appropriate pins of an EPROM. It is the job of the software to generate the necessary voltages and signals in accordance with the manufacturer's specifications for the EPROM to be programmed. Figure 4 show typical wafeforms for 27C64 to 27C256 type EPROMs as recommended by National Semiconductor.

EPROM manufacturers generally recommend programming algorithms designed to give maximum programming efficiency and speeds when programming their EPROMs. The software developed for this project uses a programming algorithm adapted from one recommended by NATIONAL SEMICONDUCTOR for their CMOS range of EPROMs. This programming algorithm has been found to work well with NMOS EPROMs from other manufacturers. For those interested, a flowchart of the programming algorithm used by the software developed for this project is given in Fig. 5.

It is anticipated that most constructors in this project will take advantage of the software offer made in this article. Space does not permit a full description of the features which this software offers although a brief outline has already been given. A lot of effort has been put into its development to make this EPROM programmer project versatile, easy to use and as foolproof as possible.

If, however, you choose to develop your own software for this programmer, be careful to ensure that it will generate the correct voltage levels and programming (PGM) pulse widths in accordance with the various EPROM manufacturers' specifications. Some hints for you: write any subroutine for generating the EPROM programming pulses in machine language (for Z80 microprocessors) and ensure that the sub-routine starts with an 'interrupt disable'. If possible, check out all waveforms on a CRO before trying out the programmer on an EPROM.

Whether you purchase the software or



Software flow chart

write your own, be careful to observe the following precautions:

1. Never insert or remove an EPROM from a socket while power is ON (as indicated by the LED lighting up.
2. Always ensure that the correct programming voltage (Vpp) is selected before programming an EPROM. Different manufacturers specify different programming voltages. The following is a list of EPROM types vs manufacturer and programming voltages derived from information given in the 1987 JAYCAR catalogue and reproduced here with their kind permission (note that ETI cannot accept any responsibility for any errors which may occur in this list) ●

Eproms can be ordered from the author at Beryl Road, Gulgong, NSW 2852.

EPROM/Manufacturer		**programming voltage**
2764	Intel, Fairchild, OKI, NEC, TI, Toshiba, AMD, Fujitsu, Hitachi	21V
2764A	Intel, AMD	12.5V
27C64	National	12.5V
27128	Intel, AMD, Fujitsu, NEC, Toshiba, TI, Mitsubishi (M5L27128K & M5M27C128K)	21V
27128	ADC, AMD, NMC27CP128, NMC27C128C (National)	12.5V
27256	Intel, Atmel, NMC27C256 National	12.5V
TMM27256D	Toshiba, Fujitsu (27256 & 27C256)	21V

If in any doubt, always check with the manufacturer.