Robert J. Cotter
The Johns Hopkins
School of Medicine
Baltimore MD 21205

The Elf EPROMmer

Got the Giant Board? Then an EPROM programmer is easy.

If you bought the Giant Board and the 4K memory board for your Elf II, then you can build a simple EPROM programmer with only a few inexpensive ICs, transistors and resistors. Then you can store all those home-made programs, your own monitor system or Tiny BASIC; and your computer will be ready to run your programs as soon as the power switch is turned on.

The Giant Board makes it easy since it contains a latched output port that can be connected to your programmer and a ROM monitor system for accessing the programs you will write onto your EPROM. The 4K board is necessary to provide an area to load your program for copying.

Also, it provides latched and buffered high-order address bits you will need to address the 1K EPROM memory and for selecting between EPROMs. The high-order bits can be brought down to unused pins on your 88-pin bus, so that you can conveniently build your programmer on a board that plugs directly into the Elf bus.

The EPROM programmer described here uses two EPROM sockets: one for programming and one for reading. In the programming mode, the EPROM is treated as an output device; while in the reading mode it becomes a randomly accessible memory. The advantage of this approach is that the READ socket can be used not only to verify that a program has been loaded correctly, but also to run the programs stored on your EPROM.

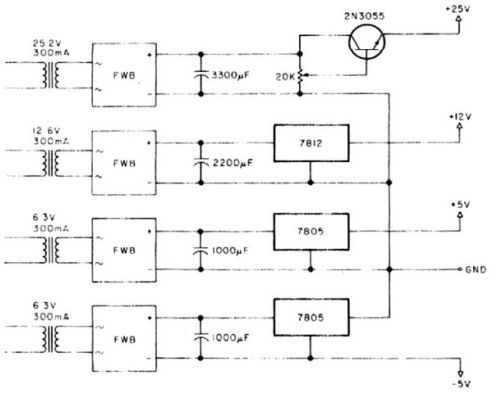

Fig. 2. Power supply schematic.

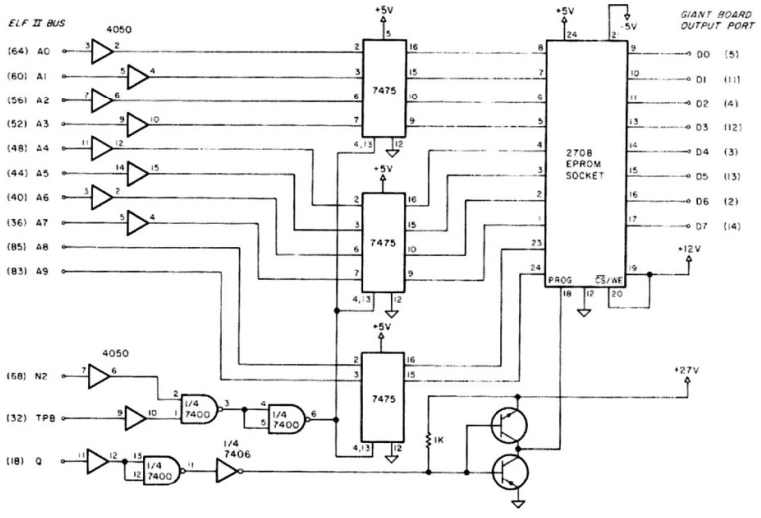

Fig. 1. EPROM programming circuit.

Programming the 2708 EPROM

Fig. 1 shows a schematic of the programming circuit. The address lines (A0 to A9), the timing pulse (TPB), the I/O select line (N2) and Q come directly from the Elf bus. The pin connections are shown in parentheses. The data lines (D0 to D7) are available on the Giant Board on a 14-pin DIP socket.

If a similar socket is mounted on the programmer, then a 14-pin ribbon cable connector can be used to connect this out-

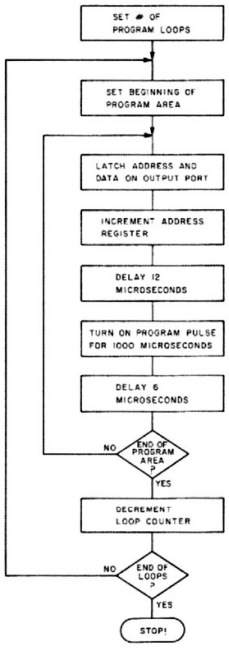

Fig. 3. Program loop flowchart.

put port to your circuit. Other connections include ±5 V, +12 V and 27 V. A power supply that provides these voltages is shown in Fig. 2.

How It Works

In the programming mode, chip select/write enable ($\overline{CS}$/WE) is held high at +12 volts by direct connection to the 12 volt supply. (This is pin 20 on the 2708 EPROM socket.) Addresses are placed sequentially (from 000 to 3FF) onto the EPROM whenever N2 and TPB are high. A 67 output instruction is used to raise N2. The 7475 four-bit latch latches its outputs as the enable pulse at pins 4 and 13 falls. Therefore, N2 must be ANDed with TPB so that this pulse goes low when correct addresses are presented to the 2708.

The 67 instruction was chosen since this also controls the output port on the Giant Board. This instruction, therefore, loads and latches the correct data bits onto the EPROM at the same time as the address bits.

Twelve microseconds after a particular address and data have been latched onto the 2708, the Q line goes high and turns on the 27 volt program pulse to write the data at that location. A timing loop holds this pulse on for 1 millisecond. When the program pulse goes low, the next address and new data are placed on the EPROM and the sequence is repeated for all 1024 locations. When all locations have been written once, this defines a program loop, which must be repeated 100 times to ensure that the data has been correctly written in. A flowchart demonstrating this sequence of events is shown in Fig. 3.

Program A shows the hexadecimal listing for programming the 2708 EPROM. In step 2, note that the area to be copied is located from 0400 to 07FF. If you wish to use this program to copy a different 1K area, then the starting address, 0400, can be changed, as long as the bottom ten bits of this address are all 0s.

Also, when the 2708 is erased, it contains 1s in all locations. Therefore, if you are not copying all 1024 locations, the remaining spaces should be filled up with FFs, so that you can add other programs later. For example, I entered my programming program into locations 0400 to 0420, filled locations 0421 to 07FF with FFs and now have an EPROM that will program other EPROMs. Having the two separate sockets makes this possible.

location	bytes	steps	comments
00	F8 6A A1	1	set programming loop counter
03	F8 04 B2	2	set beginning address of
06	F8 00 A2		area to be copied
09	E2	3	M(R2) = MX
0A	67	4	latch DATA and ADDRESS onto output port; R2 + 1
0B	C4 C4	5	delay
0D	7B	6	turn on program pulse
0E	F8 3B A3	7	load 3B into register 3
11	23	8	R3 − 1
12	83	9	R3.0 → D
13	3A 11	10	GO TO 11 if D ≠ 00
15	7A	11	turn off program pulse
16	C4	12	delay
17	92	13	R2.1 → D
18	FB 08	14	R2.1 × OR 08
1A	3A 0A	15	GO TO 0D if D ≠ 00
1C	21	16	R1 − 1
1D	81	17	R1.0 → D
1E	3A 03	18	GO TO 03 if D ≠ 00
20	00	19	STOP!

Program A. Hex listing for programming the 2708 EPROM.

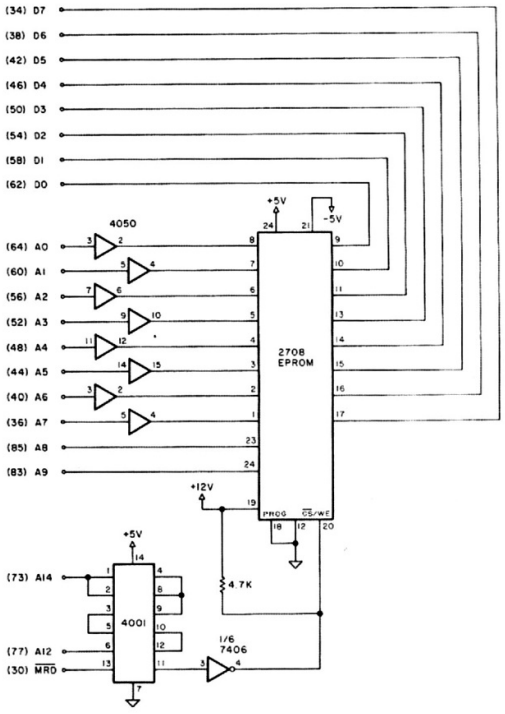

Fig. 4. Circuit modification.

Verifying and Using Your EPROM

By adding one more IC and a new socket to your board, you can verify your program or use your EPROM as a read-only memory. The schematic is shown in Fig. 4. (Address lines A0 to A7 use the same buffers as in Fig. 1.) In this socket, the PROGRAM pin is grounded and $\overline{CS}$/WE is pulled low by A14, $\overline{A12}$ and MREAD. This makes its location 4000 to 43FF. The 2708 data outputs can be connected directly to the Elf bus, since these outputs are Tri-state.

You can verify that the program has been loaded correctly by using the Elf's monitor on the Giant Board. Load the program, C0 F0 00, to get into the monitor. Press the RUN switch and load 01 40 00 to examine the memory starting at location 4000. Each time you press and release the input switch you will alternately observe the low-order address bits and the data bytes on the hex display. If the first dozen or so bytes are correct, then you have most likely made a correct copy, and you should then try to run the EPROM programs.

To run a program starting at the first location, load C0 40 00 and press the RUN switch. You can run a program at any other location by loading C0 and the correct address from 4000 to 43FF.

Some Comments

When you build your EPROM board, you might consider building several READ sockets. You can then select between EPROMs by decoding part or all of the upper address bits (A10 to A15). A circuit for accomplishing this for a 4K EPROM memory is shown in Fig. 5. The 74C154 decoder decodes A10 to A13 and produces valid outputs when A14 is high and A15 is low. The selected outputs themselves are low and are gated with $\overline{MRD}$ to produce a positive pulse for selecting the correct EPROM. Only four of the 16 possible outputs are used. The 7406 inverter then pulls $\overline{CS}$/WE low to read the EPROM. This particular inverter must be used, since it has a high voltage, open collector output.

0000-0FFF	4K RAM #1
1000-1FFF	4K RAM #2
2000-2FFF	4K RAM #3
4000-43FF	1K EPROM
4400-47FF	1K EPROM
4800-4BFF	1K EPROM
4C00-4FFF	1K EPROM
F000-F0FF	ROM monitor

Table 1. Elf II memory allocation.

When locations for the EPROMs are selected, addresses beginning with F (F000 is the ROM monitor) and any RAM address must be avoided. Table 1 shows one possible arrangement of memory locations for the Elf II, if the five Elf bus sockets are occupied by three 4K RAM boards, the Giant Board and the EPROM board as described here.

There are several inexpensive EPROM programmers currently on the market. The September 1978 issue of *Kilobaud* features an excellent article by James Grina on a 2708 programmer for the KIM-1 ("Super Cheap 2708 Programmer," p. 100). His approach is to generate the addresses for the EPROM using three 4-bit 74193 counters, thereby eliminating the need for bringing out the ten address lines.

In order to use this circuit on the Elf II, another output port (in addition to the input and output ports available on the Giant Board) has to be constructed. This port would be used to produce the program pulse, set the $\overline{CS}$/WE state and increment and reset the counters.

This approach is also used by an EPROM programmer developed by Optimal Technology, which uses three I/O ports when connected to the COSMAC VIP. However, I chose the method described in this article because the high-order addresses were readily available, and once they were brought onto the board, I had the possibility of randomly (and not sequentially) accessing the EPROM. ■

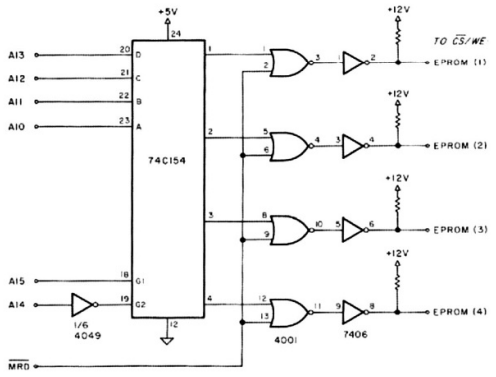

Fig. 5. 74C154 decoder circuit.