Program Those 2708s!

Robert Glaser
3922 Algiers Rd
Randallstown MD 21133

Erasable programmable read-only memories (EPROMs) can be used to great advantage in many microcomputer applications. One of the stumbling blocks to more widespread hobbyist use of EPROMs has been the difficulty of programming them. Several companies offer programming services, but this can be time-consuming as well as expensive.

One of the first EPROMs to become available was the 1702 device, which is structured as 256 words by 8 bits. This EPROM is indeed difficult to program. All of its address and data lines must be switched at 50 V levels, requiring a multitude of level-shifting transistors, in addition to the timing logic. Although it is possible to construct a programmer for the 1702, it is certainly not simple.

Salvation for the hobbyist came with the Intel 2708 EPROM. This device sports 1 K words by 8 bits of memory, four times the capacity of the 1702. It requires power supplies of +5 V, +12 V, and −5 V. For read operation, all that is required is to supply the address lines with the desired memory address, and select the individual EPROM device by grounding the chip-select input. The outputs appear on the data lines.

The greatest advantage of this 2708-type memory is its program-

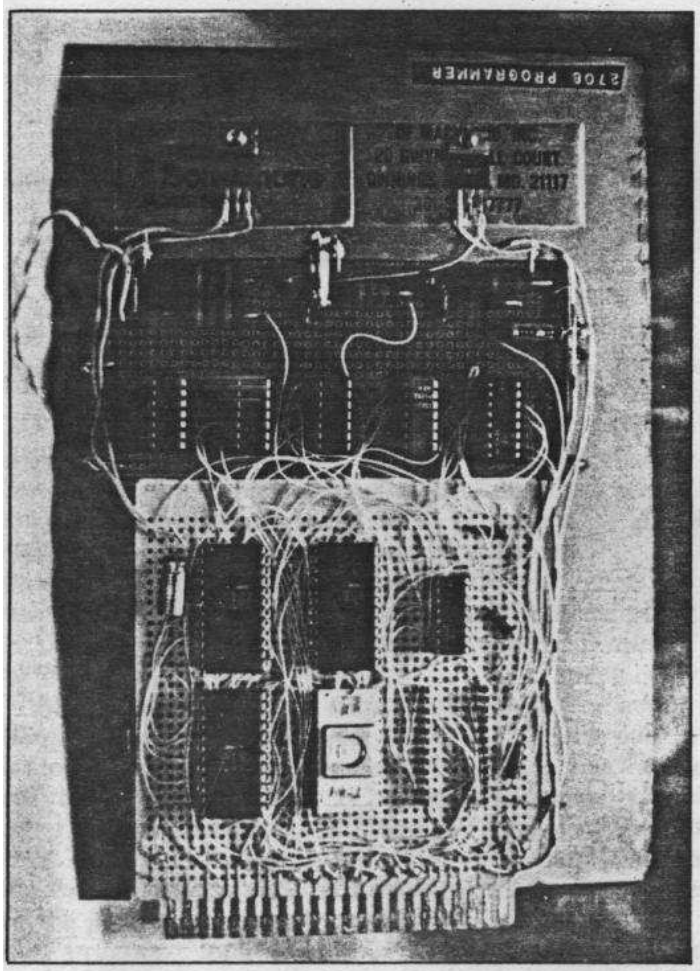

Photo 1: *Front side of the EPROM programming circuit board. Components may be identified from diagram of figure 4. A Radio Shack 44-pin card forms the base of the board, which has had other sections added to it. TO-220 packages at top are voltage regulators.*

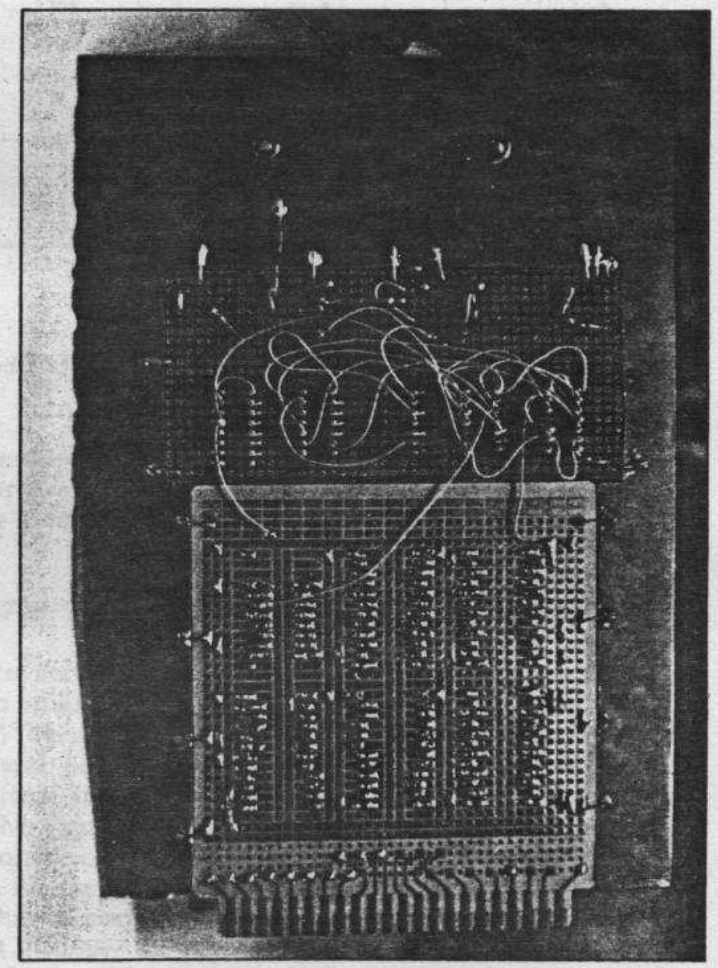

Photo 2: *The back side of the EPROM programming circuit board. The author wishes to thank Marc Leavey MD, WA3AJR for performing the photography.*

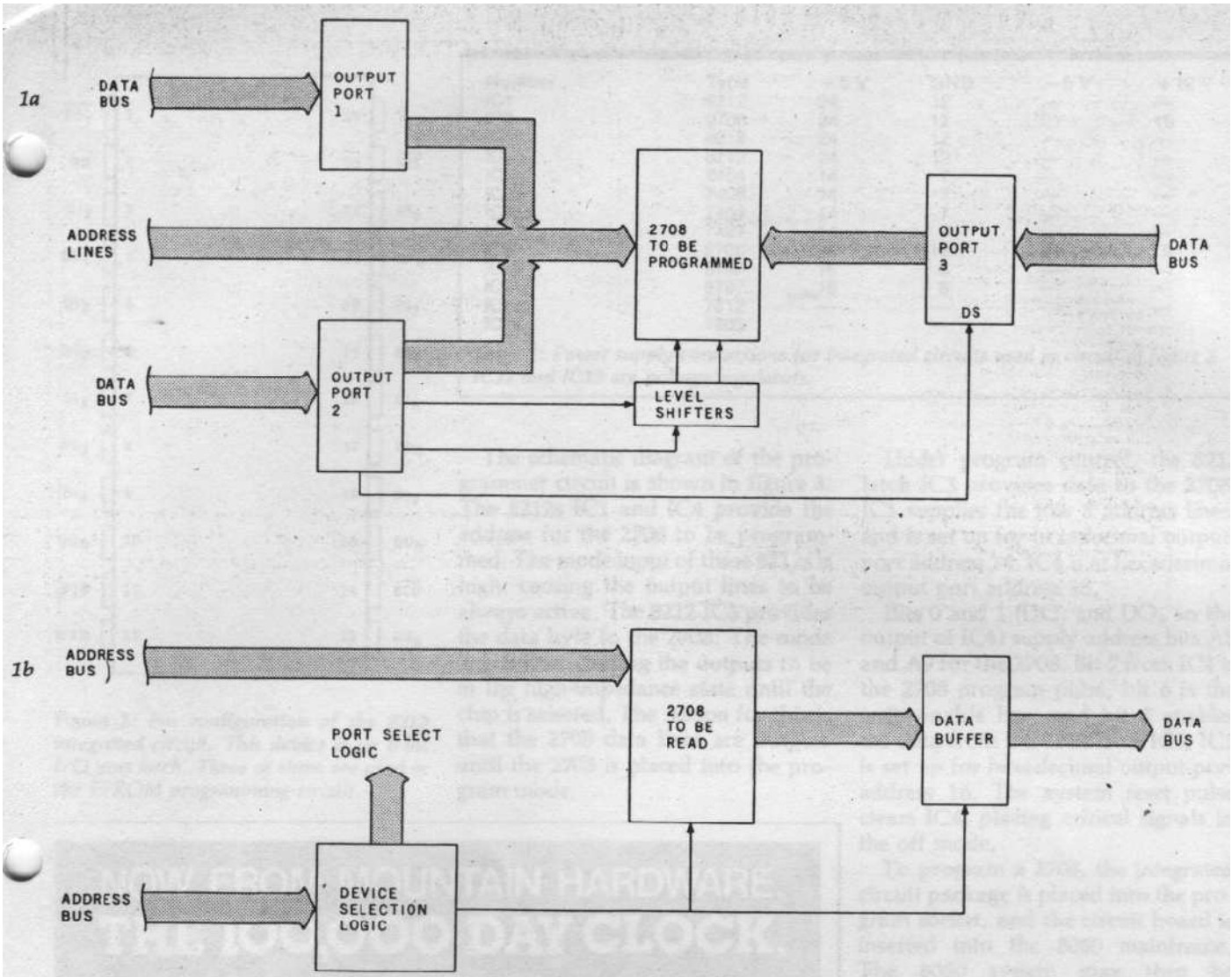

Figure 1: *Block diagrams of the 2708 EPROM programming circuit (1a) and 2708 reading circuit (1b).*

ming simplicity. All address and data lines need only be supplied with transistor-transistor logic (TTL) voltage levels. Two lines must be pulsed at nonTTL voltage levels. The write-enable line must be raised to +12 V, and the program pulse rises to +26 V.

After erasure with an ultraviolet lamp, all bits of the 2708 are in the logic 1 state. Programming consists of selectively changing the 1s to 0s. After the write-enable line is raised to +12 V, each byte is set up by applying the address and data information to the proper pins, and then pulsing the program input. The proper method is to sequence through all of the addresses many times. Each run through all addresses is called a *program loop*. The specifications of the 2708 device call for the number of program loops, multiplied by the duration of the program pulse, to form a total pulse time of at least 100 ms.

Microcomputer 2708 Programming

A simple way to accomplish the programming is to utilize a microcomputer system. With a small program routine, several output ports and some level shifters, it is easy to program the EPROM. Figure 1 shows the block diagram of the circuit I use in my 8080 system for the programming operation. Output port 1 and part of output port 2 supply the address to the 2708 device to be programmed. Output port 3 feeds the desired data to the 2708. Part of output port 2 and some level shifters provide the programming pulses for the device.

Each output port is an 8212 latch. The 8212 device is a general purpose I/O (input/output) port. The pin connections are shown in figure 2. The output of the latch is 3-state. If the mode input is high, the outputs are always enabled. When the device is selected by placing a low on $\overline{DS1}$ (active-low, device-select line) and a high on DS2 (active-high line), whatever data is present at the data input (DI) lines is latched and appears at the data output (DO) lines.

If the mode input is low, the outputs are in the high-impedance state until the device is selected. In this case, the data is latched by a signal on the strobe line. The 8212 places little loading on the data bus, and is quite suitable for the output ports used in this project.

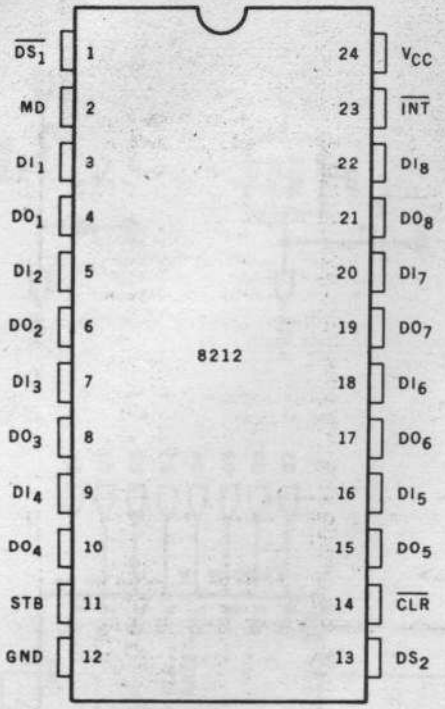

Figure 2: *Pin configuration of the 8212 integrated circuit. This device is an 8-bit I/O port latch. Three of them are used in the EPROM programming circuit.*

Number	Type	+5 V	GND	−5 V	+12 V
IC1	8212	24	12	—	—
IC2	2708	24	12	21	19
IC3	8212	24	12	—	—
IC4	8212	24	12	—	—
IC5	7404	14	7	—	—
IC6	7408	14	7	—	—
IC7	7404	14	7	—	—
IC8	7427	14	7	—	—
IC9	2708	24	12	21	19
IC10	8T97	16	8	—	—
IC11	8T97	16	8	—	—
IC12	7812	—	—	—	—
IC13	7805	—	—	—	—

Table 1: *Power supply connections for integrated circuits used in circuit of figure 3. IC12 and IC13 are voltage regulators.*

The schematic diagram of the programmer circuit is shown in figure 3. The 8212s IC1 and IC4 provide the address for the 2708 to be programmed. The mode input of these 8212s is high, causing the output lines to be always active. The 8212 IC3 provides the data byte to the 2708. The mode line is low, causing the outputs to be in the high-impedance state until the chip is selected. The reason for this is that the 2708 data lines are outputs until the 2708 is placed into the program mode.

Under program control, the 8212 latch IC3 provides data to the 2708. IC1 supplies the low 8 address lines, and is set up for hexadecimal output-port address 14. IC4 is at hexadecimal output port address 15.

Bits 0 and 1 (DO_1 and DO_2 on the output of IC4) supply address bits A8 and A9 for the 2708. Bit 7 from IC4 is the 2708 program pulse, bit 6 is the write-enable line, and bit 5 enables the data from the 8212 latch IC3. IC3 is set up for hexadecimal output-port address 16. The system reset pulse clears IC4, placing critical signals in the off mode.

To program a 2708, the integrated circuit package is placed into the program socket, and the circuit board is inserted into the 8080 mainframe. The 8080 system may then be powered up, and the program run. The 26 V power supply should be turned on just prior to supplying the address to the program.

It is important not to apply the high voltage before the system is powered up and reset. After programming, the sequence should be followed in reverse. The 26 V supply should be turned off, the computer turned off, the board unplugged, and the 2708 removed.

To read what has been written into EPROM, the device is plugged into the *read* socket. Hexadecimal address 0000 is used. If you already have an EPROM board which can read 2708s, then this portion of the circuit can be deleted. The inhibit line prevents the 2708 from being selected.

Construction

Construction will depend upon your particular system. My 8080-based system was built using 44-pin edge connectors. This permits the use

Text continued on page 206

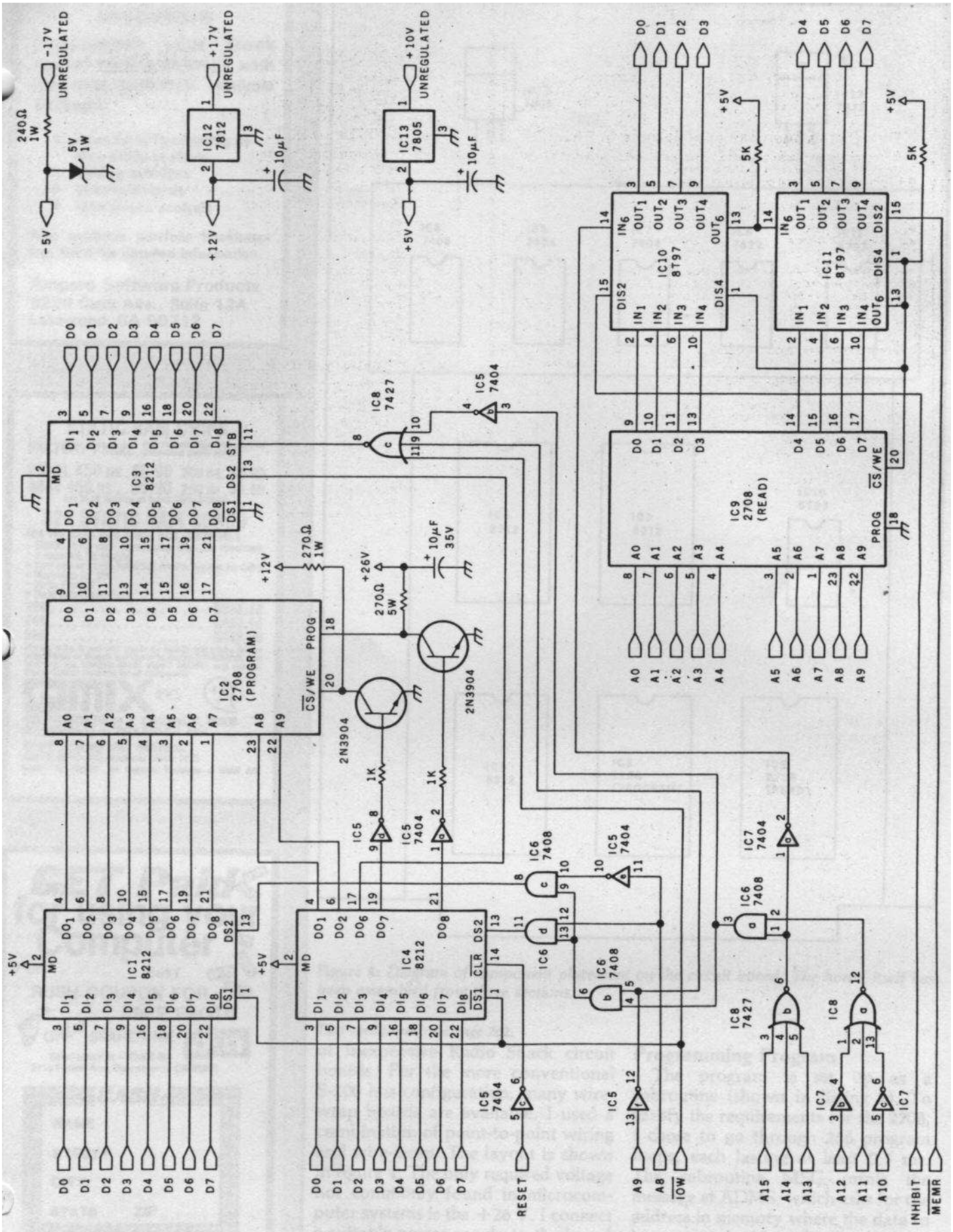

Figure 3: *Schematic diagram of the 2708 programming circuit. At right are shown voltage regulation schemes for power supplies.*

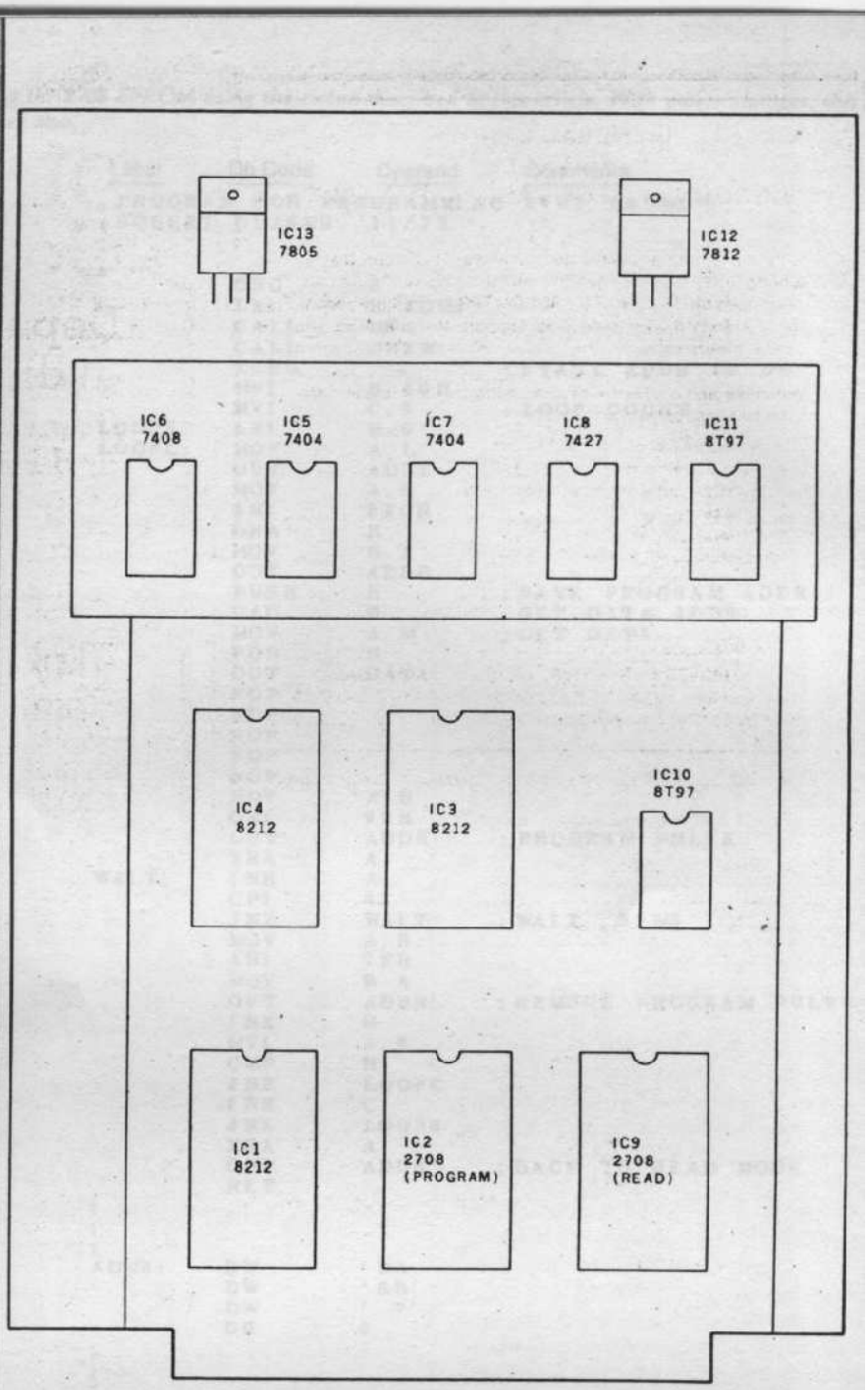

Figure 4: *Diagram of component placement on the circuit board. The board itself has been assembled from three sections.*

Text continued from page 202:

of inexpensive Radio Shack circuit boards. For the more conventional S-100 bus configuration, many wire-wrap boards are available. I used a combination of point-to-point wiring and wire-wrap. The layout is shown in figure 4. The only required voltage not commonly found in microcomputer systems is the +26 V. I connect a suitable power supply to the board when it is needed.

Programming Program

The program is set up as a subroutine (shown in listing 1). To satisfy the requirements for the 2708, I chose to go through 256 program loops, each lasting at least 0.5 ms. The subroutine MSG prints the message at ADMS, which asks for the address in memory where the data to be programmed into the 2708 is to be found. It is assumed that 1 K bytes of

Listing 1: *8080 subroutine for programming the 2708 EPROM using the circuit described in this article. With minor changes, this routine can be used to program 2716 devices also.*

```
Line  Address  Object Code      Label   Op Code  Operand   Comments
0001  0000                     ; PROGRAM FOR PROGRAMMI NG 2708 PROMS
0002  0000                     ; ROBERT GLASER, 11/77
0003  0000                     ;
0004  0000                     ;
0005  0000                             ORG      0
0006  0000  21 47 00                   LXI      H, ADMS
0007  0003  CD 79 ED                   CALL     MSG
0008  0006  CD 85 ED                   CALL     GHXW
0009  0009  EB                         XCHG                ;START ADDR TO DE
0010  000A  06 60                      MVI      B, 60H
0011  000C  0E 00                      MVI      C, 0       ;LOOP COUNT
0012  000E  21 00 00           LOOPS:   LXI      H, 0
0013  0011  7D                 LOOPC:   MOV      A, L
0014  0012  D3 14                      OUT      ADDL
0015  0014  78                         MOV      A, B
0016  0015  E6 FC                      ANI      0FCH
0017  0017  B4                         ORA      H
0018  0018  47                         MOV      B, A
0019  0019  D3 15                      OUT      ADDH
0020  001B  E5                         PUSH     H          ;SAVE PROGRAM ADDR
0021  001C  19                         DAD      D          ;GET DATA ADDR
0022  001D  7E                         MOV      A, M       ;GET DATA
0023  001E  E1                         POP      H
0024  001F  D3 16                      OUT      DATA
0025  0021  00                         NOP
0026  0022  00                         NOP
0027  0023  00                         NOP
0028  0024  00                         NOP
0029  0025  00                         NOP
0030  0026  78                         MOV      A, B
0031  0027  F6 80                      ORI      80H
0032  0029  D3 15                      OUT      ADDH       ;PROGRAM PULSE
0033  002B  AF                         XRA      A
0034  002C  3C                 WAIT:   INR      A
0035  002D  FE 2A                      CPI      42
0036  002F  C2 2C 00                   JNZ      WAIT       ;WAIT .5 MS
0037  0032  78                         MOV      A, B
0038  0033  E6 7F                      ANI      7FH
0039  0035  47                         MOV      B A
0040  0036  D3 15                      OUT      ADDH       ;REMOVE PROGRAM PULSE
0041  0038  23                         INX      H
0042  0039  3E 04                      MVI      A, 4
0043  003B  BC                         CMP      H
0044  003C  C2 11 00                   JNZ      LOOPC
0045  003F  0C                         INR      C
0046  0040  C2 0E 00                   JNZ      LOOPS
0047  0043  AF                         XRA      A
0048  0044  D3 15                      OUT      ADDH       ;BACK TO READ MODE
0049  0046  C9                         RET
0050  0047                     ;
0051  0047                     ;
0052  0047                     ;
0053  0047  41 44              ADMS:    DW       'DA
0054  0049  44 52                      DW       'RD'
0055  004B  3F 20                      DW       ' ?'
0056  004D  00                         DB       0
0057  004E                     ;
0058  004E                     ;
0059  004E                     ;
0060  004E                     MSG:     EQU      0ED79H
0061  004E                     GHXW:    EQU      0ED85H
0062  004E                     ADDL:    EQU      14H
0063  004E                     ADDH:    EQU      15H
0064  004E                     DATA:    EQU      16H
%
```

data are to be written into the EPROM from that starting point. If the 2708 is to be only partially written, the unused portion of source memory should be filled with the hexadecimal value FF.

Subroutine GHXW gets the 16-bit value which is input in hexadecimal, and places it in register pair HL. The starting address is then moved to DE. Throughout the program this remains the same. Register pair HL contains the actual address applied to the 2708.

LOOPS (loop start) is the beginning of a program loop. At LOOPC (loop continue) the cycle begins. First the address is set up at ports ADDL and ADDH. The data is then fetched and output at the DATA port. Several no-operation instructions are included to guarantee the timing specifications of the 2708.

The program pulse is then applied, and a timing loop of 0.5 ms is entered at WAIT. The program pulse is removed, and the current address is

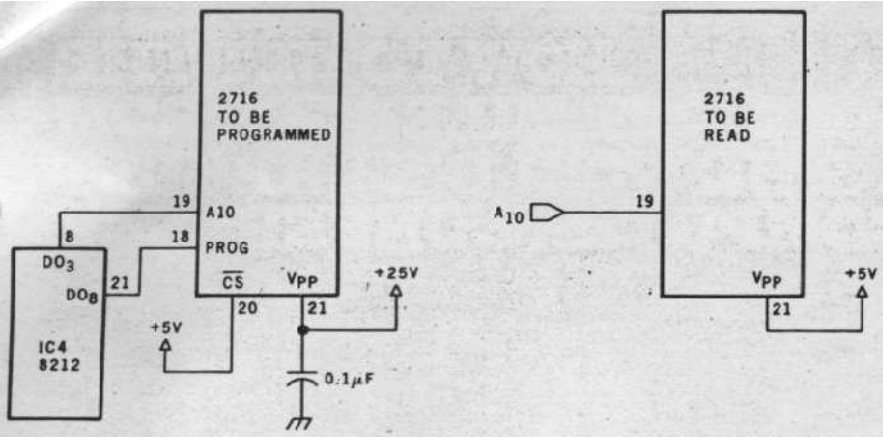

Figure 5: *Modifications to the circuit of figure 3 that enable the programming of the 2716-type EPROM. Modifications to the software are also necessary.*

examined to see if a program loop is finished. If not, the control loops back to LOOPC. If the loop is finished, the loop count is checked to see if all 256 loops have been completed. If not, control goes back to LOOPS.

When the procedure is finished, the 2708 is returned to the read mode, and the routine returns to the calling program. To be on the safe side, timing values are longer than necessary. With the 8080 running at 2 MHz with one wait state, the routine takes 3 minutes and 6 seconds.

Variations

Other EPROMs could be programmed with this setup, as well as 2708s. By changing the value 4 to a 2 in line 42 of the program, 2704s can be programmed with no other modifications. To program 2716s, some other modifications need to be made. The 2716 is a 2 K word by 8 bit EPROM and has some advantages over the 2708. It requires only a +5 V supply for read operation. For programming, the program pulse need only be a TTL level voltage. The high voltage is not pulsed.

Figure 5 shows the necessary circuitry changes to accommodate the 2716. The high voltage applied to pin 21 is +25 V, not the +26 V used for the 2708. Pin 19 is the eleventh address line.

The 2716 needs only a single program loop, but the program pulse should be 50 ms or longer. The program should be modified. Delete lines 11, 45, and 46. The value 4 in line 42 should be changed to an 8, and the delay loop at WAIT should be surrounded by an external loop of 100 to change the 0.5 ms to 50 ms.

To use non8080 systems for programming the EPROM, all that need be done is to reconcile the buses. For 6800-type systems with no discrete output ports, the output ports would have to be addressed as memory.

I have programmed dozens of 2708s with this setup and have had no problems. My application has been with dedicated 8080 controllers. One such controller is used in the local amateur radio repeater to perform various functions. Many program versions were used in this application, since control and autopatch codes are all contained in the programmable read-only memory, which led to many program revisions. The 2708 programmer board was called upon many times. ■