AN 1802-BASED EPROM PROGRAMMER

Hardware and software to program, copy, and verify data in a 2708 EPROM, using an 1802 microprocessor

BY LARRY BREGOLI

MOST computer fans have a few favorite programs such as monitors, machine-language utilities, and even high-level languages that are used so often that storing them in ROM would be beneficial. When in this permanent memory, these programs can be accessed at any time, without the often tedious loading procedure. The ROM can even work from a power-up state.

There are numerous ROM programmers available for most systems, but few for those based on the 1802 processor. The programmer described here will help to alleviate this shortage. Although designed for 2708 devices, it can be user modified to accept 2716 devices. This permits doubling the ROM size with no increase in board space.

Circuit Operation. A simplified block diagram of the EPROM programmer is shown in Fig. 1. When the I/O Select Line calls for the programmer, three events occur. In conjunction with the arrival of the $\overline{MWR}$ (memory write) signal, the Program-Cycle-Latch causes the Address-Latches to: (1) hold the address currently on the address bus; (2) remove the data latches from their three-state mode to allow them to latch and hold the data currently on the data bus; and (3) via the Program-Pulse-Circuit place the 2708 in the write-enable mode so that timing pulses from the CPU will allow a programming pulse to be applied.

The eight Three-State Switches remain in their high-impedance state until the 2708 is read. At this time the $\overline{MRD}$ (memory read) pulse allows data from the 2708 to be placed on the system data bus. The read function is handled in this manner because it is easier than using the 2708's $\overline{CS}$ (chip select) circuit that requires three voltage levels. The 2708 data lines go three-state when the $\overline{CS}$ voltage is at +5 volts, and when the data is read from the 2708, the $\overline{CS}$ line must be low (the same as ordinary RAM). The 2708 is placed in the write mode by bringing the $\overline{CS}$ line to +12 volts.

The complete schematic is shown in Fig. 2. Some of the IC grounds shown are functionally necessary while others are required because all unused CMOS inputs must be terminated with either a ground or +5 volts to avoid noise problems. The reset (pin 4) of *IC1*, the program cycle latch, is connected to the system reset line, while the clock input (pin 3) is connected to one of the N lines from the 1802. The Q signal at pin 1 of *IC7A*, is the serial-output line of the 1802 and is set high or low by the system software. Note that the Q pulses cannot get to the pulse circuit unless *IC7A* is enabled by the pin 1 output of *IC1*. The only TTL device used is *IC9*.

The 2708 both sinks and sources current at its programming input (pin 18) during programming and this must be included in the circuitry or the voltage at this pin will be pulled up by the sink current during the low-voltage level of the program pulse. Pin 4 of *IC9* provides this function. Data latches *IC2* and *IC3* may be replaced with 4076's if you want to stick with all 4000-series IC's. The $\overline{MWR}$ and $\overline{MRD}$ pulses are both gated by the $\overline{CE}$ (chip enable) signal from the addressing system. The $\overline{CE}$ does not have to select the 2708 directly since the EPROM is always in the read mode when not being programmed. There's an additional advantage in this approach, since in the read mode the 2708 uses the minimum standby power without a special power-down circuit. The data-latching-strobe to *IC2* and *IC3* is a function of TPB, a timing pulse which occurs when the data is valid, via *IC7B* and *IC10C*.

The RC network coupled to programming pin 18 of the 2708 shapes the programming pulse so that voltage spikes will not occur inside the 2708 possibly creating an error in the programmed data.

Timing. The 2708 timing must be programmed for a minimum of 100 ms for each bit, and each bit can be programmed only for a maximum of 1 ms at any given time. This means that each bit must be programmed at least 100 times.

Another criterion for programming the 2708 is that, each time a portion of the memory is programmed, the remainder of the bits must also be reprogrammed. This will be described further under "Software." The only other timing consideration to account for is the minimum 10 μs setup time for the address and data information. All of these timing requirements are

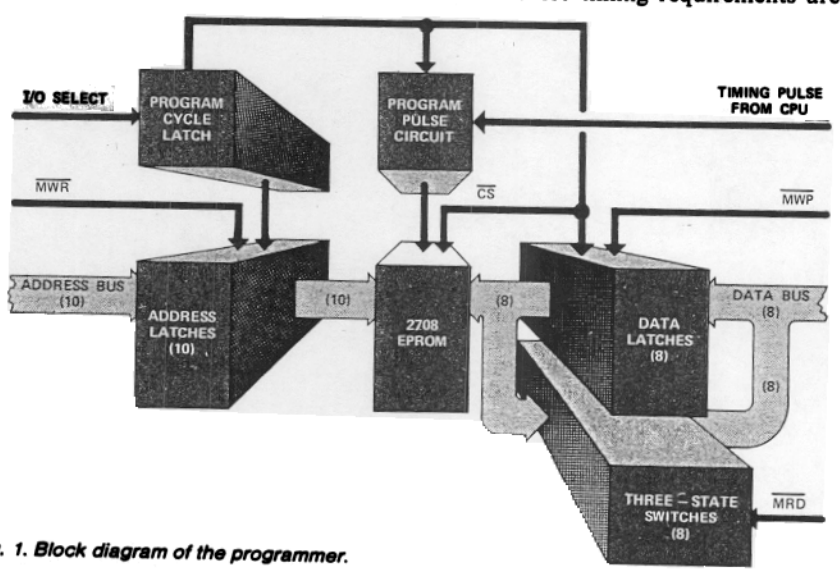

Fig. 1. Block diagram of the programmer.

handled by the programming software to keep the hardware simple.

Software. A fully erased 2708 contains all "ones" and the programmer replaces these with zeros where specified during the programming mode. Once a zero bit is programmed, it cannot be changed back to a one by programming. If you make an error and put a zero in the wrong location, the only way it can be rectified is erasing the EPROM with UV to produce all ones again, and then re-start the programming.

About 1K of RAM is required to assemble and hold the data to be programmed into the EPROM. This RAM space must be initially programmed with all ones to avoid any unwanted zeros from appearing when programming is started. This is accomplished by copying the contents of the erased 2708 in the programming socket into the RAM space.

This method is also convenient for adding data to an EPROM which has been partially programmed. In this case, the existing programs in the EPROM are also copied into RAM

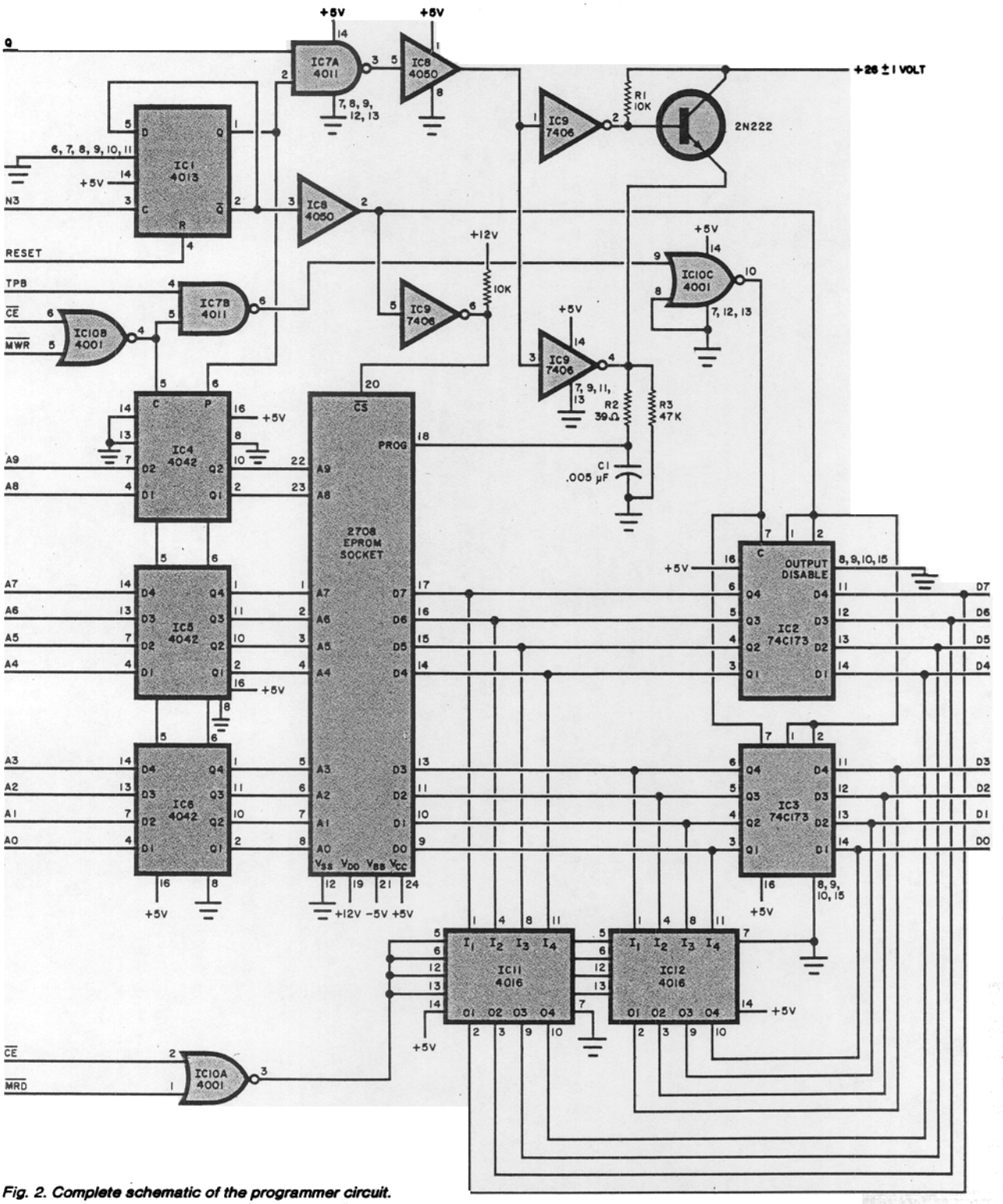

Fig. 2. Complete schematic of the programmer circuit.

and are reprogrammed back into the EPROM along with the added data. Keep in mind that, each time new data is added to the EPROM, all 1024 bytes must be programmed.

The three subroutines to be covered are called using conventional 1802 call and return techniques. The program locations are arbitrarily chosen. The items in parentheses are bytes that would have to be changed if any locations are modified. The COPY subroutine is shown in Listing 1. This is a simple move program of which many versions exist. In this case, the initiating routine sets pointers to the beginning address of the EPROM to be programmed and to the beginning address of the RAM area used to store the program. The bytes are then copied, one by one, from the EPROM into the RAM until all 1024 bytes in the EPROM have been copied.

New data to be programmed into the EPROM is placed in any available RAM space. When the new data has been entered into RAM, the BURN EPROM subroutine of Listing 2 is called to program data into the EPROM.

The BURN EPROM substitute should contain all the timing information needed to properly program a 2708 EPROM including the data and address setup times. When the subroutine is first entered, a counter is loaded with the number of times all 1024 words will be programmed. This number can vary from 100 to 1000 since the width of the programming pulse may be set from 0.1 to 1.0 ms, and the product must always be equal to or greater than 100 ms. An I/O pulse is then sent to the program-cycle-latch by way of line N3 to activate the programmer. Address pointers are then set to point to the starting addresses of both the EPROM to be programmed and RAM area holding the data. A byte, read from the RAM area, is written into the data-input latches, and simultaneously latches the address latches. Before application of the programming pulse, the 2708 needs a minimum of 10 μs to allow for data and address settling times. To accomplish this time delay, simply insert NOP instructions in the program before applying the programming pulse. The pulse-time-counter is then loaded with a number that describes the programming pulse width.

The Q line from the 1802 is then set high causing the programming pulse to begin. Next, the pulse-time-counter is decremented to zero, dictating the width of the programming pulse. The Q line is then set low, ending the pulse. The loop counter is decremented each time all 1024 words have been programmed; then a test for a zero in the loop counter is made. When the loop counter is zero, an I/O pulse is again sent to the programmer to take it out of the programming mode. It requires approximately 103 seconds to program all 1024 words into a 2708.

Some EPROMS may need more programming time to guarantee that all bits are properly burned. To do this, the number in the loop counter must be increased. If you have the BURN EPROM program already in the EPROM, simply burn the new one again using the same program.

To determine if you have burned all the bits properly, they can be tested using the VERIFY program shown in Listing 3. This routine compares each byte in the newly burned EPROM with the original data in RAM. If a mismatch is found, the Q line is set. The erroneous byte location can be shown on a monitor or a hexadecimal display. To determine if the VERIFY program is working properly, inset an intentional error in the RAM area and run the VERIFY program. ◇

LISTING 1—COPY

```
Location   Code          Comment
E4 1E      F8 (C8) B8    : Point R8 to RAM Area
E4 21      F8 (D0) B9      and
E4 24      F8 (00) A8 A9   R9 to beginning of EPROM
E4 28      09            : Get EPROM byte and
E4 29      58              put it in RAM
E4 2A      19 18         : Increment EPROM & RAM
E4 2C      99 FB (D4)    : If it's not the last
E4 2F      3A (28)         byte, get another
E4 31      D5            : RETURN
```

LISTING 2—BURN EPROM

```
Location   Code          Comment
E4 32      F8 65 AA      : Set loop counter to 101
E4 35      EB 63 2B      : Activate Programmer (N lines 3)
E4 38      F8 (C8) B8    : Point to starting locations
E4 3B      F8 (D0) B9      of RAM and EPROM
E4 3E      F8 (00) A9 A8
E4 42      2A            : Decrement loop counter
E4 43      8A            : If loop counter is
E4 44      32 (5C)         zero then exit
E4 46      08            : Get RAM byte
E4 47      59            : Write byte into latches
E4 48      18 19         : Increment RAM & EPROM
E4 4A      C4 C4         : Waste time for settling data
E4 4C      F8 29 AE      : Load pulse time counter
E4 4F      7B            : Start programming pulse
E4 50      2E 8E         : Pulse width equals
E4 52      3A (50)         approx. 0.1 ms
E4 54      7A            : Stop programming pulse
E4 55      99 FB (D4)    : If it's not last
E4 58      3A (46)         byte get another byte
E4 5A      30 (38)       : Else do another loop
E4 5C      63 2B         : Deactivate EPROM
E4 5E      D5            : RETURN
```

LISTING 3—VERIFY BURN

```
Location   Code          Comment
E4 5F      F8 (C8) B8    : Point to start of
E4 62      F8 (D0) B9      RAM & EPROM
E4 65      F8 (00) A8 A9
E4 69      E8            : Set X R8
E4 6A      99 FB (D4)    : If all bytes O.K.
E4 6D      32 (76)         then exit
E4 6F      09 F3         : Compare RAM byte with
E4 71      19 18         : EPROM and increment both
E4 73      32 (69)         if the same continue
E4 75      7B            : Else set Q
E4 76      D5            : RETURN
```