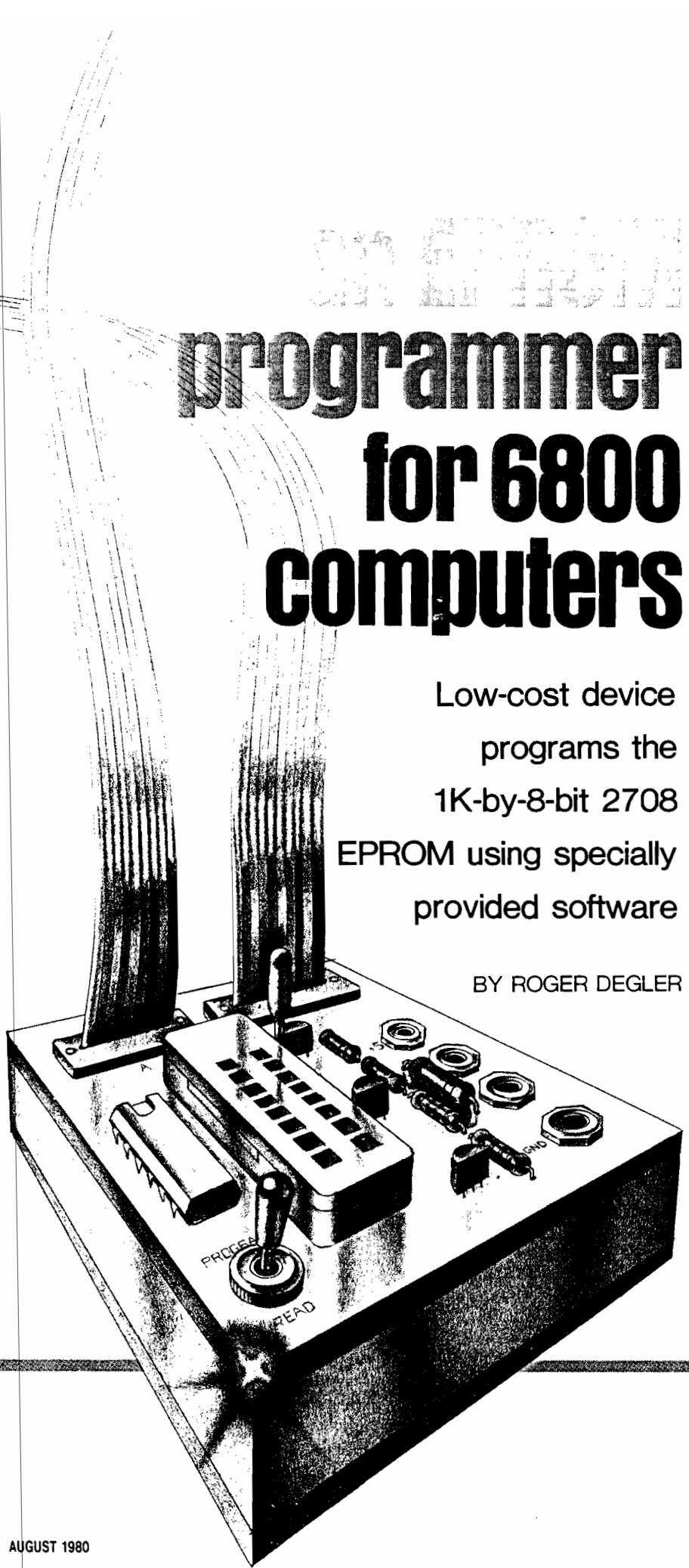
programmer for 6800 computers

Low-cost device programs the 1K-by-8-bit 2708 EPROM using specially provided software

BY ROGER DEGLER



ONE of the handiest devices available to the small-computer user is the EPROM (Erasable Programmable Read Only Memory). The basic ROM's advantage over RAM (Random Access Memory) is that its contents are retained when operating power is removed. This means that often-used programs such as bootstraps, monitors, operating systems, high-level languages, etc., can be permanently stored, thus eliminating time-consuming data retrieval from cassette or paper tape.

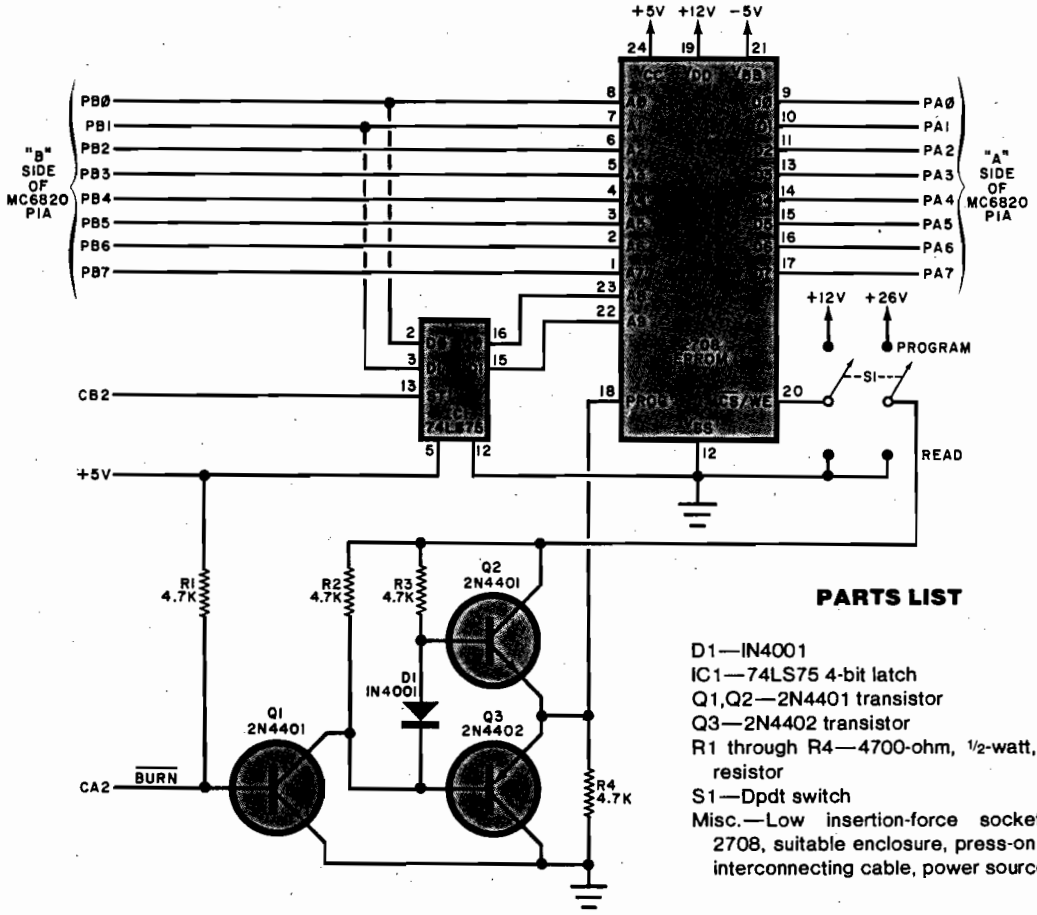
An EPROM is a special type of ROM that can be erased, and re-programmed as desired. Consequently, it makes an ideal storage medium for semipermanent data for experimental programs, or for software prototyping where changes may be made as work progresses.

The EPROM Programmer described here is an inexpensive device that can program the popular 1K by 8-bit, low-cost 2708 EPROM when used with a 6800-based computer having an MC6820 PIA (Peripheral Interface Adapter). The simple schematic on the next page illustrates this.

The computer, controlled by software to be described, supplies the EPROM Programmer with correct addresses, data, and properly timed programming pulses. In addition, the software ensures that the EPROM is blank prior to programming and verifies that the data the EPROM contains after programming is correct. The software also transfers the entire contents, or any portion of it, into RAM (where it can be altered) and writes it into blank EPROM or returns it to the same one. The software package is position-independent and occupies less than 256 bytes.

Circuit Operation. As shown in the schematic, lines PBØ through PB7 of the B side of the PIA supply the necessary addresses to the EPROM. Quad latch *IC1*, strobed by control line CB2, latches the two most significant address lines (A8 and A9) to the EPROM to form the required 10-bit address ($2^{10} = 1024$).

When switch *S1* is placed in the READ position, pin 20 (Chip Select) of the EPROM is grounded and power is removed from the transistor circuit driving the PROG input (pin 18). The computer can now examine the contents of the EPROM. In this mode, lines PAØ through PA7 of the A side of the PIA



PARTS LIST

D1—IN4001
IC1—74LS75 4-bit latch
Q1,Q2—2N4401 transistor
Q3—2N4402 transistor
R1 through R4—4700-ohm, ½-watt, 10% resistor
S1—Dpdt switch
Misc.—Low insertion-force socket for 2708, suitable enclosure, press-on type, interconnecting cable, power source.

The lines from the B side of the PIA supply the necessary addresses to the EPROM. With S1 on READ, the computer can examine the contents of the EPROM through the A side of the PIA.

allow the EPROM data to reach the computer.

In the PROGRAM setting of *S1* pin 20 (Write Enable) of the EPROM is connected to +12 volts to place the EPROM in its programming mode. Lines PA0 through PA7 now allow the computer to supply data to the EPROM. The other section of *S1* connects +26 volts to the three-transistor circuit that generates the programming pulse for pin 18. Transistors *Q2* and *Q3* form a complementary pair that actively pulls the programming input high or low as recommended by the 2708 data sheets. The when-to-burn signal comes in via CA2.

Software. This consists of four major sections; VBLNK, READ, BURN, VRFY and several subroutines. Although the order of these routines may appear strange, they are carefully arranged so that all internal branching may be in the relative mode. This allows the code to be executed at any address in the system memory without change.

Subroutine INIT initializes the PIA so that the A-lines are inputs, the B-lines are outputs with CA2 high and CB2 low. This subroutine also disables the IRQ level interrupts so that nothing will disturb the timing of the programming pulses, and because memory locations $A000 and $A001—used to store the IRQ interrupt vector by JBUG and MIKBUG—are used here to store PRMAD (starting EPROM address set by user). This routine is also called at the beginning of each of the four other major sections of the software.

Subroutine ADROUT is used to output the 10-bit address for the EPROM. The most significant two bits are output on the PIA B-side followed by raising and lowering CB2 to latch these two bits into *IC1*. The least-significant bits are then output, which together with the first two bits, make up the required 10-bit address. The EPROM address (PRMAT) is then incremented in the computer's RAM.

Subroutine INPUT calls subroutine ADROUT to output the current address to the EPROM and then loads accumulator-A with the contents of this location of the EPROM.

Subroutine VERI calls subroutine INPUT to get a byte of information from the EPROM. It then compares this byte with the contents of accumulator-B. If they are the same, it returns to the caller. If an error is detected, the index register is decremented so that it will point to the failing EPROM address. A software interrupt is then executed to return to the monitor. Accumulator-A will contain the bad data, and accumulator-B the expected data.

Subroutine WRITE is used to gener-

PROGRAM AVAILABILITY

The lengthy EPROM programmer instruction listing will be supplied on request. Send a stamped (15c in U.S.), self-addressed envelope to POPULAR ELECTRONICS, Dept. EPROM-1, One Park Ave., New York, NY 10016.

ate the correct width programming pulses for pin 18 of the EPROM. See the description of the BURN routine below for details of this pulse.

Program section VBLNK makes sure that the EPROM is erased before attempting to program it. Each byte in the EPROM is compared with FF(hex), which is the erased state. It is not necessary to start verifying with EPROM address ØØØØ. If a section of the EPROM already contains data, and it is desired to verify that the rest of the EPROM is blank so that additional data can be inserted, then the desired start of verification address is placed in memory locations PRMAD and PRMAD+1. Otherwise these two locations should be set equal to zero prior to starting execution at locations VBLNK.

The READ section is used to transfer the contents of the EPROM into the computer's RAM. Anything from 1 byte to the entire 1024 bytes may be transferred by setting up PRMAD, BEGAD and ENDAD prior to execution. Keep in mind that PRMAD is the location within the EPROM and has nothing to do with the memory address at which the EPROM will be located when installed in the system. PRMAD equal to zero implies the first location within the EPROM. Data is transferred from the EPROM starting at location PRMAD into RAM addresses BEGAD through ENDAD inclusive.

Section VRFY compares the data contained in RAM addresses BEGAD through ENDAD with the data stored in the EPROM starting at location PRMAD. If any discrepencies are found, a software interrupt (SWI) is issued to return to the monitor. With most 6800 monitors such as JBUG and MIKBUG, this will cause a display of the CPU registers. At this time, the index register will contain the location within the EPROM that contains the failing data, Accumulator-A will contain the failing data, and Accumulator-B will contain what the data should have been.

A BURN routine "burns" the data into the EPROM. Once the EPROM has been put in the PROGRAM mode via *S1*, the address and the data to be programmed into that address are applied to the EPROM. Then a +26-volt program pulse is applied to pin 18. Duration of this pulse must be at least 100 μs and not more than 1 ms. The next address and corresponding data are applied and another programming pulse is issued. This continues until all the addresses have been programmed. The entire process is repeated 256 times. This is done because the total program time applied to each address must be greater than, or equal to, 100 ms. As it is forbidden to apply more than one pulse at a time to any address, programming of the sequential addresses must be repeated many times.

The system that this is currently being used on is an MEK6800D2, a Motorola two-board microcomputer kit, that runs at a clock rate of 614.4 kHz. The number 36 (25 hex) loaded into the INDEX register in the WRITE subroutine is used to establish the width of the program pulse. If a clock rate of 1 MHz is used, this number should be changed to 61 (3D hex). This is based on eight machine cycles in the two-instruction loop at label WRT1, and an additional eight machine cycles outside this loop. If the duration of the program pulse is maintained at 500 μs and the program loop is repeated 256 times, this allows a total program time of 128 ms/location.

Data sheets for the 2708 specify that all 1024 locations should be programmed in each loop, but making the loop as small as 128 bytes has proved successful. Making the loop too small may be detrimental to the EPROM. Length of the program loop is established as ENDAD−BEGAD+1. Note that, at 614.4 kHz, the time required to program all 1024 bytes is only about two and a half minutes.

Construction. Since the circuit is not critical, the Programmer can be built using perf board with Wire-Wrap techniques, or a small pc board can be designed. To avoid pin damage, the use of a low-insertion force socket for the EPROM is suggested. Sockets for *IC1* and the three transistors are optional.

Suitable connectors must be made to contact the A and B parts of the MC6820 PIA and the CA2 and CB2 lines. Sources of +5, −5, +12 and +26 volts are also needed.

Any type of enclosure can be used as long as it will hold the power supplies with EPROM socket and *S1* on top.

Use. Switch *S1* should be in the READ position whenever an EPROM is inserted in or removed from the socket. In its unprogrammed state, the EPROM contains logic 1's at all storage locations. These 1's may be changed to 0's by programming—but 0's may be changed to 1's only by erasing the entire EPROM with exposure to 2537-Angstrom ultraviolet light.

Programming can, in certain cases, alter the contents of a previously programmed EPROM. For example, 27 (ØØ1Ø ØØ111) can be changed to 23 (ØØ1Ø ØØ11), but not vice versa. This is accomplished by using the READ routine to transfer the contents of the EPROM into RAM, locating the 27 in the RAM, changing it to 23, then using BURN to program this data back into the same EPROM.

To program in a small section of new data, for example only four bytes, use VBLNK to make sure that the four locations are in the unprogrammed state. Then with the EPROM removed from the socket, use READ to initialize a section of RAM to all 1's. This can be done because data read back from an empty socket is FF(hex).

Place the four data bytes in the RAM buffer, replace the EPROM and use *S1* at PROGRAM and BURN to program the data into the EPROM. Keep this section at least 128 bytes long. Since trying to program a 1 into the EPROM has no effect, only the four bytes of new data will make a change in the EPROM contents. After using BURN, set *S1* to READ and use VRFY subroutine to make sure that the EPROM was correctly programmed.

Once you gain some proficiency in programming, you can create "personality modules" for any function you desire, from instant bootstrap at turn-on, to full BASIC at the touch of a key. ◇