A Simple 2708 EPROM Programmer for the Motorola D2 Kit

Add 2K of firmware using a handful of components that fit right on the D2 board.

Frank W. Summers
Motorola Microsystems
514 E. Carmen St.
Tempe AZ 85283

Add up to 2K bytes of programmable memory to your Motorola MEK6800-D2 kit with this simple, low-cost 2708 EPROM programmer. Requiring only three transistors, six resistors, four capacitors and two switches, it can be assembled right on the D2 kit printed circuit board for approximately $7.

The reason for using the 2708-type EPROM is simple—its cost. The newer, larger, single supply EPROMs are too expensive for most home computer users. The 2708 type (with its +5 V, +12 V and −5 V requirement) is available to the hobby user for less than $10. The fact that the D2 kit has two EPROM sockets available for 2708s makes this a natural matchup. Just think, you can buy a D2 kit and add this programmer for less than the cost of many programmers alone!

Don't worry about investing in an expensive ultraviolet EPROM eraser, because most local computer stores offer this service for a small fee. If this isn't convenient, you can build an EPROM eraser for less than $20 using a germicidal lamp (G.E. #G8T5) and a suitable fixture. If you try this be sure to follow the precautions that come with the lamp about exposure to ultraviolet radiation.

Circuit Description

Only a brief description of the circuit is offered because it is simple. The +27 V for the programming pulse is supplied from three 9 V batteries connected in series. Since the current required from these batteries is only 20 mA and only during the actual programming of a device, the shelf life of the batteries will probably determine their usage. The power switch (S1) will remove the +5 V and +12 V from the EPROM socket and disconnect CB2 of PIA U21. When the EPROM socket is empty and S1 is off, there is nothing connected to the PIAs so they can be used for other applications.

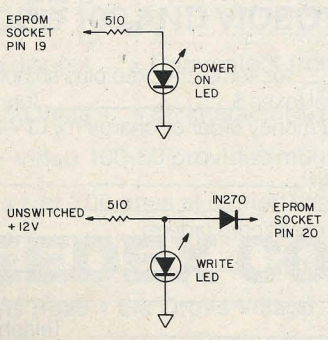

Fig. 1. Optional LEDs to indicate Power On and Write mode.

The read/write switch (S2) applies ground to pin 20 ($\overline{CS}$/WE) and removes the ground from the +27 V source when in the read position. When in the write position pin 20 ($\overline{CS}$/WE) is pulled up to +12 V through a 10k resistor (R1) and the ground is applied to the +27 V source. This allows programming pulses to be applied to pin 18 (PROG) through transistor Q1 as controlled by CB2 of the PIA (U21 of the D2 kit).

The two 100 Ohm resistors (R4 & R5) balance the base drive to Q2 and Q3. The 10 Ohm resistor (R6) and the .005 uF capacitor (C4) limit the rise-and-fall time of the +27 V programming pulse to the EPROM. All data to and from the EPROM is through the B side of the user PIA (U20 of the D2 kit). The addressing for the EPROM is from the A side of the user PIA (U20 of the D2 kit), plus CA2 and CB2 for address lines A8 and A9, respectively.

Programming

The software to operate the programmer requires 375 bytes of memory and supplies these four functions:

1. PROGRAM—programs the EPROM from the memory locations specified.
2. VERIFY—compares the EPROM with the memory locations specified.
3. DOWNLOAD—copies the EPROM into the RAM locations specified.
4. ERASED?—checks that the specified EPROM locations are erased.

The only inputs required for all of these operations are: (1) starting memory address, (2)

ending memory address and (3) EPROM address offset. Once these parameters are entered they will remain the same until changed by the user.

When any function is started these addresses are checked for proper range. The function is stopped and error message E0 is displayed if the difference between the ending and starting address is greater than 1024. If adding the EPROM offset to this difference exceeds 1024, error message E1 is displayed; error message E2 indicates that the EPROM address tried to exceed 1024 while the program function was executing.

Obviously this should never happen unless the programmer program itself bombs. Error message E3 indicates that the EPROM and specified memory locations didn't agree during the VERIFY function. Error message E4 indicates an unerased location was found in the specified EPROM locations during the ERASED function. The return of the JBUG prompt "-" to the display means the designated function has completed with no errors.

A programming pulse of one millisecond duration is applied for each EPROM address to be programmed. If less than 1024 locations are being programmed, enough delay is added to simulate programming the entire device. This routine is repeated 100 times to give each location a total of 100 milliseconds programming time. This limits the duty cycle of any one location as specified in most manufacturers' data books.

The programming time will be about two minutes whether programming 1 byte or 1024 bytes. This feature allows you to safely program an EPROM in short blocks, adding routines or entire programs as desired without erasing and reprogramming the entire device each time. Also, you can safely program the entire EPROM from a small RAM area by putting each successive block of code in RAM and programming the appropriate block of EPROM.

Another safety feature is that the D2 kit's keyboard is disabled while programming to avoid the possibility of the escape key being pressed and leaving the +27 V applied to the EPROM.

The delay routine is for a system clock frequency of 614.4 kHz. If your system has a different clock frequency, the number of counts in the delay loop will have to be adjusted accordingly. (Change location $00A9 to $A6 for 1 MHz.)

Test Procedure

So now you have the 2708 programmer circuit added to your D-2 kit and the software typed in. Before you do anything else you should save the program on tape. That done, you can use the following test procedure to verify the circuit and the program before plugging an EPROM into the socket. All you will need is a VOM, a watch with a second hand and a 2708 EPROM.

First check that the −5 V is on pin 21 of the EPROM socket. Then check for the +12 V on pin 19 and +5 V on pin 24 . . . make sure they are switchable with S1. Pin 20 should have +12 V when the read/write switch is on write and 0 V when in the read position. To check the +27 V on pin 18 the program will have to be temporarily changed. Change location $0035 from $8D to $3E. This is the WAI (wait for interrupt) instruction and will stop program execution at this point.

With the power switch on and the read/write switch set to write, the voltage on pin 18 should be 0 V. Referring to the operating instructions, start the PROGRAM mode with the first, last and EPROM offset addresses set to $0000. The voltage on pin 18 should now be between +25 V and +27 V. Switch back to read, and the voltage on pin 18 should drop to 0 V. Hit reset, restore location $0035 to $8D and turn the power switch off.

Now you will need some known data patterns to write to the EPROM socket for verification. Starting at location $0178 store $FF, $00, $55 and $AA. Set the first address to $0178 and the last address to $017B. You are now set up to simulate

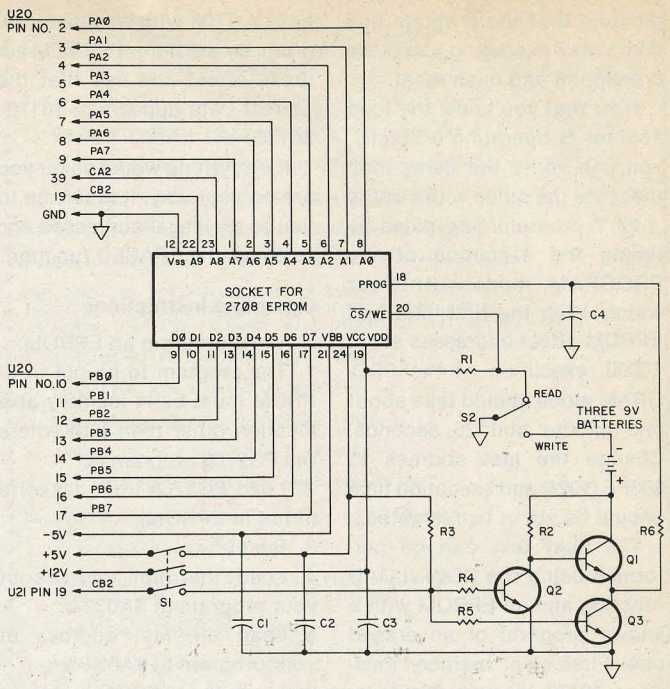

Fig. 2. Schematic of 2708 EPROM programmer for D2 kit.

programming these four bytes to the first four locations of the EPROM.

Set a breakpoint at $0030 and start the PROGRAM mode. All address lines of the EPROM socket should now read a logical 0 level (less than 0.5 V). All data lines should read a logical 1 level (more than 3.0 V). Continue program execution to the breakpoint again by typing E then G. Now address line A0 should read a 1 level; A1 through A9 should read a 0 level; and the data lines should all read a 0 level. Type E and G again and you should read a 1 level on A1, a 0 level on all other address pins, and the data lines should be alternate 1s and 0s (D0 = 1, D1 = 0, etc.).

Continue once more to the breakpoint and the results should be: A0 and A1 at a 1, all other address lines at 0, and the data lines alternate 0s and 1s (D0 = 0, D1 = 1, etc.). That's it for the data lines. Hit E, set the EPROM address offset to $0155 and start the PROGRAM mode again. The address lines at the EPROM socket should now be alternate 1s and 0s (A0 = 1, A1 = 0, etc.). Push E, set the EPROM address offset to $02AA and start the PROGRAM mode again. The address lines should now be alternate 0s and 1s (A0 = 0, A1 = 1, etc.). Clear the breakpoint and hit reset.

You can make a check of the loop counter by setting a breakpoint at $0068 and executing the PROGRAM mode. Now push the E key and the G key 100 times. This should cause the "-" to appear on the display in-

Q_1, Q_2, Q_3	2N4401 or equivalent
S_1	3PST toggle switch
S_2	SPDT toggle switch
C_1, C_2, C_3	.1 uF, ±20% ceramic capacitor
C_4	.005 mfd, ±20% ceramic capacitor
R_1, R_2, R_3	10k Ohm, ±10% 1/4 Watt carbon resistor
R_4, R_5	100 Ohm, ±10% 1/4 Watt carbon resistor
R_6	10 Ohm, ±10% 1/4 Watt carbon resistor
MISC.	24-pin socket for EPROM, three 9 V batteries, battery clips and battery holders.

Parts list.

dicating that the program has completed executing. Clear the breakpoint and push reset.

Now that you know the loop counter is operating correctly, you can verify the delay loop that sets the pulse width of the +27 V programming pulse by timing the execution of the PROGRAM mode with your watch. With the first, last and EPROM offset addresses set at $0000, execution of the PROGRAM mode should take about one minute and 25 seconds. Change the last address to $03FF (1024) and execution time should be about two minutes.

The next text can be performed using the DOWNLOAD function and an EPROM with a known program or an erased one. First clear memory locations $0177 through $017F by storing $00 to each location. Set the first address to $0179 and the last address to $017C. Plug the EPROM in, turn power on and execute the DOWNLOAD function.

When the "-" returns to the display, read locations $0177 through $017F. Locations $0179-$017C should contain the first four bytes of the program in the EPROM ($FF if erased). Locations $0177, $0178, $017D, $017E and $017F should still contain $00. Now run the VERIFY function without changing anything, and the "-" should return indicating that EPROM and RAM contents for those four locations match. If you have a 2708 with a known program, try a different EPROM address offset and see that the correct data appears in $0179-$017C.

If everything works so far you are on your way. It is left up to you to try illegal addresses and to check the ERASED function.

Operating Instructions

A. To program an EPROM:

1. The program to be put in EPROM must be in memory at a location other than that where the PG2708 program is.
2. Load PG2708 from cassette (if not in EPROM).
3. Reset.
4. Load the first address of your program at $A032-3.
5. Load the last address of your program at $A034-5.
6. Load the EPROM address offset $A036-7.
7. Make sure the EPROM power switch is off and the Read/Write switch is on read.
8. Plug the 2708 into the programmer socket.
9. Turn the EPROM power switch on.
10. Turn the Read/Write switch to write.
11. Type 0000G to execute the PROGRAM function.
12. When the "-" returns to the display, push Reset and then turn the Read/Write switch to read.
13. Type 0003G to VERIFY that the EPROM was programmed correctly.
14. Turn the EPROM power switch off and remove the 2708.

B. To download an EPROM to RAM:

1. Load PG2708 from cassette (if not in EPROM).
2. Reset.
3. Load the first RAM address at $A032-3.
4. Load the last RAM address at $A034-5.
5. Load the EPROM address offset at $A036-7.
6. Make sure the EPROM power switch is off and the Read/Write switch is on read.
7. Plug the 2708 into the programmer socket.
8. Turn the EPROM power switch on.
9. Type 0006G to execute the DOWNLOAD function.
10. Turn the EPROM power switch off and remove the 2708.

C. To verify that EPROM and memory agree:

1. Load PG2708 from cassette (if not in EPROM).
2. Reset.
3. Load the first memory address at $A032-3.
4. Load the last memory address at $A034-5.
5. Load the EPROM address offset at $A036-7.
6. Make sure the EPROM power switch is off and the Read/Write switch is on read.
7. Plug the 2708 into the programmer socket.
8. Turn the EPROM power switch on.
9. Type 0003G to execute the VERIFY function.
10. The "-" returns to the display if the EPROM and memory agree.
11. Turn the EPROM power switch off and remove the 2708.

D. To check that a designated section of the EPROM is erased:

1. Load PG2708 from cassette (if not in EPROM).
2. Reset.
3. Load a first* memory address at $A032-3.
4. Load a last* memory address at $A034-5.

*(These can be any block of memory—they are used only to determine how many EPROM locations to check.)

5. Load the EPROM address offset at $A036-7.
6. Make sure the EPROM power switch is off and the Read/Write switch is on read.
7. Plug the 2708 into the programmer socket.
8. Turn the EPROM power

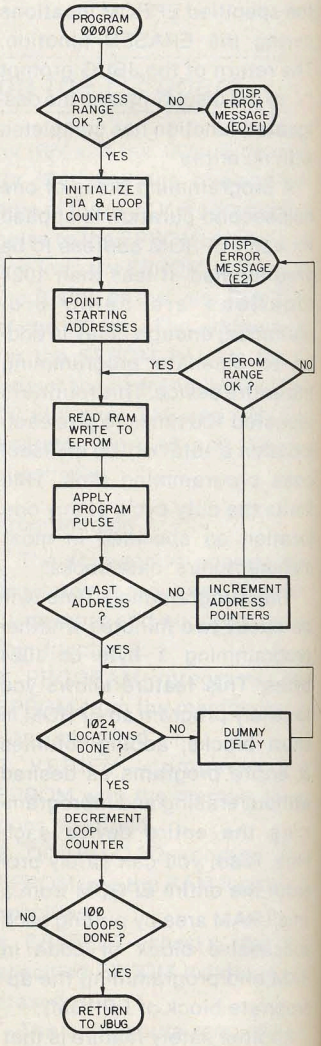

Fig. 3. PROGRAM mode flowchart.

Program listing.

```
00001                    TTL    MCM2708 EPROM PROGRAMMER FOR D-2 * REV. 2.2
00002A 0000              ORG    $0000
00003                    ******************************************************
00004                    *
00005                    ***** MCM2708 EPROM PROGRAMMER BY FRANK SUMMERS *****
00006                    *
00007                    ******************************************************
00008                    *
00009                    * NO FUNCTIONAL CHANGES THIS REVISION.
00010                    * CORRECTED LOCATION COUNTER INITIALIZATION.
00011                    *
00012                    ******************************************************
00013                    * THIS ROUTINE WILL PROGRAM THE MOTOROLA MCM2708,
00014                    * MCM68708, OR MOST OTHER 2708 TYPE EPROMS ON A
00015                    * MOTOROLA MEK6800D-2 SINGLE BOARD COMPUTER (D2 KIT)
00016                    *
00017                    * 375 BYTES OF MEMORY IS REQUIRED FOR THIS PROGRAM.
00018                    *
00019                    * THE SIMPLE EPROM PROGRAMMER CIRCUIT (ATTACHED)
00020                    * CAN BE ASSEMBLED ON THE D-2 KIT PCB OR CAN BE
00021                    * BUILT SEPARATELY AND CONNECTED WITH A CABLE.
00022                    *
00023                    * NOTE THAT THE 2708 REQUIRES +5V, +12V, & -5V.
00024                    *
00025                    * THE +27V PROGRAMMING PULSE IS SUPPLIED BY THREE
00026                    * 9V BATTERIES.
00027                    *
00028                    * 100 27V PULSES OF ONE MSEC DURATION ARE APPLIED
00029                    * FOR EACH EPROM ADDRESS TO BE PROGRAMMED.
00030                    *
00031                    * THE KEYBOARD IS DISABLED TO PREVENT ACCIDENTALLY
00032                    * HITTING "ESCAPE" & LEAVING THE +27V APPLIED.
```

switch on.

9. Type 0009G to execute the ERASED? function.

10. The "-" returns to the display if the designated EPROM locations are erased and ready for programming.

11. Turn the EPROM power switch off and remove the 2708.

Two or more of these functions are usually used together. For example, to copy an existing 2708 you will first DOWNLOAD it into RAM, VERIFY that it is stored correctly, check that the new 2708 is ERASED, PROGRAM the new 2708 and finally VERIFY that the program is in the new 2708. All of these operations can be performed after the address information is loaded only once. You should always use VERIFY immediately after a DOWNLOAD or a PROGRAM operation. Also, you should always check that an EPROM is ERASED before trying to PROGRAM it. (Even new EPROMs are not always erased!)

Error Messages

If E0 or E1 appears on the display (indicating out of range addresses), check the first, last and offset addresses at $A032–$A037. E0 means the last address minus the first address is

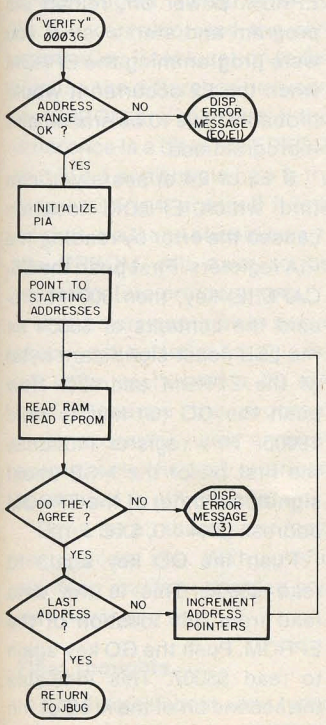

Fig. 4. VERIFY mode flowchart.

```
00033                    *
00034                    * IF LESS THAN 1024 LOCATIONS ARE BEING PROGRAMMED
00035                    * ENOUGH DELAY IS ADDED TO KEEP THE DUTY CYCLE
00036                    * LOW TO EACH LOCATION PROGRAMMED.
00037                    *
00038                    * THE FOUR MODES OF OPERATION ARE:
00039                    *
00040                    * 1. PROGRAM ($0000) PROGRAMS THE EPROM FROM RAM.
00041                    *
00042                    * 2. VERIFY ($0003) VERIFIES THAT EPROM & RAM AGREE.
00043                    *
00044                    * 3. DOWNLOAD ($0006) READS EPROM INTO RAM.
00045                    *
00046                    * 4. ERASED ($0009) VERIFIES THAT EPROM IS ERASED.
00047                    *
00049                    ********** OPERATING INSTRUCTIONS ***********
00050                    *
00051                    * CAUTION! THE READ/WRITE SWITCH SHOULD BE LEFT IN
00052                    * THE READ POSITION EXCEPT WHEN ACTUALLY PROGRAMMING
00053                    * AN EPROM. LEAVING IN WRITE WILL DISCHARGE BATTERIES.
00054                    *
00055                    * 1. LOAD STARTING RAM ADDRESS AT $A032-3.
00056                    *
00057                    * 2. LOAD LAST RAM ADDRESS AT $A034-5.
00058                    *
00059                    * 3. LOAD EPROM ADDRESS OFFSET AT $A036-7.
00060                    *
00061                    * 4. INSERT EPROM & TURN POWER SWITCH ON.
00062                    *
00063                    * 5. SET SWITCH TO WRITE FOR PROGRAMMING ONLY!
00064                    *
00065                    * 6. DOUBLE CHECK ADDRESSES BEFORE PROGRAMMING!
00066                    *
00067                    * 7. START DESIRED OPERATION.
00068                    *
00069                    * 8. THE JBUG PROMPT WILL RETURN WHEN FINISHED.
00070                    *    (PROGRAMMING TAKES 1.5 TO 2 MINUTES)
00071                    *
00072                    * 9. HIT RESET & SWITCH TO READ.
00073                    *
00074                    ************** ERROR MESSAGES ***************
00075                    *
00076                    * E0 = RAM ADDRESS RANGE GREATER THAN 1024.
00077                    *    (SELECTED FUNCTION NOT ATTEMPTED)
00078                    *
00079                    * E1 = RAM ADDRESS RANGE + EPROM ADDRESS
00080                    *    OFFSET GREATER THAN 1024.
00081                    *    (SELECTED FUNCTION NOT ATTEMPTED)
00082                    *
00083                    * E2 = EPROM ADDRESS OVERRANGE WHILE PROGRAMMING.
00084                    *  (SELECTED FUNCTION ABORTED WHEN OVERRANGE OCCURRED)
00085                    *
00086                    * E3 = EPROM AND RAM DID NOT AGREE DURING VERIFY.
00087                    *
00088                    * E4 = DESIGNATED EPROM LOCATIONS NOT ERASED.
00089                    *
00091                    *
00092                    ***************** EQUATES ********************
00093                    *
00094          8004  A PIADRA EQU    $8004    LSB OF EPROM ADDRESS
00095          8005  A PIACRA EQU    $8005    CA2 = EPROM ADDRESS LINE A8
00096          8006  A PIADRB EQU    $8006    DATA TO & FROM EPROM
00097          8007  A PIACRB EQU    $8007    CB2 = EPROM ADDRESS LINE A9
00098          8023  A PROG   EQU    $8023    CB2 = +27V PROGRAM PULSE CONTROL
00099          E08D  A JBUG   EQU    $E08D    JBUG REENTRY POINT
00100          E0FE  A OUTDS  EQU    $E0FE    JBUG DISPLAY ROUTINE
00101          A00C  A DISBUF EQU    $A00C    JBUG DISPLAY BUFFER
00102          A032  A STADR  EQU    $A032    START ADDRESS
00103          A034  A ENDADR EQU    $A034    END ADDRESS
00104          A036  A OFFSET EQU    $A036    EPROM ADDRESS OFFSET
00105          A038  A LOCCTR EQU    $A038    LOCATION COUNTER
00106          A03A  A LPCTR  EQU    $A03A    LOOP COUNTER
00107          A03B  A TEMPOS EQU    $A03B    TEMPORY STORAGE FOR OFFSET
00108                    *
00109                    ***************** JUMP TABLE *****************
00110                    *
00111A 0000 7E 000C  A         JMP    PROGM    PROGRAM MODE
00112A 0003 7E 0104  A         JMP    VERIFY   VERIFY MODE
00113A 0006 7E 0117  A         JMP    DNLOAD   DOWNLOAD MODE
00114A 0009 7E 0128  A         JMP    ERASED   ERASED? MODE
00115                    *
00116                    *
00117                    ************ PROGRAM ROUTINE *****************
00118                    *
00119                    *
00120A 000C BD 00E9  A PROGM  JSR    TEST     CHECK ADDRESS RANGES
00121A 000F 8D 38 0049         BSR    INIT     INITIALIZE PIA
00122A 0011 8E A078  A START  LDS    #$A078   RESTORE STACK POINTER
00123A 0014 F6 A036  A         LDAB   OFFSET   MSB EPROM ADDRESS
00124A 0017 F7 A03B  A         STAB   TEMPOS   SAVE IT
00125A 001A 8D 54 0070         BSR    FORMAT   FORMAT IT FOR PIA
00126A 001C B6 A037  A         LDAA   OFFSET+1 LSB EPROM ADDRESS
00127A 001F B7 8004  A         STAA   PIADRA
00128A 0022 CE 0400  A         LDX    #$400
00129A 0025 FF A038  A         STX    LOCCTR   SETS LOCATION COUNTER = 1024
00130A 0028 FE A032  A         LDX    STADR    POINT TO START ADDRESS
00131A 002B A6 00    A NEXT    LDAA   0,X      READ RAM BYTE
00132A 002D B7 8006  A         STAA   PIADRB   WRITE BYTE TO EPROM
00133A 0030 86 34    A         LDAA   #$34
00134A 0032 B7 8023  A         STAA   PROG     APPLY 27V PULSE, NMI DISABLED
00135A 0035 8D 71 00A8         BSR    DELAY    ONE MSEC DELAY
00136A 0037 86 3C    A         LDAA   #$3C
00137A 0039 B7 8023  A         STAA   PROG     REMOVE 27V PULSE, NMI DISABLED
00138A 003C BC A034  A         CPX    ENDADR   LAST ADDRESS?
00139A 003F 27 1D 005E         BEQ    TSTLOC   CHECK LOCATION COUNTER
00140A 0041 08                 INX             POINT TO NEXT ADDRESS
00141A 0042 8D 6A 00AE         BSR    INCROM   INC EPROM ADDRESS
00142A 0044 8D 0089  A         JSR    DECLOC   DEC LOCATION COUNTER
00143A 0047 20 E2 002B         BRA    NEXT     GO DO NEXT ADDRESS
00144                    *
00146                    *
00147                    ******** PROGRAM MODE SUBROUTINES ********
00148                    *
00149A 0049 7F 8005  A INIT   CLR    PIACRA
00150A 004C 7F 8007  A         CLR    PIACRB
00151A 004F CE FF34  A         LDX    #$FF34
00152A 0052 FF 8004  A         STX    PIADRA   A SIDE OUTPUTS, CA2=0
00153A 0055 FF 8006  A         STX    PIADRB   B SIDE OUTPUTS, CB2=0
00154A 0058 86 64    A         LDAA   #$64
00155A 005A B7 A03A  A         STAA   LPCTR    SETS LOOP COUNTER = 100
```

```
00156A 005D 39                  RTS
00157                   *
00158                   *
00159A 005E 7D A038  A TSTLOC TST     LOCCTR   TEST LOCATION COUNTER (MSB)
00160A 0061 26 65 00C8        BNE     DEC
00161A 0063 7D A039  A        TST     LOCCTR+1 TEST LOCATION COUNTER (LSB)
00162A 0066 26 60 00C8        BNE     DEC
00163A 0068 7A A03A  A DECLP  DEC     LPCTR    DEC LOOP COUNTER
00164A 006B 26 A4 0011        BNE     START    IF NOT LAST LOOP - START OVER
00165A 006D 7E E08D  A        JMP     JBUG     DONE - BACK TO JBUG
00166                   *
00167                   *
00168A 0070 F6 A03B  A FORMAT LDAB    TEMPOS   GET EPROM ADDRESS (MSB)
00169A 0073 27 0B 0080        BEQ     SET0
00170A 0075 5A                DECB
00171A 0076 27 11 0089        BEQ     SET1
00172A 0078 5A                DECB
00173A 0079 27 19 0094        BEQ     SET2
00174A 007B 5A                DECB
00175A 007C 27 21 009F        BEQ     SET3
00176A 007E 20 5A 00DA        BRA     E2       IF EPROM MSB GREATER THAN 3
00177A 0080 C6 34    A SET0   LDAB    #$34     SETS EPROM MSB = 0
00178A 0082 F7 8005  A        STAB    PIACRA
00179A 0085 F7 8007  A        STAB    PIACRB
00180A 0088 39                RTS
00181A 0089 C6 3C    A SET1   LDAB    #$3C     SETS EPROM MSB = 1
00182A 008B F7 8005  A        STAB    PIACRA
00183A 008E C6 34    A        LDAB    #$34
00184A 0090 F7 8007  A        STAB    PIACRB
00185A 0093 39                RTS
00186A 0094 C6 34    A SET2   LDAB    #$34     SETS EPROM MSB = 2
00187A 0096 F7 8005  A        STAB    PIACRA
00188A 0099 C6 3C    A        LDAB    #$3C
00189A 009B F7 8007  A        STAB    PIACRB
00190A 009E 39                RTS
00191A 009F C6 3C    A SET3   LDAB    #$3C     SETS EPROM MSB = 3
00192A 00A1 F7 8005  A        STAB    PIACRA
00193A 00A4 F7 8007  A        STAB    PIACRB
00194A 00A7 39                RTS
00195                   *
00196                   *
00197A 00A8 86 66    A DELAY  LDAA    #$66     SETS ONE MSEC DELAY
00198A 00AA 4A        DLY     DECA
00199A 00AB 26 FD 00AA        BNE     DLY
00200A 00AD 39                RTS
00201                   *
00203                   *
00204A 00AE 7C 8004  A INCROM INC     PIADRA   INC EPROM LSB
00205A 00B1 27 01 00B4        BEQ     INCMSB
00206A 00B3 39                RTS
00207A 00B4 7C A03B  A INCMSB INC     TEMPOS   INC EPROM MSB
00208A 00B7 20 B7 0070        BRA     FORMAT
00209                   *
00210                   *
00211A 00B9 7A A039  A DECLOC DEC     LOCCTR+1 DEC LOCATION COUNTER (LSB)
00212A 00BC 27 01 00BF        BEQ     DECMSB
00213A 00BE 39                RTS
00214A 00BF 7D A038  A DECMSB TST     LOCCTR
00215A 00C2 27 A4 0068        BEQ     DECLP
00216A 00C4 7A A038  A        DEC     LOCCTR   DEC LOCATION COUNTER (MSB)
00217A 00C7 39                RTS
00218                   *
00219                   *
00220A 00C8 8D EF 00B9 DEC    BSR     DECLOC
00221A 00CA 8D DC 00A8        BSR     DELAY    DUMMY DELAY IF < 1024 BYTES
00222A 00CC 20 90 005E        BRA     TSTLOC   LAST LOCATION DONE YET?
00223                   *
00224                   *
00225A 00CE 7F A011  A E0     CLR     DISBUF+5 SETS 6TH DIGIT OF DISBUF TO "0"
00226A 00D1 20 0E 00E1        BRA     ERROR
00227                   *
00228                   *
00229A 00D3 86 01    A E1     LDAA    #1
00230A 00D5 B7 A011  A        STAA    DISBUF+5 SETS 6TH DIGIT OF DISBUF TO "1"
00231A 00D8 20 07 00E1        BRA     ERROR
00232                   *
00233                   *
00234A 00DA 86 02    A E2     LDAA    #2
00235A 00DC B7 A011  A        STAA    DISBUF+5 SETS 6TH DIGIT OF DISBUF TO "2"
00236A 00DF 20 00 00E1        BRA     ERROR
00237                   *
00238                   *
00239A 00E1 86 0E    A ERROR  LDAA    #$E
00240A 00E3 B7 A010  A        STAA    DISBUF+4 SETS 5TH DIGIT OF DISBUF TO "E"
00241A 00E6 7E E0FE  A        JMP     OUTDS    EXIT & DISPLAY ERROR MESSAGE
00242                   *
00243                   *
00244A 00E9 B6 A035  A TEST   LDAA    ENDADR+1
00245A 00EC F6 A034  A        LDAB    ENDADR
00246A 00EF B0 A033  A        SUBA    STADR+1
00247A 00F2 F2 A032  A        SBCB    STADR
00248A 00F5 C1 04    A        CMPB    #4
00249A 00F7 24 D5 00CE        BCC     E0       OUT OF EPROM RANGE
00250A 00F9 BB A037  A        ADDA    OFFSET+1
00251A 00FC F9 A036  A        ADCB    OFFSET
00252A 00FF C1 04    A        CMPB    #4
00253A 0101 24 D0 00D3        BCC     E1       OUT OF EPROM RANGE
00254A 0103 39                RTS
00255                   *
00256                   *
00258                   *
00259                   *
00260                   ********** VERIFY ROUTINE **********
00261                   *
00262                   *
00263A 0104 8D 44 014A VERIFY BSR     SETUP
00264A 0106 A6 00    A VER    LDAA    0,X      READ RAM BYTE
00265A 0108 B1 8006  A        CMPA    PIADRB   COMPARE WITH EPROM BYTE
00266A 0108 26 2F 013C        BNE     E3       MEMORY & EPROM DON'T AGREE
00267A 010D BC A034  A        CPX     ENDADR   LAST ADDRESS?
00268A 0110 27 62 0174        BEQ     JBGJMP   RETURN TO JBUG
00269A 0112 08                INX              POINT TO NEXT ADDRESS
00270A 0113 8D 99 00AE        BSR     INCROM   INC EPROM ADDRESS
00271A 0115 20 EF 0106        BRA     VER      GO READ NEXT BYTE
00272                   *
00273                   *
00274                   ********** DOWNLOAD ROUTINE **********
00275                   *
00276                   *
```

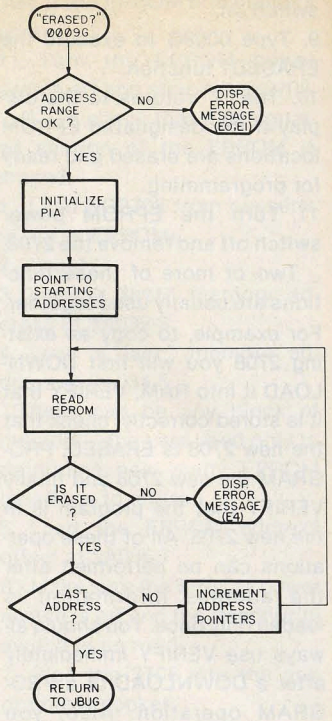

Fig. 5. ERASED? mode flowchart.

greater than 1024, and E1 means the last address minus the first address plus the EPROM address offset is greater than 1024.

If E2 appears on the display the EPROM address tried to exceed 1024 while the program was executing. You should hit reset, switch to READ, switch EPROM power off, reload the program and start over. If you were programming the EPROM when the E2 occurred it would probably have to be erased and reprogrammed.

If E3 or E4 appears you can find which EPROM location caused the error by reading the PIA registers. First push the ESCAPE (E) key, then 8004M. Record the contents of $8004 as the LSB (least significant byte) of the EPROM address. Now push the GO (G) key to read $8005. This register indicates the first bit of the MSB (most significant byte) of the EPROM address ($X4 = 0, $XC = 1).

Push the GO key again to read $8006. This is the data read from this location of the EPROM. Push the GO key again to read $8007. This indicates the second bit of the MSB of the EPROM address ($X4 = 0, $XC = 1). Use this bit with the one a

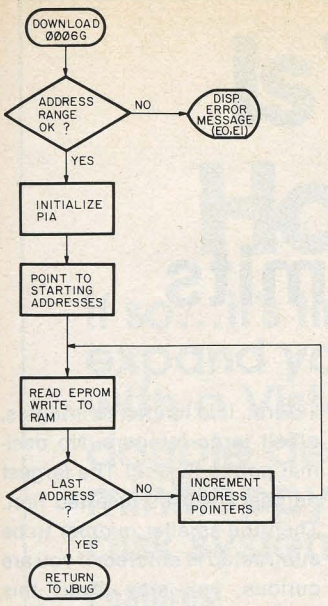

Fig. 6. DOWNLOAD mode flowchart.

```
00277A 0117 8D 31 014A DNLOAD BSR    SETUP
00278A 0119 B6 8006  A DNLD   LDAA   PIADRB   READ EPROM BYTE
00279A 011C A7 00    A        STAA   0,X      WRITE BYTE TO RAM
00280A 011E BC A034  A        CPX    ENDADR   LAST ADDRESS?
00281A 0121 27 51 0174        BEQ    JBGJMP   RETURN TO JBUG
00282A 0123 08                INX             POINT TO NEXT ADDRESS
00283A 0124 8D 88 00AE        BSR    INCROM   INC EPROM ADDRESS
00284A 0126 20 F1 0119        BRA    DNLD     GO DOWNLOAD NEXT BYTE
00285                  *
00286                  *
00287                  ********** ERASED? ROUTINE **********
00288                  *
00289                  *
00290A 0128 8D 20 014A ERASED BSR    SETUP
00291A 012A B6 8006  A ERAS   LDAA   PIADRB   READ EPROM BYTE
00292A 012D 81 FF    A        CMPA   #$FF     IS IT ERASED?
00293A 012F 26 12 0143        BNE    E4       EPROM NOT ERASED
00294A 0131 BC A034  A        CPX    ENDADR   LAST ADDRESS?
00295A 0134 27 3E 0174        BEQ    JBGJMP   RETURN TO JBUG
00296A 0136 08                INX             POINT TO NEXT ADDRESS
00297A 0137 BD 00AE  A        JSR    INCROM   INC EPROM ADDRESS
00298A 013A 20 EE 012A        BRA    ERAS     GO DO NEXT BYTE
00299                  *
00300                  *
00302                  *
00303                  ********** READ MODE SUBROUTINES **********
00304                  *
00305A 013C 86 03    A E3     LDAA   #3
00306A 013E B7 A011  A        STAA   DISBUF+5 SETS 6TH DIGIT OF DISBUF TO "3"
00307A 0141 20 9E 00E1        BRA    ERROR
00308                  *
00309                  *
00310A 0143 86 04    A E4     LDAA   #4
00311A 0145 B7 A011  A        STAA   DISBUF+5 SETS 6TH DIGIT OF DISBUF TO "4"
00312A 0148 20 97 00E1        BRA    ERROR
00313                  *
00314                  *
00315A 014A 8D 9D 00E9 SETUP  BSR    TEST     CHECK ADDRESS RANGES
00316A 014C 8D 13 0161        BSR    INIT1    INITIALIZE PIA
00317A 014E F6 A036  A        LDAB   OFFSET   MSB EPROM ADDRESS
00318A 0151 F7 A03B  A        STAB   TEMPOS   SAVE IT
00319A 0154 BD 0070  A        JSR    FORMAT   FORMAT IT FOR PIA
00320A 0157 B6 A037  A        LDAA   OFFSET+1 LSB EPROM ADDRESS
00321A 015A B7 8004  A        STAA   PIADRA
00322A 015D FE A032  A        LDX    STADR    POINT TO START ADDRESS
00323A 0160 39                RTS
00324                  *
00325                  *
00326A 0161 7F 8005  A INIT1  CLR    PIACRA
00327A 0164 7F 8007  A        CLR    PIACRB
00328A 0167 CE FF34  A        LDX    #$FF34
00329A 016A FF 8004  A        STX    PIADRA   A SIDE OUTPUTS, CA2=0
00330A 016D CE 0034  A        LDX    #$0034
00331A 0170 FF 8006  A        STX    PIADRB   B SIDE INPUTS, CB2=0
00332A 0173 39                RTS
00333                  *
00334                  *
00335A 0174 7E E08D  A JBGJMP JMP    JBUG     OPERATION DONE, RETURN TO JBUG
00336                  *
00337                  *
00338                         END
TOTAL ERRORS 00000


DEC    00C8  DECLOC 00B9  DECLP  0068  DECMSB 00BF  DELAY  00A8  DISBUF A00C
DLY    00AA  DNLD   0119  DNLOAD 0117  E0     00CE  E1     00D3  E2     00DA
E3     013C  E4     0143  ENDADR A034  ERAS   012A  ERASED 0128  ERROR  00E1
FORMAT 0070  INCMSB 00B4  INCROM 00AE  INIT   0049  INIT1  0161  JBGJMP 0174
JBUG   E08D  LOCCTR A038  LPCTR  A03A  NEXT   002B  OFFSET A036  OUTDS  E0FE
PIACRA 8005  PIACRB 8007  PIADRA 8004  PIADRB 8006  PROG   8023  PROGM  000C
SET0   0080  SET1   0089  SET2   0094  SET3   009F  SETUP  014A  STADR  A032
START  0011  TEMPOS A03B  TEST   00E9  TSTLOC 005E  VER    0106  VERIFY 0104
```

from $8005 to decode the MSB of the EPROM address. If both bits are 0 then the MSB = $00; if the first bit is 1 and the second is 0 then the MSB = 1; if the first bit is 0 and the second is 1 then the MSB = $02; and if both bits are 1 then the MSB = $03.

If the error is E4 you now know which location is not erased and the contents of that location (an erased location reads $FF), but about all you can do is to try erasing the EPROM again. If the error is E3 you can also read the corresponding RAM location to determine the difference between the EPROM and the RAM. If the difference is a bit in the EPROM that is a 1 and should be a 0, try programming it again. If it is a bit that is a 0 and should be a 1, the EPROM will have to be erased before programming.

If E3 errors continue to occur check your batteries for at least 8.5 volts each with a 20 mA load. Because the program stops execution when an error occurs, only the first error can be located in either case.

Don't let all these error messages scare you. With careful planning and keypunching you may never see one.

Final Thoughts

A few notes about what types of programs to put in EPROM and how to modify them if necessary to get you started: First, this program itself is an ideal candidate if you have limited RAM available because it will leave your entire RAM area free for the program that you are putting in EPROM. Other likely candidates to consider are: memory tests (so you can test all of your RAM), subroutines that you use frequently (saves RAM every time they are called), added functions that your monitor ROM doesn't perform and any other programs that you want ready to run immediately when your system is powered up.

Before putting a program in EPROM, you must first make sure it has no self-modifying code and that all jumps to absolute addresses within the program are changed to match what the addresses will be when the EPROM is plugged into its normal socket. No self-modifying code means all variables must be located in RAM somewhere else. In the D2 kit you can use the scratchpad RAM ($A000-$A07F) used by the monitor if you don't interfere with the monitor's variables. Generally $A032-$A05F can be used safely as is the case with this program, which uses $A032-$A03B.

There are eight jumps to absolute addresses within the program used with the 2708 EPROM programmer. Four of these are in the jump table at the beginning of the program. The others are located at lines 120, 142, 297 and 319. All of these must be changed to put this program in EPROM or to relocate anywhere else in RAM.

The 2708 EPROM programmer circuit was held to the minimum to keep the cost down. If you didn't mind spending a little more, it could be built into a separate box and connected to the D2 kit through connector J1. The +27 V could be supplied from a separate power supply if available. LEDs could be added to indicate when power was on and when the read/write switch was in the write position (see Fig. 1). If you anticipate heavy use, a more expensive zero-insertion-force type socket should be used for the EPROM. ■