John Laabs
RFD #1 Box 511
No. Stonington CT 06359

Photos by
Jim Viveiros
Misquamicut RI

Build a $20 EPROM Programmer

... for the 5204 4K chip

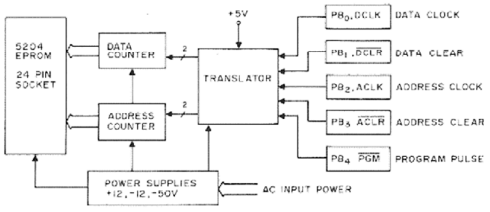


Fig. 1a. Programmer block diagram.

The availability of low-cost electrically programmable read only memory (EPROM) and the convenience of having a few favorite routines permanent as ROM in my KIM system provided the motivation to construct a 5204 EPROM programmer extension for KIM. The programmer design was to be as simple as possible, minimize I/O pin usage, and have its software completely relocatable to allow the routine to install itself in EPROM and be located in KIM unused memory areas without alteration. To reduce the programmer hardware requirement, the decision was made not to read the EPROM during programming as do most commercial programmers. The read EPROM modification to hardware and software could be added later should the need arise. Normally an EPROM will be programmed and then changed to a memory socket in KIM to be tested. Fig. 1a is a block diagram of the programmer and Fig. 1b details the KIM-1 port assignments for interfacing to the programmer.

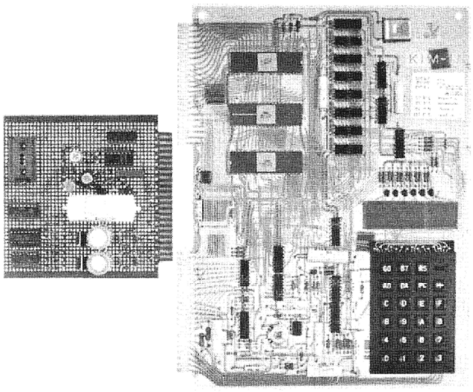
PBDD = ***11111 , * = don't care, PBDD = 1F
PB_0 = DCLK Clocks data on positive edge ↑ each transition increments the data.
PB_1 = $\overline{\text{DCLR}}$ Clears data to zero on low level. Normally high.
PB_2 = $\overline{\text{ACLK}}$ Clocks address on positive going edge ↑ each transition increments the address.
PB_3 = $\overline{\text{ACLR}}$ Clears address to zero on low level. Normally high.
PB_4 = $\overline{\text{PGM}}$ Provides program pulse to EPROM on low level. Normally high.

PB_7	PB_5	PB_4	PB_3	PB_2	PB_1	PB_0	
*	*	PGM	ACLR	ACLK	DCLR	DCLK	PORT PBD

Fig 1b. KIM-1 port assignments.

Why the 5204 EPROM?

The 5204 EPROM by



KIM-1 and programmer board - a neat combination.

National Semiconductor (MM5204) is a 4K electrically programmable (and erasable) read only memory. The device is a non-volatile memory organized 512 x 8 (½K bytes per chip). The 5204 requires +5 and -12 volt power supplies for ROM operation. Additional noteworthy features are its typical access time of 750 ns typical (1 μs maximum), power saver control for low power applications, and a three state data bus which permits easy memory expansion.

An alternative to the 5204 would be the readily available and inexpensive 1702A EPROM. The 1702A is organized 256 x 8 or ¼K bytes. Its access time is 1 μs maximum, identical to the 5204. The cost of a 1702A is about $7 as compared to $8 for the 5204. An immediate observation is the smaller cost per EPROM byte with the 5204. Naturally, if your requirements call for smaller memory blocks, such as a small system application, the 1702A may be a better choice. The 1702A is slightly more difficult to program due to the requirement to complement addressing during programming. This feature could be easily added to my basic programmer if you chose to modify it for the 1702A. The 1976 Intel Data Catalog is a good reference for the 1702A description and programming.

To access slow memory (access time less than 500 ns) with the 6502 a wait cycle is usually forced during memory read or write. The *MCS6500 Microcomputer Hardware Manual* by MOS Technology, Inc. describes a simple circuit to accomplish this wait cycle in section 2.3.4.1. By experimentation, however, I found that the combination of the 6502 having the address valid extended periods and the typical access time for the 5204 of 750 ns this delay circuit could be deleted. The 5204 EPROM has allowed me to expand KIM -1 with inexpensive ROM without affecting processor speed.

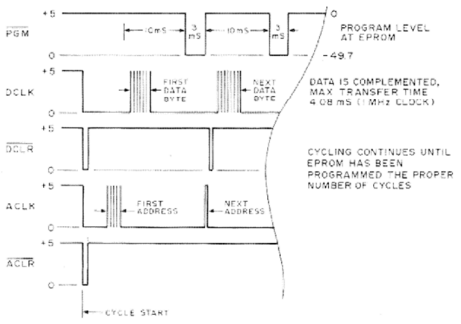


Fig. 2. Programmer timing diagram.

Programming the 5204 EPROM

Much of the literature I dug into discussed programming of the 1702A, 2708, and the 5204. The general concensus seems to be to not apply excessive voltages to prevent catastrophic failures, not to apply extremely fast waveforms to the address and data lines to prevent capacitive coupling internally from damaging the device, and not to apply excessively high duty cycle waveforms during programming to prevent high power dissipation in the device.

One good source of information is the *PROM User's Guide* published by Prolog Corporation. Specific information on the 5204 EPROM may be obtained from National Semiconductor's MOS IC Data Book. Although some conflicting information is in these publications, the general guide lines were followed and in turn, good results obtained. For example, the first National MOS book I used indicated the program pulse width should be between 2.5 ms and 5 ms with a duty cycle of less than 25%. With this data in hand I selected a 3 ms program pulse, an off time of 10 ms, and a 23% duty cycle (see Fig. 2). In a later MOS memory publication National specifies the minimum pulse width as .5 ms. I saw no real reason to alter the timing, however, you may want to save time by adjusting the pulse timers in the routine. All you need do is scale down both times by an equal factor. A word of caution is in order though; be aware that the off period will be affected with less than 4.08 ms (1 MHz clock) off time although the programmer will still operate.

Sockets	
1	24 pin dip
2	14 pin dip
3	16 pin dip
Miscellaneous	
1	4½" square, .1 inch center, Vero or Vector board
1	44 pin edge connector (use with Vero board)
60	component mounting pins
Discretes	
2	100 Ohm 1% ¼W
1	1k Ohm 10% ½W
4	15k Ohm 10% ¼W
1	10k Ohm 10% ¼W
2	2.7k Ohm 10% ¼W
1	5.6k Ohm 10% ½W
9	100k Ohm 5% ¼W
1	100 pF disk cap
2	1.0 uF 50 volt monolithic cap
1	10k Ohm trimpot
8	IN4002 rectifier
1	12 volt, 1 Watt Zener diode
1	2N2905A PNP transistor or eq.
1	2N5287 PNP transistor or eq. (with heat sink)
1	2N1893 NPN transistor or eq.
2	500 uF 35 volt electrolitic cap
1	100 uF 75 volt electrolitic cap
1	7812 +12 volt regulator
1	7912 -12 volt regulator
1	uA723 precision regulator
2	4040 CMOS counter
1	LM3900 Op amp
1	4050 CMOS buffer
2	25.2 VCT transformer (Stancor P8180)

Table 1. Parts list.

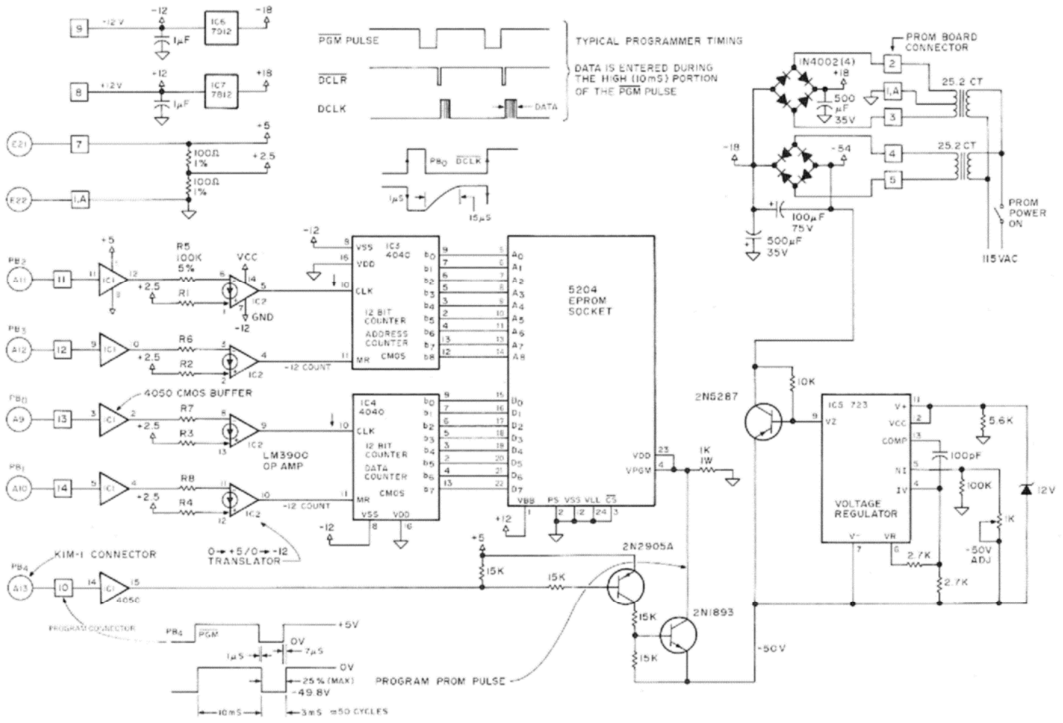


Fig. 3. Schematic diagram of 5204 EPROM Programmer.

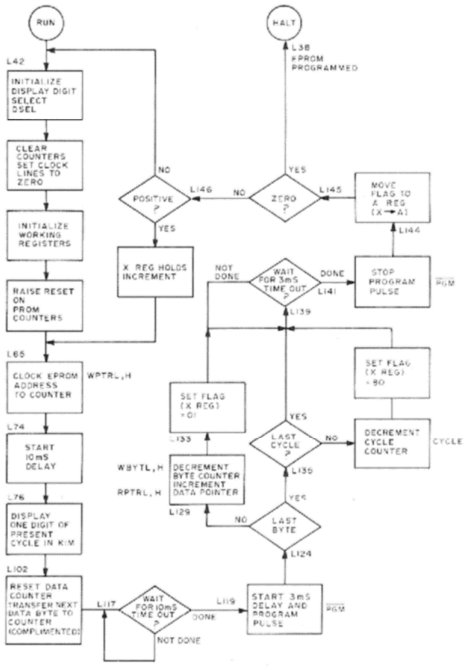


Fig. 4. EPROM Programmer flow diagram.

National suggests the 5204 be overprogrammed sequentially. Overprogramming consists of programming the EPROM in excess of the number of program cycles actually required to establish the correct data pattern. National states that each address location must be programmed for a minimum of 32 cycles with a pulse width of 5 ms or 64 cycles as the pulse width reaches 2.5 ms. I selected 50 cycles for a pulse width of 3 ms. For shorter program pulse widths National notes the EPROM should be programmed until the correct pattern is read back and then overprogrammed five times. This is symbolized by the notation x + 5x in the National literature. Since my programmer does not read this EPROM during programming the number of cycles are fixed by the routine or the operator at initialization.

The results of the technique used here have been satisfying and successful. To date I have 100% programmed a dozen EPROMS without a problem, except for one defective EPROM, which was replaced by the supplier at no charge.

The Programmer Circuit

The circuit shown in Fig. 3 contains three regulator circuits with rectifiers for providing + 12, - 12, and - 50 volts for the programmer. The 4040 CMOS counters provide the EPROM data and address registers. The LM3900 quad op amp translates the KIM 5 volt logic to -12 volt logic to control the counters. Two transistors level shift the program pulse to -50 volts. The third transistor is a series element in the uA723 -50 volt negative voltage regulator circuit. The 4050 CMOS buffer was used to convert the KIM TTL outputs to CMOS to gain the additional voltage swing pro-

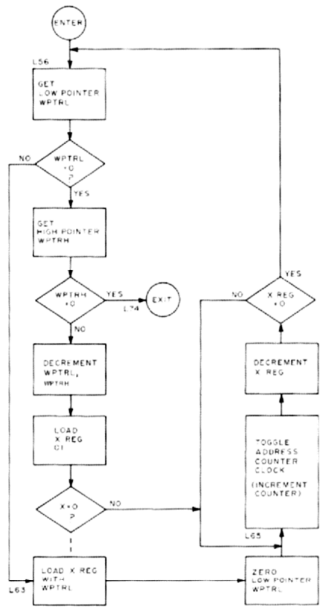


Fig. 5. Increment EPROM address routine flowchart.

vided with CMOS for increased reliability in the voltage translation process.

The address and data counters are operated identically. The counter is reset and then clocked until it contains the proper bit pattern for programming. The address is initialized and incremented to the cycle end. The data counter has the proper data pattern toggled in after being complemented by the programmer routine. The 5204 requires complemented data during programming.

Construction

My board is constructed on a 4½" x 4½" Vero board which was cut down from a standard 4½" x 6" 44 pin Vero proto board (#10290/FG W44/74). The sockets and components used were non-critical; however, the EPROM socket should be a type that allows the EPROM easy entry and removal. Most economical sockets will fill the requirement. A zero force type socket that is used on commercial programmers would be ideal and expensive. A standard Augat wire-wrap socket (Augat #324-AG2F) works fine. It all depends if you're going to program thousands of EPROMS. If you are, you should probably purchase one of the zero force type sockets.

The circuit board power supply rectifiers and regulators are shown in the photos. The transformers, which came from my scrap pile, are mounted adjacent to the card support. The discrete components are all parts which were readily available and are non-critical in nature. The only requirements on the transistors are that they have a breakdown voltage in excess of 50 volts and be capable of dissipating a couple of Watts. The -50 volt regulator series pass transistor should have a small heatsink. If one is not available, fabricate one out of a thin sheet of copper or aluminum.

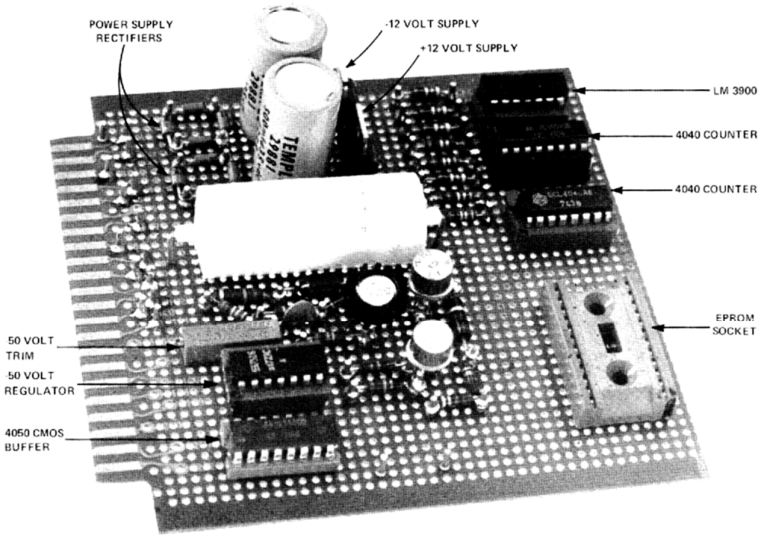
The complete parts complement should cost about $20 excluding EPROMS.

About the Program

The first usage of the program was to program an EPROM with itself to allow the EPROM routine to reside in KIM memory as ROM. I also felt that the program would serve more systems if it is independent of where it resides. You may, for instance want to relocate it from a cassette into different portions of RAM so as not to disturb the program you want to enter into EPROM.

The programmer routine runs in two sections (see Fig. 4 and 5 and Program A). The first section, beginning with EPROM start, does a little housekeeping and then halts allowing the insertion of the EPROM in the programmer socket. The routine tests the number of bytes to be programmed; if the count is correct, it enters the number of bytes to be programmed into a counter. If the count is incorrect (i.e., if the number of bytes to be programmed is less than zero or greater than ½K), the routine exits to display FFFF in the KIM monitor. Then the break vector is established, the number of cycles is set to 50, and the I/O port to the programmer is initialized. After this the programmer routine halts in the KIM monitor at the first instruction in the next section (run, L42) of the routine.

The halt allows power down of the programmer to permit the EPROM to be installed in the socket. Power



5204 EPROM Programmer board.

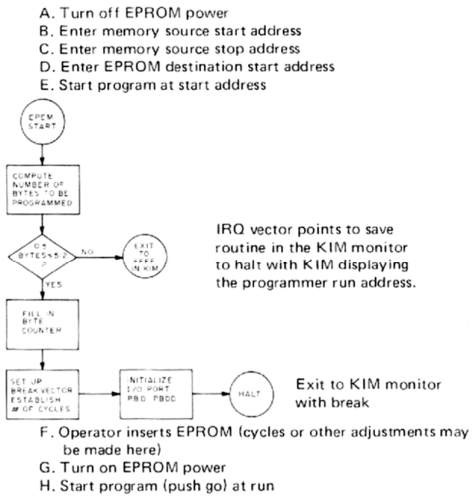


Fig. 6. Running the programmer.

is then restored and the EPROM may now be programmed. Running the routine, at the RUN location, programs the EPROM. While this program runs the present cycle is displayed in the KIM data digits, in hexadecimal, with the address digits dark. When finished the routine halts again at the RUN address. The programmer may now be powered down and the programmed EPROM removed.

The heart of the programmer routine, the RUN section, initializes the counters on the programmer for address and data entry. The address is entered to the programmer followed by the start of the off period of the programmer 10 ms pulse. During this delay one of the digits of the cycle is displayed and the data is transferred from the RAM byte specified to the EPROM data counter. The routine then waits for the end of the 10 ms period.

At the end of the 10 ms period, the programmer program pulse is started along with the on period time of 3 ms. During this period the routine establishes the next operation by setting pointers to the next byte of data or cycle through the setting of a flag in the x register. With the termination of the 3 ms period the programmer pulse is turned off, the flag is tested, and the next operation is performed. The operation consists of getting the next byte of data, starting a new cycle, and performing a halt when the EPROM is programmed.

An interesting note is the time consuming loop that clocks the data to the EPROM programmer data counter. Because this loop is completed during the off period of the programmer pulse the time period to program an EPROM is independent of this loop. This technique is an excellent example of how to use the KIM timer and the advantage of such a timer in a microcomputer system. The computer may do other tasks during waiting periods as long as you're back within the time period. With KIM the timer can easily be programmed to interrupt should uncertainties develop in the time required to complete the assigned tasks. Also, since the cycle digit is displayed during this 10 ms period, the digit stays on until the next 10 ms period is entered or the routine halts.

Operation

Operator loads via KIM keyboard the start read, end read, and start write address in page zero. The power to the programmer circuit should be turned off, EPROM removed. (Refer to Fig. 6.)

Operator starts the program at the start (first instruction). At this time the program computes the number of bytes to be programmed, sets up the $\overline{\text{IRQ}}$ vector for a program stop on break to KIM, establishes the number of programmer cycles, (which may be changed when the program halts) establishes the I/O configuration for the programmer, and then halts with the run address displayed by the KIM monitor.

The program will exit to FFFF in KIM if more than 512 bytes are selected to be written or if less than zero bytes were selected.

When the program halts, the number of cycles may be altered if desired.

The next step is to plug a 5204 EPROM into the programmer socket, turn on the programmer power, and then start the program (push go at the displayed address; run).

The cycles are counted down and displayed in the KIM display (in hexadecimal). The program halts back at the run address. Turn off the programmer, remove the programmed EPROM. ■

Program A. 5204 EPROM Programmer software routine (continued on page 76).

```
                                *Page Zero Registers
                                *Cycle and Display Counters
00DF                    DSEL    =$00DF  DISPLAY DIGIT SELECT CODE
00E0                    CYCLE   =$00E0  NUMBER OF PROGRAM CYCLES
                                *Instruction Data
00E1                    ERAL    =$00E1  END READ ADDRESS LOW
00E2                    ERAH    =$00E2  END READ ADDRESS HIGH
00E3                    SRAL    =$00E3  START READ ADDRESS LOW
00E4                    SRAH    =$00E4  START READ ADDRESS HIGH
00E5                    SWAL    =$00E5  START WRITE ADDRESS LOW
00E6                    SWAH    =$00E6  START WRITE ADDRESS HIGH
                                *Working Registers
00E7                    BYTL    =$00E7  NUMBER OF PROGRAM LOW BYTES
00E8                    BYTH    =$00E8  NUMBER OF PROGRAM HIGH BYTES
00E9                    RPTRL   =$00E9  NEXT BYTE POINTERS
00EA                    RPTRH   =$00EA  FOR DATA SOURCE
00EB                    WPTRL   =$00EB  WRITE POINTERS FOR EPROM
00EC                    WPTRH   =$00EC  ADDRESS START
00ED                    WBYTL   =$00ED  BYTE COUNT WORKING REGISTERS
00EE                    WBYTH   =$00EE
                        *Initialize routine: Establish data pointers, check
                        *for number of bytes 0 ≤ bytes ≤ 512 establish cycles,
                        *stop at program EPROM start address
0200  D8                EPGM    CLD             START PROGRAM
0201  38                        SEC             DETERMINE NUMBER
0202  A5  E1                    LDA   ERAL      OF BYTES
0204  E5  E3                    SBC   SRAL      TO
0206  85  E7                    STA   BYTL      BE
```

```
0208  A5  E2          LDA  ERAH     PROGRAMMED
020A  E5  E4          SBC  SRAH
020C  10  03          BPL  L26      TEST IF NEGATIVE
020E  4C  29 19  L25  JMP  FFFF     SOMETHINGS WRONG, EXIT
0211  C9  02     L26  CMP  02       TEST IF ⩾512
0213  10  F9          BPL  L25      EXIT IF POSITIVE (02 OR MORE)
0215  85  E8          STA  BYTH     ELSE, STORE
0217  18              CLC
0218  A9  00          LDA  00       SET UP
021A  8D  FE 17       STA  17FE     BREAK
021D  A9  1C          LDA  1C       VECTOR
021F  8D  FF 17       STA  17FF     TO POINT AT STOP ROUTINE
0222  A9  31          LDA  31       INITIALIZE NUMBER
0224  85  E0          STA  CYCLE    OF CYCLES TO 50
0226  A9  1F          LDA  1F       ESTABLISH OUTPUT
0228  8D  03 17       STA  PBDD     PORT AND
0223  A9  10  STOP    LDA  10       SET OUTPUT
022D  8D  02 17       STA  PBD      STATUS
0230  00              BRK           BREAK WITH (RUN)
0231  00                            NEXT ADDRESS IN KIM

      Program EPROM routine, initialize routine
      stops at run for EPROM setup in
      programmer circuit

0232  A9  00     RUN   LDA  00       INITIALIZE
0234  85  DF          STA  DSEL     DIGIT SELECT
0236  A9  F0          LDA  F0       CLEAR COUNTERS
0238  2D  02 17       AND  PBD      READY CLOCKS
023B  8D  02 17       STA  PBD
023E  A2  06          LDX  06       RESTORE
0240  B5  E2     L48  LDA  L4,X     WORKING
0242  95  E8          STA  L10,X    REGISTERS
0244  CA              DEX
0245  D0  F9          BNE  L48
0247  A9  0A          LDA  0A       RAISE RESETS
0249  0D  02 17       ORA  PBD      ON
024C  8D  02 17       STA  PBD      COUNTERS
024F  A0  00          LDY  00       ZERO Y (X = 0)
0251  A5  EB     L56  LDA  WPTRL    GET LOW POINTER FOR INCREMENT
0253  D0  0C          BNE  L63      IF NOT ZERO INCREMENT COUNTER
0255  A5  EC          LDA  WPTRH
0257  F0  21          BEQ  L74      EXIT IF ZERO
0259  C6  EB     L60  DEC  WPTRL    ELSE, INCREMENT
025B  C6  EC     L60A DEC  WPTRH    TO NEXT
025D  A2  01     L61  LDX  01       PAGE
025F  D0  04          BNE  L65
0261  A6  EB     L63  LDX  WPTRL
0263  84  EB          STY  WPTRL    ZERO WPTRL
0265  A9  FB     L65  LDA  FB       ENTER HERE FOR 1 STEP
0267  2D  02 17       AND  PBD      WITH X = 01
026A  8D  02 17       STA  PBD
026D  A9  04          LDA  04
026F  0D  02 17       ORA  PBD
0272  8D  02 17       STA  PBD
0275  CA              DEX
0276  D0  ED          BNE  L65
0278  F0  D7          BEQ  L56

      Start 10 ms delay, display cycle digit

027A  A9  9C     L74  LDA  9C       START 10ms DELAY
027C  8D  06 17       STA  ÷64      (1706)
027F  A9  7F          LDA  7F       SET UP SEGMENT PORT
0281  8D  41 17       STA  PADD1    FOR OUTPUT
0284  8E  40 17       STX  PAD1     X = 0 FROM ABOVE, SEGMENTS OUT
0287  8E  42 17       STX  PBD1     TURN OFF DIGIT
028A  A5  DF          LDA  DSEL     SELECT DIGIT TO BE DISPLAYED
028C  D0  13          BNE  L91      BRANCH TO DISPLAY HIGH DIGIT
028E  A9  12          LDA  DIG 1    LOAD 12  FOR LOW DIGIT
0290  8D  42 17       STA  PBD1     SELECT DIGIT
0293  A9  0F          LDA  0F
0295  25  E0          AND  CYCLE    GET LOW DATA
0297  E6  DF          INC  DSEL     SELECT NEXT DIGIT
0299  D0  13          BNE  L99      ALWAYS NOT EQUAL TO ZERO
029B  F0  8E     STOP 1 BEQ  STOP   INTERMEDIATE STOP BRANCH
029D  10  C6     L89  BPL  L65      INTERMEDIATE BRANCH
029F  30  91     RUN 1 BMI  RUN     INTERMEDIATE RUN BRANCH
02A1  A9  10     L91  LDA  DIG2     LOAD 10 FOR HIGH DIGIT
02A3  8D  42 17       STA  PBD1     SELECT DIGIT
02A6  A5  E0          LDA  CYCLE    GET HIGH
02A8  4A              LSR           DIGIT
02A9  4A              LSR           OF CYCLE DATA
02AA  4A              LSR
02AB  4A              LSR
02AC  C6  DF          DEC  DSEL
02AE  A8         L99  TAY
02AF  B9  E7 1F       LDA  ABS, Y   GET CONVERSION DATA
02B2  8D  40 17       STA  PADD     TURN ON SEGMENTS

      Get data byte to EPROM data counter,
      wait for end of 10 ms pulse,
      start 3 ms EPROM program pulse

02B5  A9  FD          LDA  FD       RESET DATA COUNTER
02B7  2D  02 17       AND  PBD
02BA  8D  02 17       STA  PBD
```

```
02BD  A9  02  17        LDA  02
02BF  0D  02  17        ORA  PBD
02C2  8D  02  17        STA  PBD
02C5  A9  FF            LDA  FF          GET COMPLEMENTED
02C7  A0  00            LDY  00          DATA
02C9  51  E9            EOR  (IND),Y     FOR COUNTER RPTRL
02CB  F0  0A            BEQ  L117        EXIT IF DATA IS ZERO
02CD  AA                TAX              ELSE INCREMENT COUNT
02CE  EE  02  17  L113  INC  PBD
02D1  CE  02  17        DEC  PBD         (4.08ms LOOP MAX WITH 1 MHz CLOCK)
02D4  CA                DEX
02D5  D0  F7            BNE  L113
02D7  AD  07  17  L117  LDA  1707        GET TIMER STATUS
02DA  F0  FB            BEQ  L117        IF NOT DONE, CHECK
02DC  A9  30            LDA  30          START 3ms TIME
02DE  8D  06  17        STA  ÷64         FOR PROGRAM PULSE
02E1  A9  EF            LDA  EF
02E3  2D  02  17        AND  PBD
02E6  8D  02  17        STA  PBD         START PULSE

        Test for last byte, last cycle, wait
        for 3 ms time-out. Stop program pulse

02E9  E4  ED            CPX  WBYTL
02EB  D0  06            BNE  L129
02ED  E4  EE            CPX  WBYTH
02EF  F0  0E            BEQ  L135        BOTH ZERO BRANCH TO TEST CYCLE
02F1  C6  EE            DEC  WBYTH
02F3  C6  ED      L129  DEC  WBYTL
02F5  E6  E9            INC  RPTRL
02F7  D0  02            BNE  L133
02F9  E6  EA            INC  RPTRH
02FB  A2  01      L133  LDX  01          SET FLAG AND WAIT
02FD  D0  08            BNE  L139        FOR TIME OUT, FLAG = 01, POS
02FF  E4  E0      L135  CPX  CYCLE
0301  F0  04            BEQ  L139        WAIT FOR TIME OUT FLAG = 00
0303  C6  E0            DEC  CYCLE
0305  A2  80            LDX  80
0307  AD  07  17  L139  LDA  1707        TEST FOR TIME OUT
030A  F0  FB            BEQ  L139
030C  A9  10            LDA  10          STOP PROGRAM
030E  0D  02  17        ORA  PBD         PULSE
0311  8D  02  17        STA  PBD
0314  8A                TXA              GET FLAG FOR TEST
0315  F0  84            BEQ  STOP1       IF ZERO, ALL DONE, EXIT AT START
0317  10  84            BPL  L89         BRANCH IF POSITIVE TO INC. ADDRESS
0319  30  84            BMI  RUN1        RUN NEXT CYCLE
```