P.R. Boldt

2716/2732 programmer

EPROM programmers are available in all shapes and sizes. The larger, more complicated machines tend to be rather expensive. However, a simple one can be constructed if the owner's microprocessor system is made to do the majority of the work, as we proved in the October 1981 issue of Elektor. The EPROM programmer described here is somewhat 'middle of the road' in that it has been designed specifically for use with the Elektor SC/MP and Junior Computer systems. It is very compact (all the components can be mounted on a single 'Eurocard' board) and it can be used to program 2732 devices as well as the popular 2716s. Also, it is possible to verify that the programmed data is correct.

There are a number of differences between the programmer published in the October 1981 issue and the circuit featured here. The earlier 'plug-in' programmer can only be used to program 2716 devices and the microprocessor system used has to have a 'hold' facility. This latter means that it can not be used with the Junior Computer. The latest circuit can be used to program both 2716 and 2732 devices and has been designed specifically for use with the SC/MP and Junior Computer systems, although it can probably be used in other systems as well. A standard connector can be mounted on the printed circuit board for the new programmer which will mate directly with the SC/MP data bus or the Junior Computer expansion connector. These reasons seem sufficient grounds for publishing the new design.

A detailed description of how the 2716 EPROM can be programmed was given in the previous article (October 1981, page 10-14), therefore only a brief description is required here. Just to refresh your memory, a programming voltage of 25 V has to be connected to the V_{pp} input of the device in question. As far as the 2716 is concerned, this is pin 21, but in the case of the 2732, it is pin 20. Also, a programming pulse with a duration of at least 50 ms has to be applied to the $\overline{CE}$ input (pin 18) of the EPROM so that the information present on the data lines can be stored in the corresponding address location.

If you have a computer which has a 'hold' facility at your disposal, then the 'plug-in EPROM programmer' will be quite sufficient. However, it can not be used with the Junior Computer, as there is no facility for holding the address and data lines stable for the 50 ms period required for programming. This means that for the idea to be implemented on the Junior Computer, the information presented to the address and data lines has to be 'latched' until programming is completed. Although these latches are not required by the SC/MP, they are included on the printed circuit board to make the unit more 'versatile'. The board can be 'programmed' for use with either the SC/MP or the 6502 (Junior Computer) by means of wire links. The circuit is designed in such a way that all the signals required during the programming are generated by means of hardware.

The circuit diagram

Before having a closer look at the programming, it is a good idea to examine the intricacies of the circuit itself (see figure 1). As stated previously, the address and data information has to be temporarily stored in latches. This is accomplished by means of IC1 . . . IC4. The 2716 requires 11 address lines, whereas the 2732 requires 12. The twelfth address line (A11) is connected to the 2732 via switch S1 (the 2716/2732 selector switch).

The data lines, DØ . . . D7, are fed to two latches connected in parallel (IC1 and IC2). The inputs of IC1 are connected to the outputs of IC2 and vice versa. Consequently, it is possible for the computer to read the data contained in the EPROM being programmed. As can be seen from the left-hand side of the diagram, a few other connections to the computer are also required.

Furthermore, some form of address decoding has to be provided, so that the EPROM can be programmed from any area of computer memory. 'Page addressing' is accomplished by means of a four bit comparator, IC5. The 'A' inputs of this device are connected directly to the high order address lines A12 . . . A15, whereas the 'B' inputs are connected to switches S3 . . . S6. An 'open' switch produces a high logic level, therefore S3 corresponds to address line A12, S4 to A13, S5 to A14 and S6 to A15. A so called 'page' always consists of 4 kilobytes. This may seem quite a lot, but it is essential as the circuit has to be able to program and read (verify) a 4k EPROM (2732). This means that when a 2716 is programmed it can be accessed in two address ranges — xØØØ . . . x7FF and x8ØØ . . . xFFF, where x is any hexadecimal value (Ø . . . F) depending on the positions of S3 . . . S6.

The timing for the programming pulses is derived from IC6, which is a CMOS version of the well known 555 timer and which is connected as an astable multivibrator. The values of capacitor C1 and resistors R5 . . . R7 determine the pulse duration of the multivibrator. With the values shown, this pulse duration is 10 ms. The output of the multivibrator is fed to the input of an 8 bit shift register, IC7, via an inverter, N5. The reset input of IC6 (pin 4) is fed from the Q output of flipflop FF2. The clock inputs of flipflops FF1 and FF2 are provided with clock pulses via inverter N6, which is connected to pin 31a of the 'bus' connector. Consequently, both flipflops receive a clock pulse each time the processor outputs a write signal.

The 'A = B' output of IC5 will become high as soon as the preset page address is recognised. If the flipflops receive a write pulse from the processor at this time, the Q outputs of both FF1 and FF2 will go high simultaneously. FF2 removes the reset from the multivibrator (IC6), thereby starting the timing se-

1

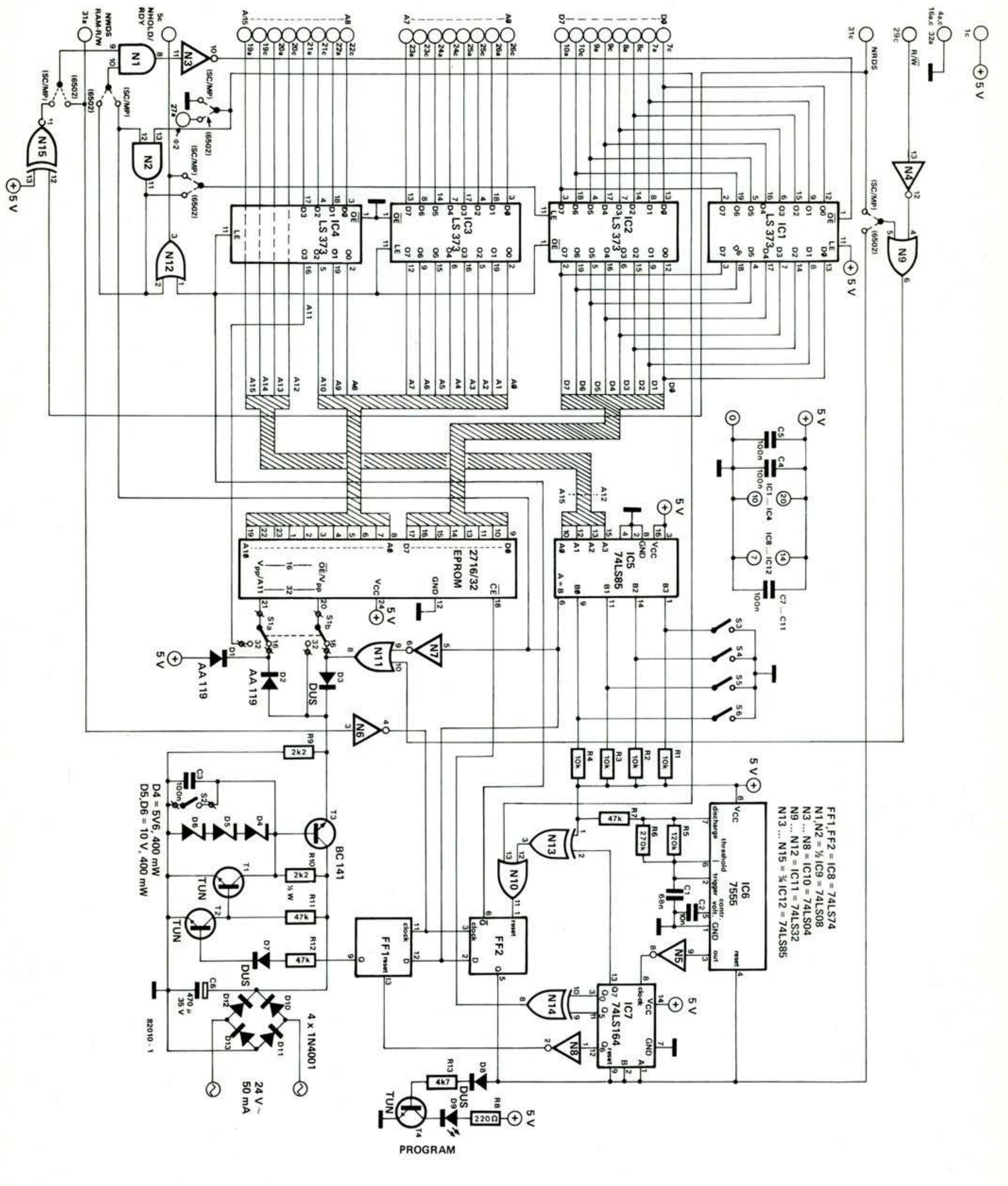

Figure 1. The complete circuit diagram of the 2716/2732 EPROM programmer. The EPROM to be programmed is shown at the centre of the diagram with the latches to the left and the 25 V supply and timing circuitry to the right.

2

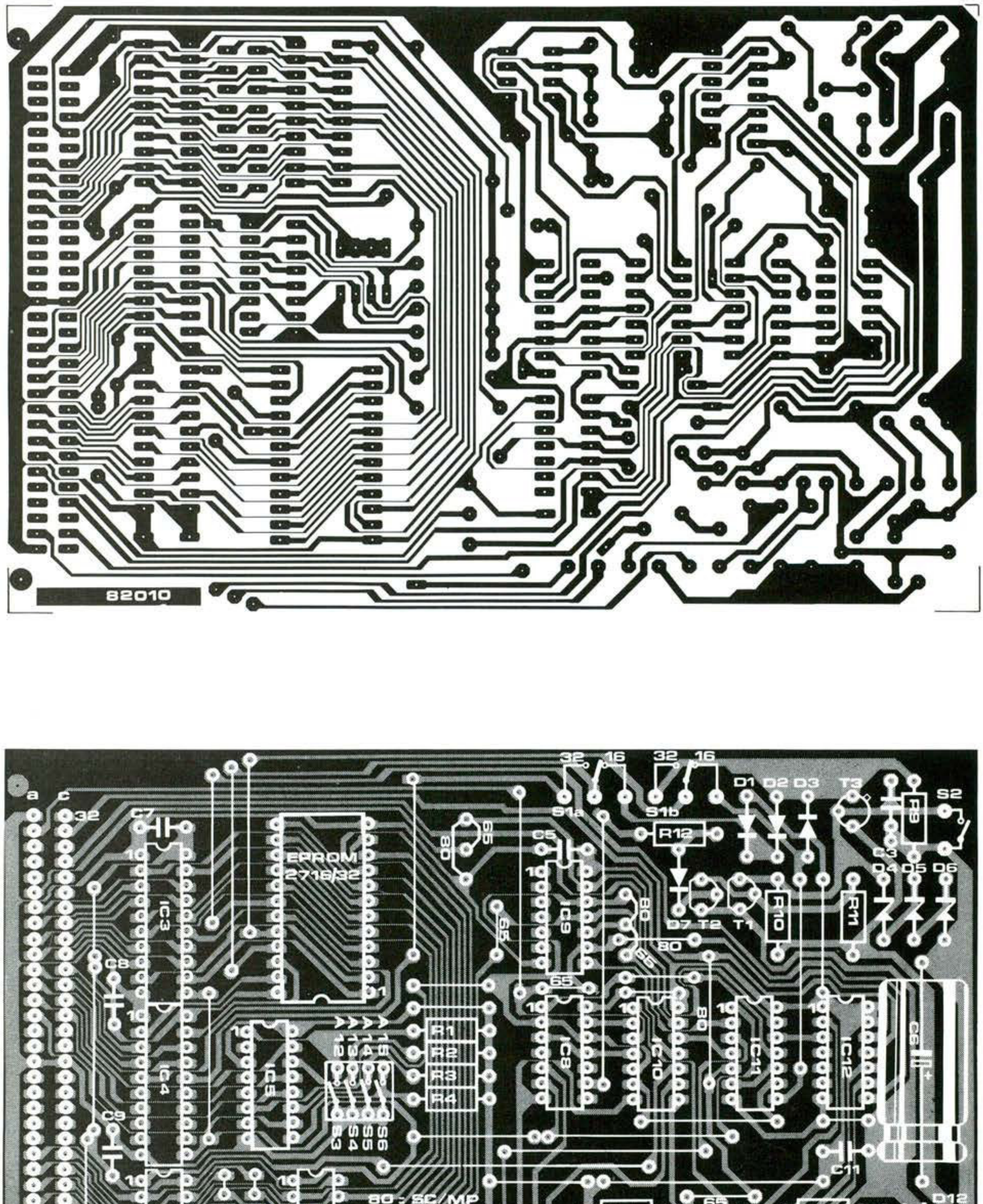

Figure 2. The printed circuit board and component overlay for the programmer.

Parts List

Resistors:
R1 . . . R4 = 10 k
R5 = 120 k
R6 = 270 k
R7,R11,R12 = 47 k
R8 = 220 Ω
R9 = 2k2
R10 = 2k2/½ W
R13 = 4k7

Capacitors:
C1 = 68 n
C2 = 10 n
C3 . . . C5,C7 . . . C11 = 100 n
C6 = 470 μ/35 V

Semiconductors:
D1,D2 = AA 119
D3,D7,D8 = DUS
D4 = 5V6/400 mW zener diode
D5,D6 = 10 V/400 mW zener diode
D9 = LED
D10 . . . D13 = 1N4001
T1,T2,T4 = TUN
T3 = BC 141
IC1 . . . IC4 = 74LS373
IC5 = 74LS85
IC6 = 7555
IC7 = 74LS164
IC8 = 74LS74
IC9 = 74LS08
IC10 = 74LS04
IC11 = 74LS32
IC12 = 74LS86

Miscellaneous:
S1 = DPST
S2 = SPST
S3 . . . S6 = 4 miniature DIL switches
24 pin zero insertion socket
64 way right-angle male connector, DIN 41612

quence. FF1, on the other hand, enables the small power supply constructed around transistors T1 . . . T3 to provide the required 25 V for the programming input of the EPROM. The 25 V power supply can be switched on and off via FF1 or, alternatively, it can be disabled by means of switch S2 so that the EPROM can only be 'read'.

Now to continue the story; the programming voltage is switched on and the timer has been started. The trailing edge of the pulses supplied by the timer enter a succession of 'ones' into the shift register, IC7. This means that the Q_0 output of the shift register becomes high after 10 ms, and so does the $\overline{CE}$ input of the EPROM. The $\overline{CE}$ input remains high for a duration of 50 ms, due to the fact that the Q_0 and Q_5 outputs of the shift register are gated together by means of the EXNOR gate N14. After a further delay of 10 ms, output Q_6 will go high, which resets flipflop FF1 and turns off the programming voltage. During this time the latches are enabled via the $\overline{Q}$ output of FF2 until the Q_7 output of IC7 goes high 10 ms later. The Q output of FF2 is also connected to transistor T4 via D8 and R13. This transistor turns on LED D9 all the time that the EPROM is being programmed.

The section of the circuit just described has nothing to do when the contents of the EPROM are being read — as the processor does not provide a write signal at this time. During the read process the outputs of IC2 are set in the 'tri-state' mode and IC1 is enabled, which means that the information contained in the EPROM is (almost) connected directly to the data lines of the processor.

Switch S1 is used to select between the two possible types of EPROM that can be programmed. If S1 is in the position indicated in the diagram (2716), the programming voltage is connected to pin 21 via diode D2 and the $\overline{OE}$ input is connected to the output of gate N11. With the switch in the other position (2732), address line A11 is connected to pin 21 and the programming voltage is applied to pin 20. Various control signals for the programmer are derived from those of the microprocessor via gates N1 . . . N4, N9, N10, N12 and N15.

The printed circuit board

The printed circuit board and component overlay for the EPROM programmer is given in figure 2. Mounting the components onto the Eurocard sized board should not cause any problems, especially for readers who have gained construction experience with the SC/MP or Junior Computer systems. A word of advice: it is important to select a good quality socket for the EPROM as this has to be used repeatedly. Ideally, a 'zero insertion force' type should be used.

Depending on whether the programmer board is to be used with the SC/MP system or with the Junior Computer, certain wire links will have to be installed. These are clearly marked on the component side of the board. All other connections are made via the standard 'bus' connector. The only component not mounted on the board is the 24 V transformer which is required for the programming voltage supply.

Using the programmer with the SC/MP

It is a very simple matter to use the EPROM programmer with the Elektor SC/MP system. The first requirement is to close switch S2. When the desired address range has been set by means of switches S3 . . . S6, an 'empty' EPROM is mounted in the socket and the board installed into the system. After the 24 V a.c. voltage has been applied, switch S2 is opened and the power supply to the computer turned on. Now the EPROM can be programmed. It is wise to close S2 as soon as programming has been completed.

The SC/MP system does not require any extra software for programming EPROMs. ELBUG is quite sufficient. Memory locations can be programmed by using the 'modify' key followed by the address of the location to be programmed (Mo . . . YYYY). An unprogrammed EPROM location will indicate 'FF' on the display. If, for example, the value Ø8 is to be stored at the chosen location, all that has to be done is to enter Ø8 and the data will be stored in the specified address location.

If larger data blocks are to be programmed, it is recommended to store them somewhere in RAM first. They can then be transferred to EPROM via the block transfer routine (BL . . . SSSS, EEEE, BBBB; where S = start address, E = end address and B = initial address of where the data block is to be located). The EPROM can be read in exactly the same manner as for a 'normal' memory location.

Using the programmer with the Junior Computer

Before the Junior Computer can be used to program EPROMs, a short 'piece' of software is required. A suitable program is given in table 1. This is stored in locations Ø2ØØ . . . Ø277. The EPROM programmer card has to be connected to the expansion connector of the Junior Computer before the 24 V a.c. voltage is applied. Finally, the Junior Computer has to be switched on and the program in table 1 entered into memory. The programmer board will then be ready for use.

It is, of course, now possible to store the program in table 1 in EPROM, so that it does not have to be entered each time it is required. This means that the absolute jump instructions in the program have to be altered so that the routines will work correctly in a different memory area. It is possible to 'store' the program in the TM EPROM situated on the extension board. This still has a number of empty locations, ØC8Ø . . . ØCFF, which can be used for exactly this purpose (in the EPROM itself these are locations 48Ø . . . 7FF). If the TM EPROM is to be used to store the 'program' program the start address should be ØC8Ø instead of Ø2ØØ (see table 1) and the absolute jump instructions (all three byte instructions ending with the value Ø2) should be modified accordingly. We will now give a brief description of the three program sections: Program, Duplicate and Verify.

The program routine

The low and high order bytes of the EPROM start address are first stored in memory locations MOVL and MOVH (ØØØ4 and ØØØ5), respectively. Then the program routine is started from address Ø2ØØ. Now the address and the data contained therein will appear on the Junior Computer display.

If, for example, the value A9 is to be programmed into the specified location, the key 'A' is depressed (the display remains unchanged) followed by key 9. The display will then go blank for a

```
0200                ORG    $0200

                    DATE :  10-7-'81


                    PAGE ZERO DATA BUFFERS :

0200                SAL     *     $0000  DATA BLOCK START ADDRESS
0200                SAH     *     $0001
0200                EAL     *     $0002  DATA BLOCK END ADDRESS + 1
0200                EAH     *     $0003
0200                MOVL    *     $0004  EPROM PROGRAM START ADDRESS
0200                MOVH    *     $0005

0200                INH     *     $00F9  DISPLAY BUFFER  ( DATA )
0200                POINTL  *     $00FA     "        "    ( ADDRESS L )
0200                POINTH  *     $00FB     "        "    ( ADDRESS H )

                    EXTERNAL SUBROUTINES :

0200                GETBYT  *     $1D6F
0200                SCAND   *     $1D88

                    JUNIOR MONITOR START :

0200                RESET   *     $1C1D


                    PROGRAM START ADDRESS     : $0200  ( PROG )
                    DUPLICATE START ADDRESS   : $0222  ( DUPL )
                    VERIFY START ADDRESS      : $0233  ( VERIFY )


                    ****************
                    EPROM-PROGRAMMER
                    ****************

0200 20 55 02  PROG   JSR    TRF     TRANSFER MOVL(H) TO POINTL(H)
0203 A0 00     PRGR   LDYIM  $00     CLEAR Y-REGISTER
0205 B1 FA            LDAIY  POINTL  GET DATA SPECIFIED BY POINTL(H) AND
0207 85 F9            STAZ   INH     STORE THIS IN DISPLAY BUFFER INH
0209 20 6F 1D         JSR    GETBYT  READ TWO HEXKEYS AND STORE THEIR VALUE IN THE
                                     ACCUMULATOR. RETURN WITH N=1 IF
                                     ONLY HEXKEYS WHERE DEPRESSED.
                                     IF A COMMAND KEY WAS DEPRESSED,
                                     RETURN WITH N=0
020C 10 07            BPL    PR      COMMAND KEY DEPRESSED?
020E A0 00            LDYIM  $00     CLEAR Y-REGISTER
0210 91 FA            STAIY  POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
                                     THE EPROM MEMORY LOCATION
                                     SPECIFIED BY POINTL(H)
0212 4C 03 02         JMP    PRGR
0215 C9 12     PR     CMPIM  $12
0217 D0 35            BNE    PRGR    +KEY?
0219 E6 FA            INCZ   POINTL  INCREMENT ADDRESS BY ONE
021B D0 E6            BNE    PRGR
021D E6 FB            INCZ   POINTH
021F 4C 03 02         JMP    PRGR
0222 20 55 02  DUPL   JSR    TRF     TRANSFER MOVL(H) TO POINTL(H)
0225 A0 00     DU     LDYIM  $00
0227 B1 00            LDAIY  SAL     GET DATA SPECIFIED BY SAL(H)
0229 91 FA            STAIY  POINTL  PROGRAM THE CONTENTS OF THE ACCUMULATOR IN
                                     THE EPROM MEMORY LOCATION
                                     SPECIFIED BY POINTL(H)
022B 20 5E 02         JSR    INCMNT  INCREMENT SAL(H) AND POINTL(H) BY ONE
022E D0 F5            BNE    DU      NOT LAST ADDRESS
0230 4C 1D 1C         JMP    RESET   RETURN TO JUNIOR MONITOR

0233 20 55 02  VERIFY JSR    TRF     TRANSFER MOVL(H) TO POINTL(H)
0236 A0 00     VER    LDYIM  $00
0238 B1 FA            LDAIY  POINTL  GET DATA SPECIFIED BY POINTL(H)
023A D1 00            CMPIY  SAL     COMPARE THIS DATA WITH DATA SPECIFIED BY SAL(H)
023C F0 0F            BEQ    NEXT    DATA EQUAL?
023E 20 88 1D  ANYKEY JSR    SCAND   DISPLAY EPROM ADDRESS AND DATA
```

```
Ø241 DØ FB                BNE   ANYKEY ANY KEY DEPRESSED?
Ø243 2Ø 88 1D             JSR   SCAND  DISPLAY EPROM ADDRESS AND DATA
Ø246 DØ F6                BNE   ANYKEY ANY KEY DEPRESSED?
Ø248 2Ø 88 1D   NKEY      JSR   SCAND  DISPLAY EPROM ADDRESS AND DATA
Ø24B FØ FB                BEQ   NKEY   NO KEY DEPRESSED?
Ø24D 2Ø 5E Ø2   NEXT      JSR   INCMNT INCREMENT SAL(H) AND POINTL(H) BY ONE
Ø25Ø DØ E4                BNE   VER    NOT LAST ADDRESS?
Ø252 4C 1D 1C             JMP   RESET  RETURN TO JUNIOR MONITOR

                ***********
                SUBROUTINES
                ***********

Ø255 A5 Ø4      TRF       LDAZ  MOVL
Ø257 85 FA                STAZ  POINTL TRANSFER MOVL TO POINTL
Ø259 A5 Ø5                LDAZ  MOVH
Ø25B 85 FB                STAZ  POINTH TRANSFER MOVH TO POINTH
Ø25D 6Ø                   RTS

Ø25E 2Ø 88 1D   INCMNT    JSR   SCAND  DISPLAY FOR ABOUT 5MS POINTH, POINTL
                                       AND INH ( = EPROM ADDRESS AND DATA
                                       ON THIS ADDRESS )
Ø261 E6 ØØ                INCZ  SAL    INCREMENT SAL(H) BY ONE
Ø263 DØ Ø2                BNE   INCDA
Ø265 E6 Ø1                INCZ  SAH
Ø267 E6 FA      INCDA     INCZ  POINTL INCREMENT POINTL(H) BY ONE
Ø269 DØ Ø2                BNE   COMP
Ø26B E6 FB                INCZ  POINTH
Ø26D A5 Ø1      COMP      LDAZ  SAH
Ø26F C5 Ø3                CMPZ  EAH    COMPARE EAH WITH SAH
Ø271 DØ Ø4                BNE   RTRN   EAH NOT EQUAL SAH?
Ø273 A5 ØØ                LDAZ  SAL
Ø275 C5 Ø2                CMPZ  EAL    COMPARE EAL WITH SAL
Ø277 6Ø         RTRN      RTS
```

Table 1. The software required for the Junior Computer to program and verify EPROMs.

short while (about 70 ms) and then the value A9 will appear to indicate that the EPROM location has been programmed correctly.
The next address will appear after the '+' key has been operated. New data can then be programmed into this location. The contents of the EPROM can be read by simply depressing the '+' key repeatedly, or by returning to the monitor program of the Junior Computer (depress the reset key).

The duplicate routine

This routine is used to copy a section of memory from one area to another. In order to do this, the memory locations SAL and SAH (ØØØØ and ØØØ1) have to contain the start address of the data block which has to be copied; EAL and EAH (ØØØ2 and ØØØ3) have to contain the end address +1 of the data block to be copied; MOVL and MOVH (ØØØ4 and ØØØ5) have to contain the start address of the memory area which the data block has to be moved to (somewhere inside the EPROM).
The duplicate routine can be started from address Ø222, whereupon the display is blanked. Each time a memory location is programmed the display will light briefly to indicate how the process is progressing. Once the entire data block has been programmed into the EPROM the Junior Computer will display the last address +1.

The verify routine

This routine enables selected data blocks to be compared with the contents of the EPROM. Again, the start address, the end address +1 and the destination address have to be entered before the routine is started (start address = Ø233).
The program will stop when (or if) an error is detected. The address containing the error will then appear on the display, together with the data contained in the EPROM at that location. If any key is then depressed (except for 'reset' or 'NMI') the program will continue with its check. After the final check the Junior Computer will display the last address +1.
With all the information provided in this article, it should be possible to program your own EPROMs quickly and efficiently. You will soon discover the advantages of having your very own EPROM programmer.