experimenting with the SC/MP (4)

By abolishing the restriction of having to work exclusively in binary code, the hexadecimal input/output unit described in this article considerably increases the ease and speed with which the user can communicate with the SC/MP – providing, of course, that he has a thorough grasp of t'ie relevant software.

H. Kampschulte, H. Huschitt

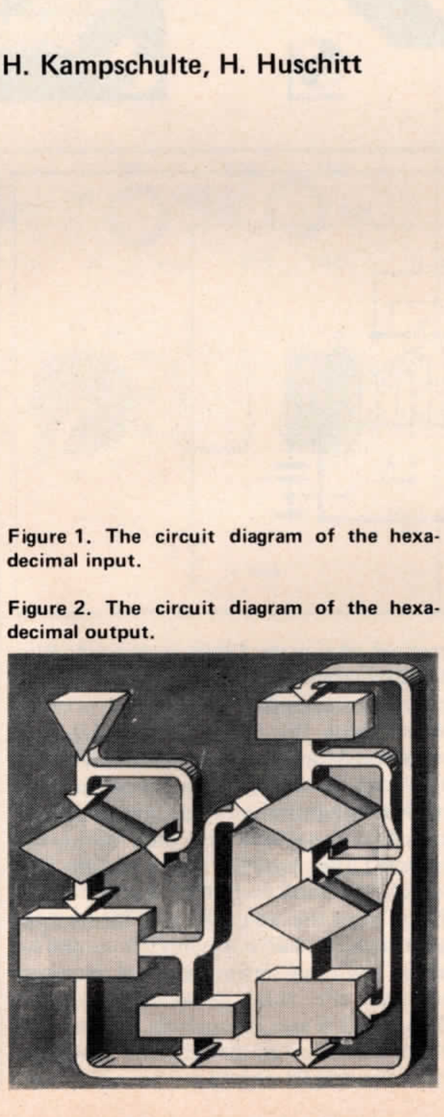
Until now the only method of transferring data to and from the SC/MP system has been via the display LEDs and the address and data switches on the RAM I/O card. This has involved converting hexadecimal data and addresses into their binary equivalents, then setting these up, bit for bit, on the appropriate switches – a procedure which is extremely time-consuming, and also greatly increases the chance of entering a bit falsely. With a hexadecimal input/output however, each hexadecimal digit has its own key and the entered digit is displayed directly on a seven-segment display.
The hardware for the HEX I/O consists of separate input and output circuits.

Hexadecimal input

The keyboard is composed of 24 push-button switches (see figure 1), 16 of which are reserved for the hexadecimal digits ∅ . . . F, whilst the remaining 8 switches (S1 . . . S8) are used as command keys. A command key can be used to make the microprocessor execute a specific routine simply by pressing a single switch.
The 24 keys are connected to three octal-to-binary encoders (IC1 . . . IC3). The outputs of two of the encoders are NANDed together, so that the state of all 24 keys can be expressed in 8 bits. These 8 bits are then routed onto the data bus of the SC/MP via tri-state buffers.
The 8-bit code generated by the keyboard can be divided into three subsections (see table 1a). Bits ∅∅ . . . ∅3 identify the hexadecimal digits, bits ∅4 . . . ∅6 contain the inverted value of the command keys, whilst bit ∅7 is '∅' if no key is pressed.
Table 1b gives three examples of the keyboard code. The left hand column indicates the key which has been pressed and the corresponding code is shown on the right. Only one key should by pressed at a time, since only the code of the 'highest' key will appear at the output if several keys are pressed simultaneously. Keys 8 . . . F have priority over ∅ . . . 7, which in turn override the command keys.
The outputs of the tri-state buffers (IC4, IC5) are enabled (via address logic gates N4 . . . N9) by the address decoder of the CPU card. The keyboard will respond to any address between 17 x 8 and 17 x F, where 'x' can be any value (i.e. 'don't care'). Alternatively, as described in part 3 ('page-address structure'), the address can be located on the first memory page between ∅7 x 8 and ∅7 x F.

Figure 1. The circuit diagram of the hexadecimal input.

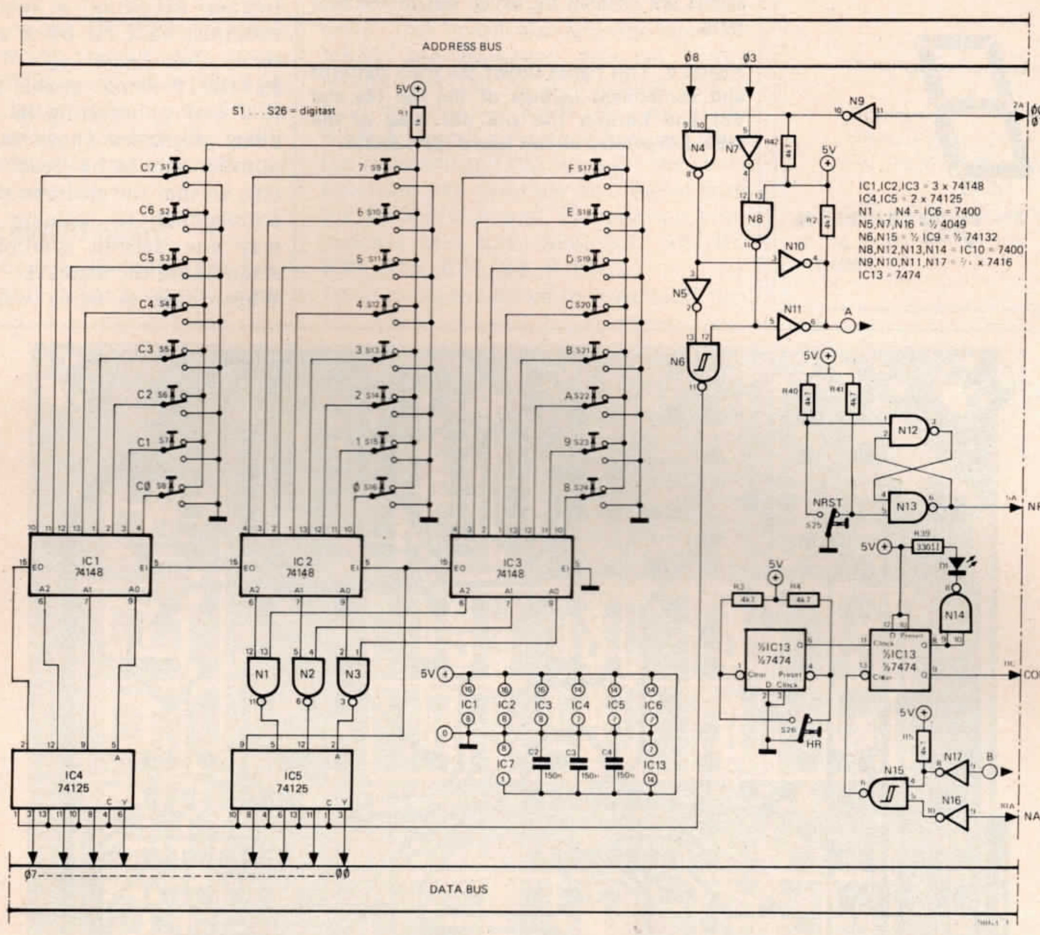
Figure 2. The circuit diagram of the hexadecimal output.



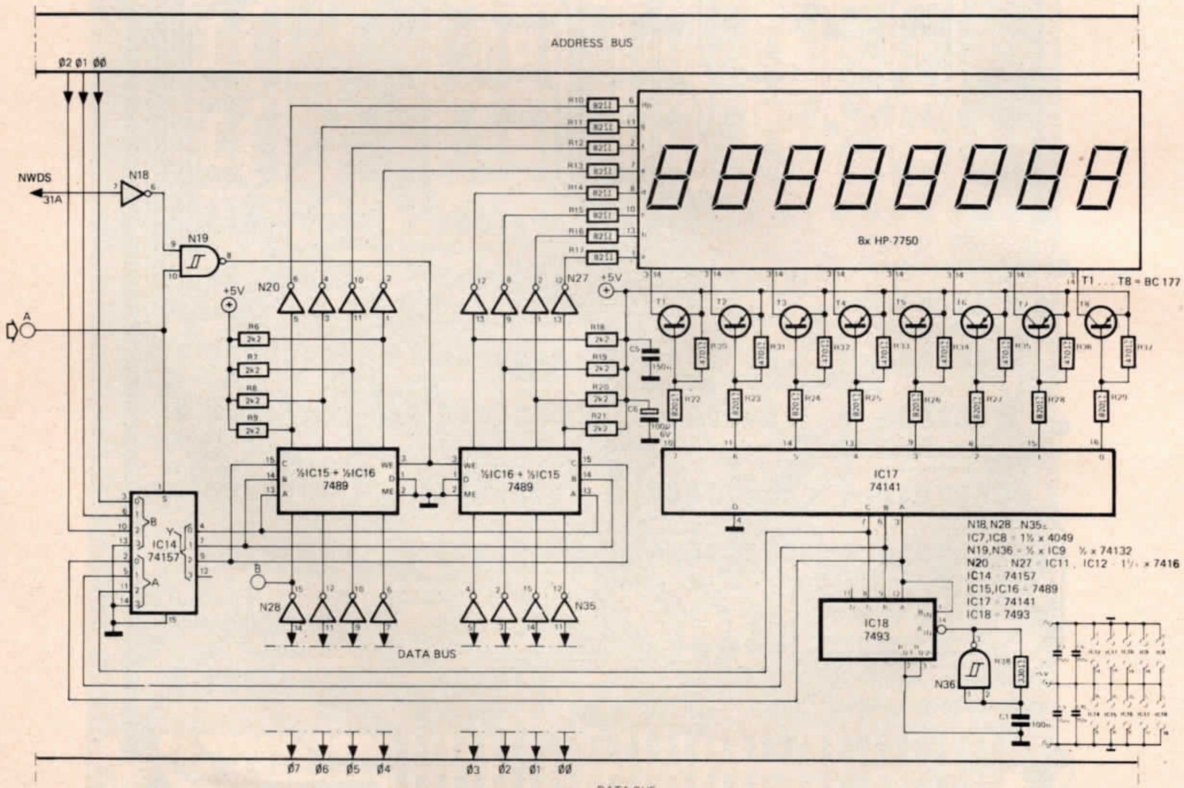
Hexadecimal output

Figure 2 shows the circuit diagram of the hexadecimal output. Basically, this is a relatively self-contained unit that can store eight hexadecimal digits (and several other symbols) and display them on eight seven-segment LED displays.
Each of the LED displays has its own address, so that the output unit actually has a total of eight (groups of) addresses: 17 x ∅ . . . 17 x 7 (or ∅7 x ∅ . . . ∅7 x 7), where 'x' again stands for 'don't care'. The last three bits of the address indicate the particular seven-segment display. Thus the address ending . . . ∅∅∅ will indicate display ∅.
In the circuit, these last three bits are applied to the 'B' input of a multiplexer IC (IC14). When entering new data, this information is passed to the address inputs of a 16 x 8 bit 'scratch-pad' memory (IC15 and IC16). 8 bytes of this memory are used to store the data for the 8 displays, as present on the data bus, during the Negative Write Data Strobe (NWDS).
Having stored the information, the next step is to display it. A clock generator (N36) drives a 4-bit binary counter (IC18), three outputs of which are actually used. These three outputs are connected to the 'A' input of the multiplexer (IC14); when this input is selected (in the display mode) the memory will be scanned continuously and the data for the 8 displays will appear sequentially at its output. At the same time, the three outputs of the counter are decoded by a BCD to Decimal decoder-driver (IC17) and used to enable each of the 8 displays in turn.
The data appearing at the memory output are buffered by the open-collector buffer/drivers N2∅ . . . N27 and used to enable the segments of the displays. Each bit corresponds to one of the seg-

1


ADDRESS BUS
S1 . . . S26 = digitast
IC1,IC2,IC3 = 3 x 74148
IC4,IC5 = 2 x 74125
N1 . . . N4 = IC6 = 7400
N5,N7,N16 = ½ 4049
N6,N15 = ½ IC9 = ½ 74132
N8,N12,N13,N14 = IC10 = 7400
N9,N10,N11,N17 = ⅔ x 7416
IC13 = 7474
NRST
CONT
NADS
DATA BUS

2


ADDRESS BUS
NWDS
8x HP-7750
T1 . . . T8 = BC 177
IC17
74141
IC18
7493
N18,N28 . . . N35 = IC7,IC8 = 1⅓ x 4049
N19,N36 = ½ x IC9 = ½ x 74132
N20 . . . N27 = IC11, IC12 = 1⅓ x 7416
IC14 = 74157
IC15,IC16 = 7489
IC17 = 74141
IC18 = 7493
DATA BUS

3

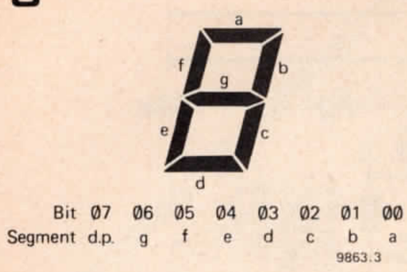


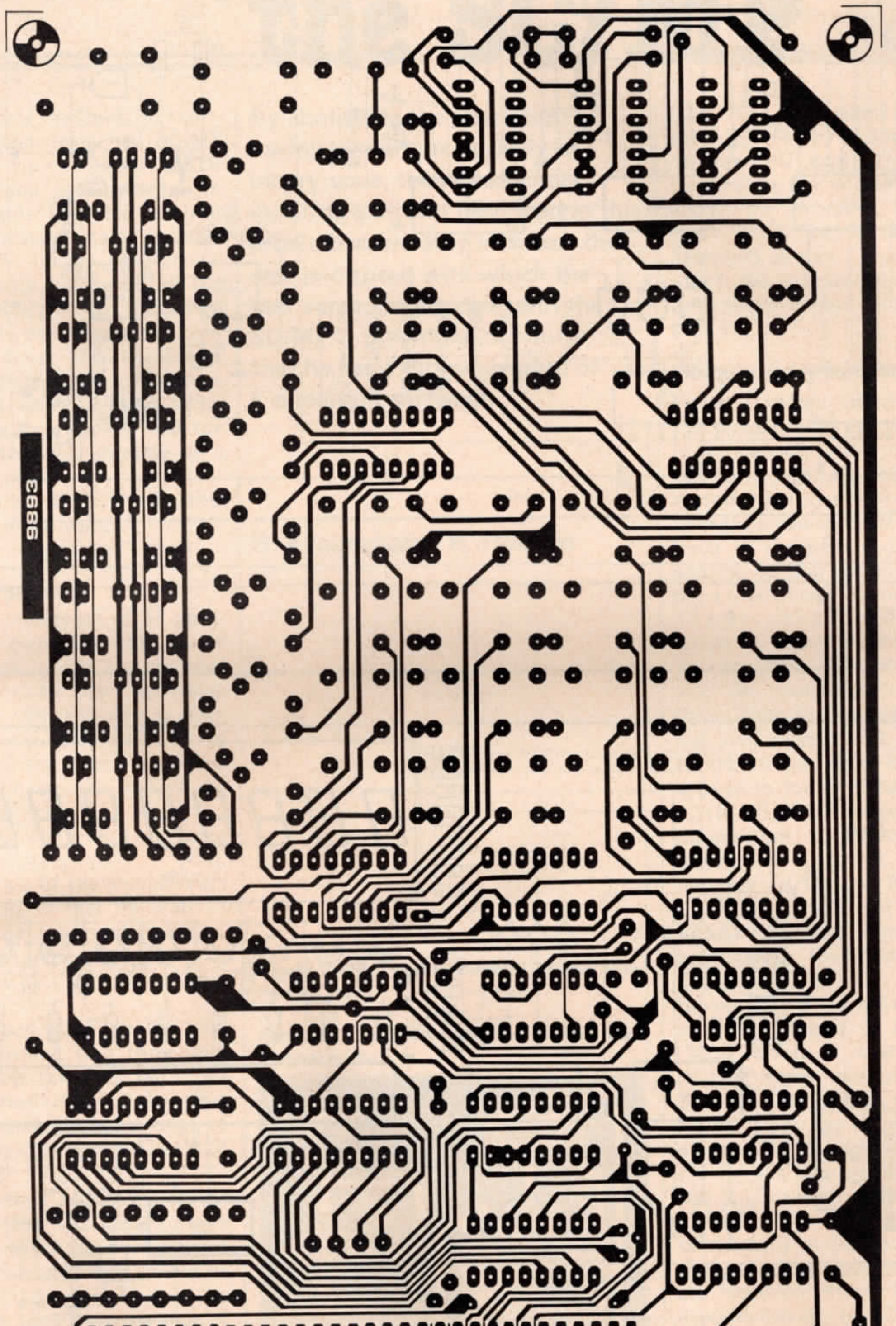
Figure 3. This figure shows which display segments are enabled by which bits of the data byte.

Figure 4. This figure shows the track patterns and component layouts of the top (4a and 4b) and bottom (4c and 4d) sides of the HEX I/O printed circuit board (EPS 9893).

ments, as shown in figure 3, so no further decoding is required. If, for example, data bit ØØ is a '1', then segment 'a' lights up. The data-byte Ø111Ø11Ø would enable segments b, c, e, f and g, resulting in the letter 'H' being displayed (note that the binary number should be 'read' from right to left, so that the extreme right-hand digit corresponds to segment 'a'!). In this way any desired symbol can be represented on the displays.

Whilst data is being written into the

4a



scratch-pad memory the display is randomised. This is due to the fact that the clocked counter (IC18) has no halt facility, thus when the B inputs of the multiplexer (IC14) are enabled, the digit enable is no longer synchronised with the segment drive. However the write-cycle is so short that this effect should be scarcely perceptible.

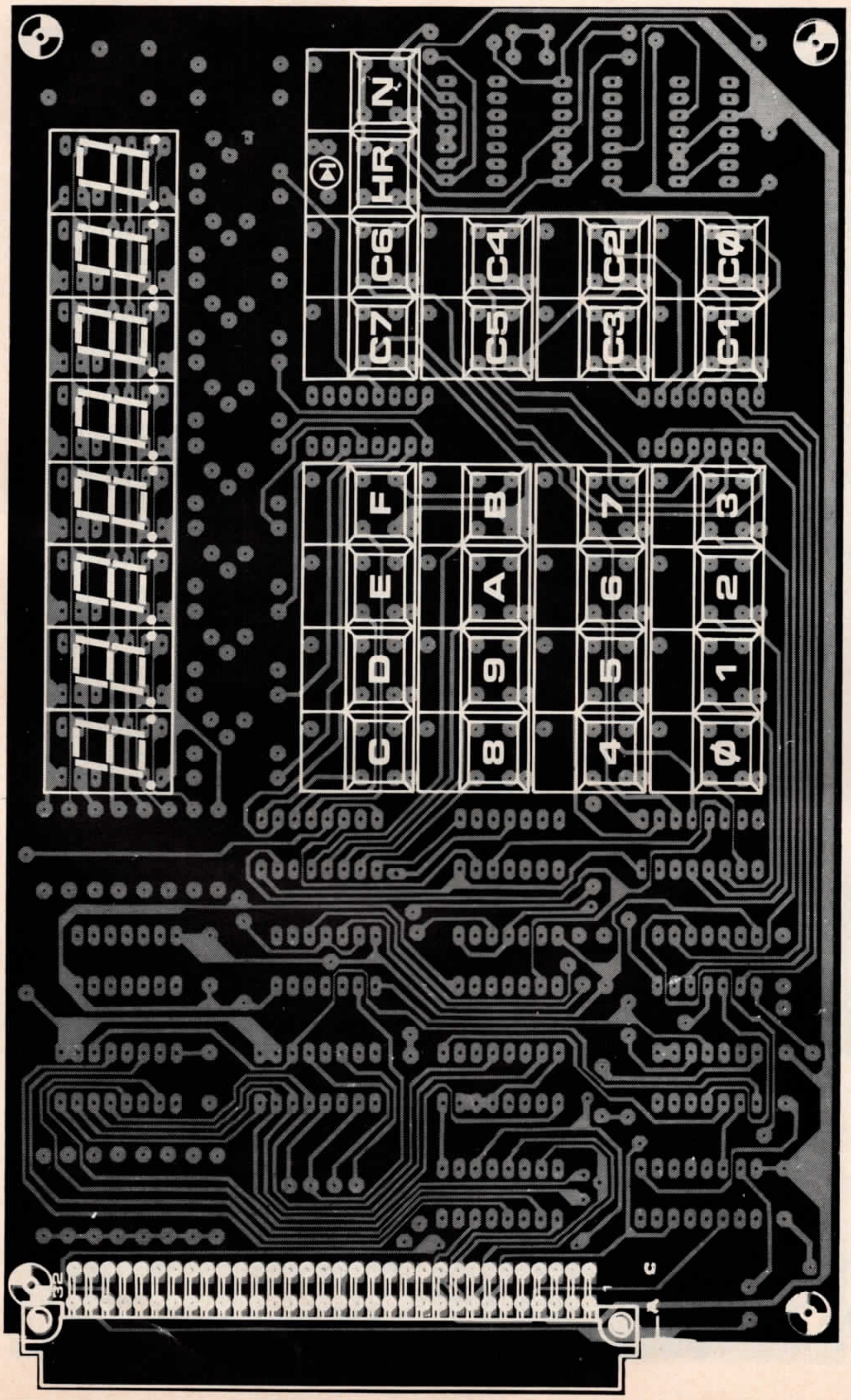
The printed circuit board

All the components (including the keyboard and display) for both the input and output circuits are mounted on the same board. As may be seen from figure 1, this board also accomodates the NRST and Halt-reset switches, which means that the SC/MP system can be built without the RAM I/O card. If the RAM I/O card is retained however, then the switches and flip-flops for the above functions can naturally be omitted from the HEX I/O board. The relevant components are: S25, S26, R3, R4, R5, R39, R40, R41, D1, IC13.
To keep its size down to reasonable pro-

Parts list to figure 4b.

Displays 0 ... 7 = HP-7750
D1 = LED in S26
S1 ... S25 = Schadow digitast SPDT
S26 = Schadow digitast SPDT + LED

4b

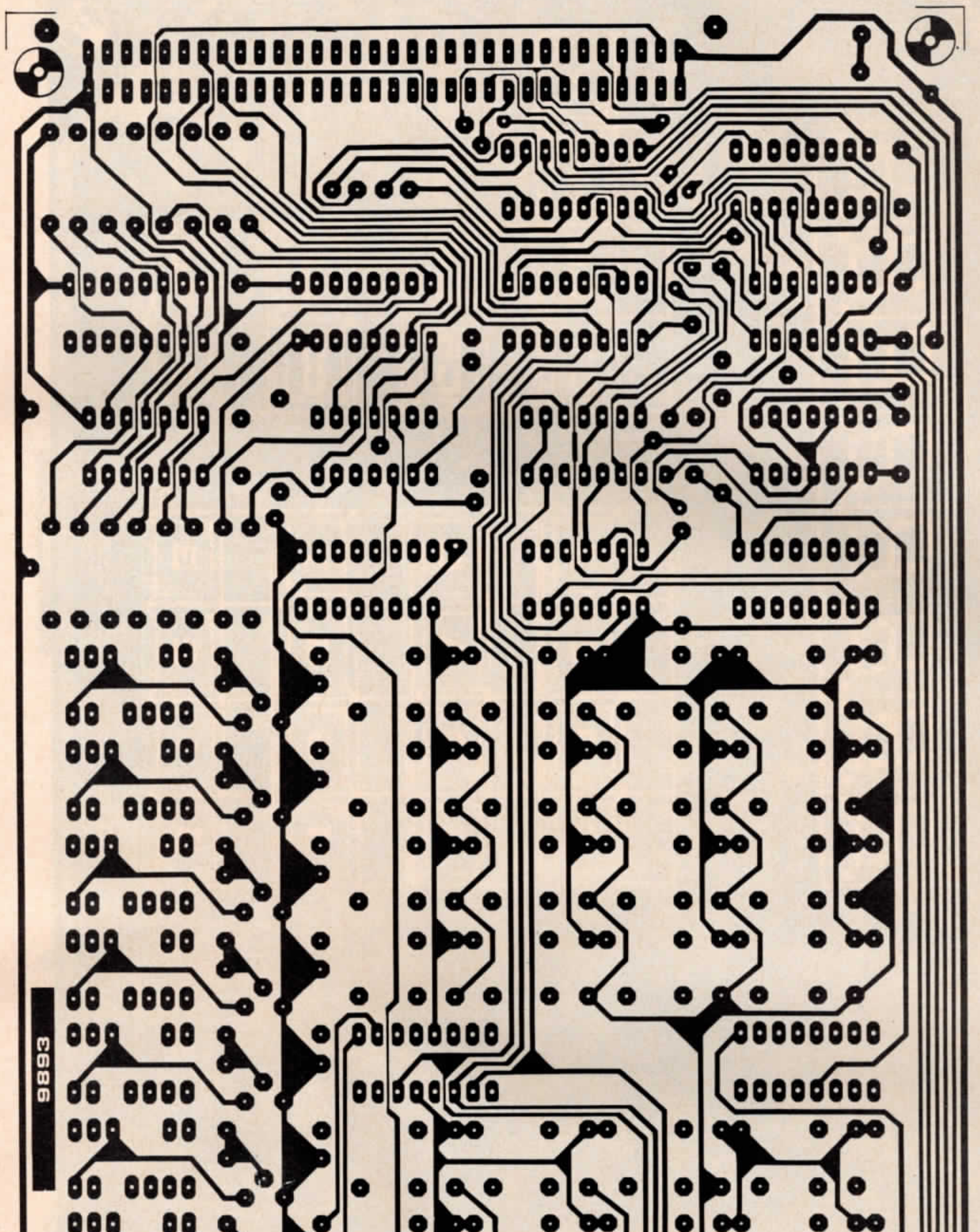


portions, the board is double-sided with plated through holes. Figures 4a and 4b show the track pattern and component layout on the upper side of the board; figures 4c and 4d show the underside. The design of the board takes into account the possibility of mounting the input/output unit in a console or plinth. To this end the upper side of the board (see figure 4b) contains only the keyboard switches, the connector and the displays. Ideally, the displays should then be soldered direct to the board, i.e. without using connector sockets. The board can be covered with a sheet of red perspex, with a section cut out to allow access to the keyboard. All the remaining components are mounted on the underside of the board (see figure 4d).
All connections to and from the HEX I/O board (including those to the busboard) are made via connectors. Figure 5 shows the details of the wiring between the HEX I/O board and the bus board. The connections shown as dotted lines should only be made if the RAM I/O card is omitted.

Power supply

Before starting on the software, it is important to make sure that all hardware is operating satisfactorily. This will not always be the case if the supply voltages are not accurately maintained. All supply voltages should be within 5% of the nominal values, and this is particularly the case if SC/MP II is used (V_{CC} = 5 V ± 5%). Note that this voltage should be present at the pins of the IC!

4c



Even if the output of the power supply itself is within the tolerance, a voltage drop in (excessively) long supply lines may just be sufficient to reduce the voltage at the IC to below the minimum required for reliable performance. In case of doubt, it is advised to check the supply voltages at the pins of the ICs.
Next month we hope to publish a suitable power supply for the SC/MP.

I/O software

In contrast to the RAM I/O card, the

Parts list to figure 4d.

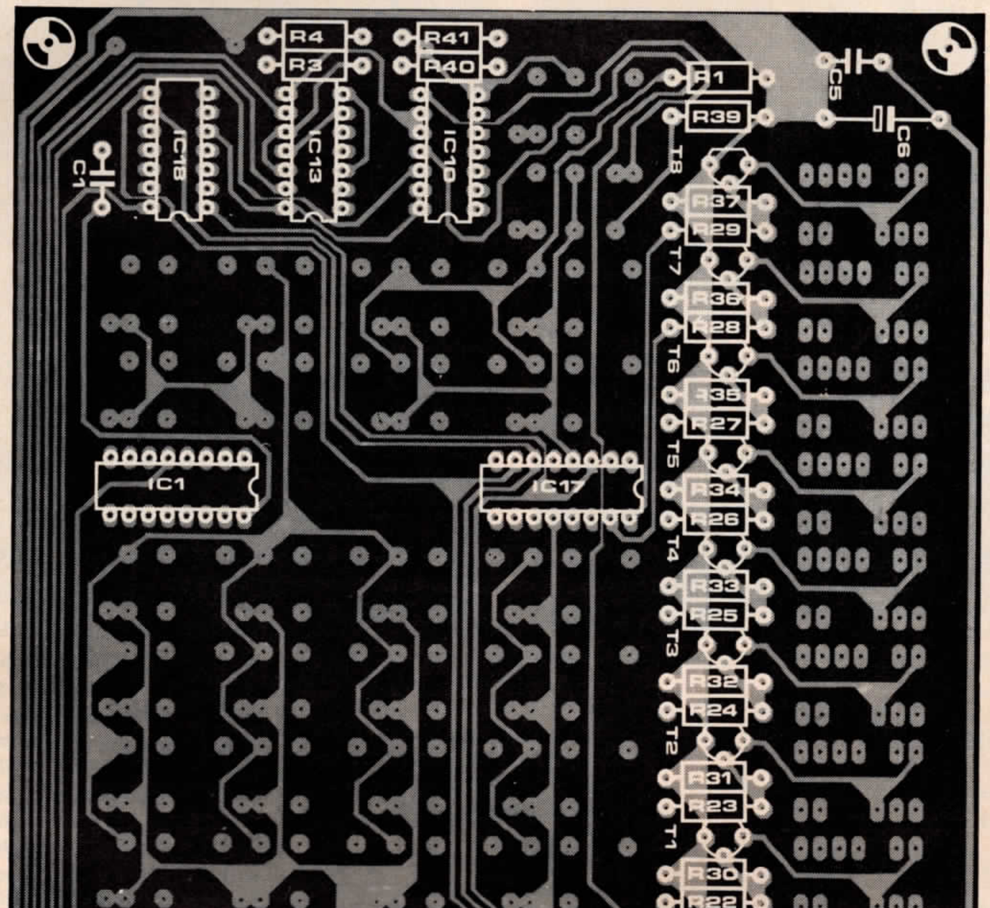
Resistors:
R1 = 1 k
R2 . . . R5, R40 . . . R42 = 4k7
R6 . . . R9, R18 . . . R21 = 2k2
R10 . . . R17 = 82 Ω*
R22 . . . R29 = 820 Ω
R30 . . . R37 = 470 Ω
R38,R39 = 330 Ω

Capacitors:
C1 . . . C5,
C7 . . . C10 = 100 . . . 150 n
C6 = 100 μ/6 V (tantalum)

Semiconductors:
IC1 . . . IC3 = 74148
IC4,IC5 = 74125
IC6,IC10 = 7400
IC7,IC8 = 4049
IC9 = 74132
IC11,IC12 = 7416 (7406)
IC13 = 7474
IC14 = 74157
IC15,IC16 = 7489
IC17 = 74141
IC18 = 7493
T1 . . . T8 = BC 177 or equ.

* If greater display brightness is required, the value of R10 . . . R17 may be reduced to 47 Ω.

4d



5

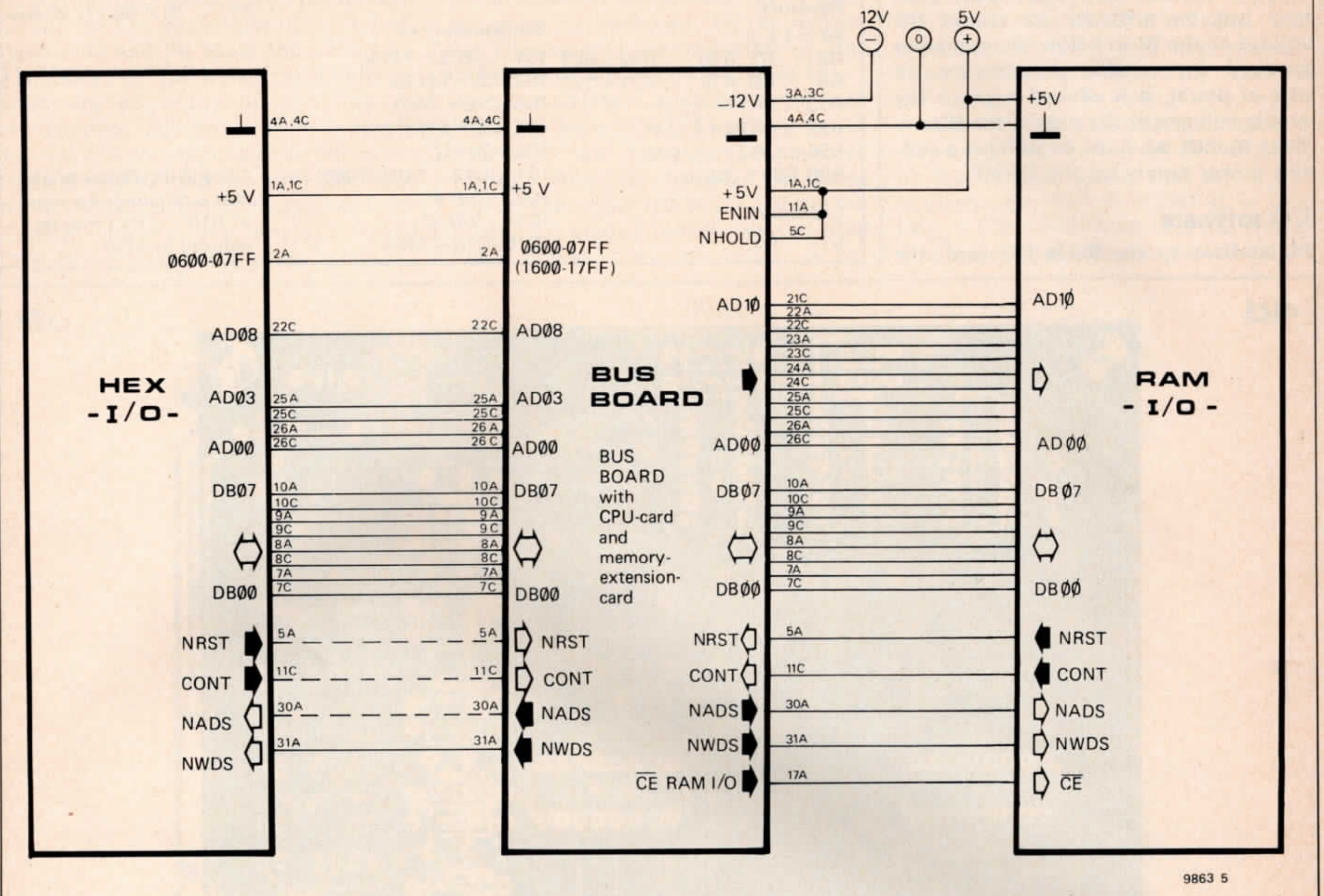


Table 1a.

Bit	Field	Range
03 02 01 00	Data	(0-F)
06 05 04	Command	(0-7)
07	Key	(0,1)

Table 1b.

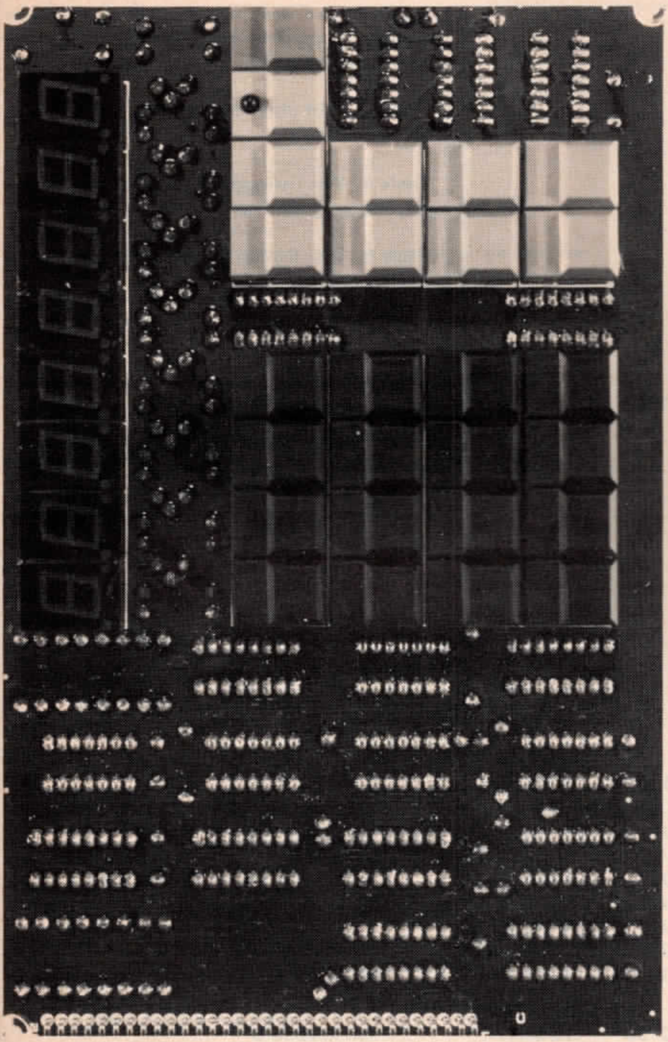
		Code	
		01110000	(70)
Command	3	11000000	(C0)
Data	C	11111100	(FC)

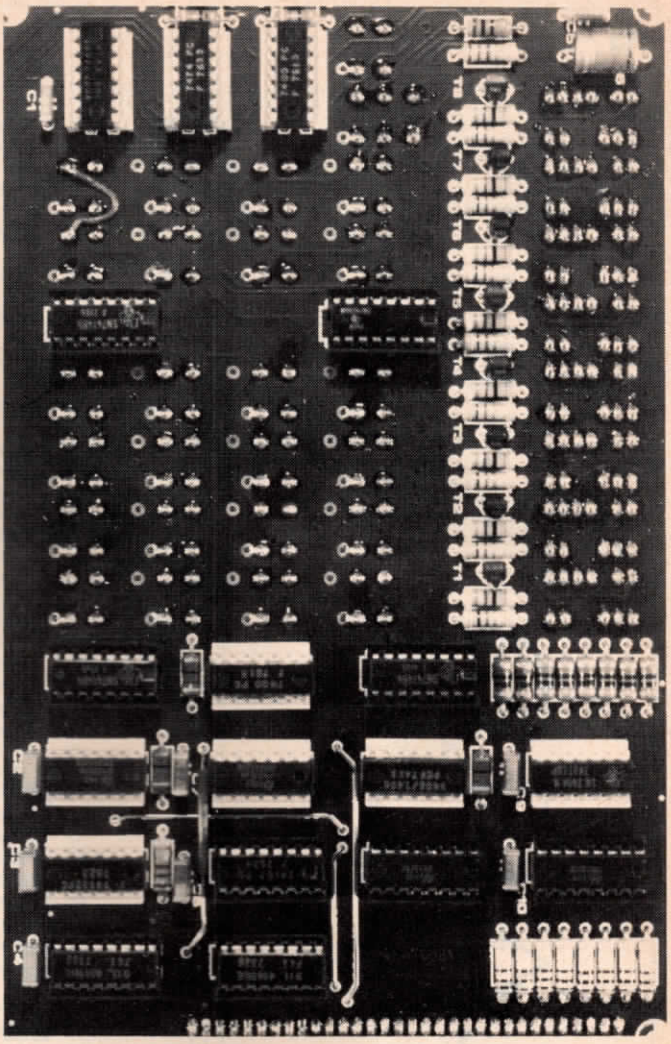
Table 2.

```
                START = 0000
0000   08       NOP
0001   C417     LDI 17        ; load PTR 1 with address
0003   35       XPAH 1        ; of output
0004   C400     LDI 00        ; load data for display 5
0006   C905     ST 5 (1)      ; into display memory
0008   C454     LDI 54        ; load letter 'n' for
000A   C907     ST 7 (1)      ; display 7
000C   C45C     LDI 5C        ; load letter 'o' for
000E   C906     ST 6 (1)      ; display 6 and
0010   C901     ST 1 (1)      ; display 1
0012   C479     LDI 79        ; 'E' for
0014   C904     ST 4 (1)      ; display 4
0016   C450     LDI 50        ; etc.
0018   C900     ST 0 (1)
001A   C902     ST 2 (1)
001C   C903     ST 3 (1)
001E   00       HALT
                • END
```

hexadecimal input/output requires the assistance of a certain amount of software to perform its task. As far as the output section is concerned, this is relatively simple. To transfer the required data to a particular display a 'STORE' instruction is used. The displacement value of this instruction determines which display is addressed.
In the example shown in figure 6, the letter 'P' is to be displayed on display 3. In this case the effective address is calculated using indexed addressing via PTR 1.
Table 2 lists a programme which will also test the hardware involved. This programme, which must be loaded into the RAM of the RAM I/O card, will display the words 'no Error'.
A second example programme which will display a well-known name is listed in table 3.

The software required to interrogate the keyboard is somewhat more complicated. Figure 7 shows the flow-diagram for the simplest possible programme which allows the state of the 16 data keys to be tested. After the start of the programme, the 8 bits of keyboard data are loaded into the AC. This continues until bit 07 is '1', thereby indicating that a key has been pressed. Before the keyboard data is further processed, a delay instruction is executed to ensure the data is valid (i.e. allow for contact bounce).





Since only the state of the data keys is to be tested, the AC is first masked by ØF, after which the new contents of the AC are stored in memory. The state of data-bit Ø7 is tested once more, and so on until it is 'Ø' (the key is released), upon which a second delay instruction follows. The cycle may then be repeated if so desired.

A slightly more complicated demonstration programme which will display the contents of the data keys on the readout is listed in table 4. Once the programme has been loaded into the RAM on the RAM I/O card and started by operating the NRST switch, pressing one of the data keys, e.g. key A, will result in the letter A appearing on display Ø. If then key B is pressed, the letter B will appear on the next display, and so on until all 8 displays are enabled. The programme can then be repeated by operating the NRST.

The above programme is simply intended to demonstrate the HEX I/O, and cannot in fact do anything apart from display the 'contents' of a data key. The programme which enables the keyboard to perform its true function, i.e. modify the contents of the memory, is given in table 5 (see figure 8 for the flow-diagram). This involves investing a certain amount of time, since the programme, which is 200 bytes long, must be written into RAM using the data switches.

Table 1. This table shows the formats of the 8-bit code generated by the keyboard.

Table 2. This programme will cause the words 'no Error' to appear on the displays.

Table 3. A 'surprise' programme.

Figure 5. The connections between the HEX I/O board and the bus board. The connections which are shown as dotted lines are made only if the NRST and Halt-reset functions on the RAM I/O card are no longer being used.

Table 3.

ØØØØ	Ø8
ØØØ1	C417
ØØØ3	35
ØØØ4	C45Ø
ØØØ6	C9ØØ
ØØØ8	C45C
ØØØA	C9Ø1
ØØØC	C478
ØØØE	C9Ø2
ØØ1Ø	C475
ØØ12	C9Ø3
ØØ14	C479
ØØ16	C9Ø4
ØØ18	C9Ø6
ØØ1A	C438
ØØ1C	C9Ø5
ØØ1E	C4ØØ
ØØ2Ø	C9Ø7
ØØ22	ØØ

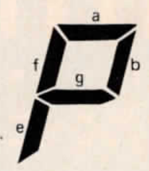
Once the programme is started (by pressing the NRST key), the display will show 1EØØxx, where 'xx' are the contents of 'address' 1EØØ. The command key which generates the code 1ØØØØØØØ (i.e. key CØ) is the 'NEXT' key. Pressing this key results in 1EØ1yy appearing on the displays, where 'yy' represent the contents of address 1EØ1. In this way it is possible to read out the contents of every location of the same page in memory. After 1FFF the displays show 1ØØØ since there is no carry to the 4 most significant address bits.

In order to modify the contents of a memory location, the new data is entered at the desired address by means of the data keys. For example, pressing key A twice, will result in 'AA' being written into the desired address.

If the 'user's programme' is to commence at a start address other than 1EØØ, then the loader programme should be modified accordingly. Using the data switches, the lower-order byte of the desired start address is loaded into address ØØØ9 and the higher-order byte into address ØØØC.

Once the user's programme has been loaded into memory, one would normally expect to start it by operating the NRST key. However, since the loader programme precedes the user's programme in the memory, the loader programme must first be modified: it will have to start with an instruction 'jump

6



```
'P'   d,p.  g  f  e  d  c  b  a
       Ø    1  1  1  Ø  Ø  1  1  = 73
                              LDI 73        C473

(PTR 1) = 17ØØ
Address display 3 = 17x3      ST3 (1)       C9Ø3
                                  9863 6
```

Figure 6. An example of how a particular symbol, in this case the letter P, is presented on a specific display.

Figure 7. The flow diagram for a simple keyboard routine.

Table 4. The listing for the HEX I/O demonstration programme.

Table 5. The listing for the HEX I/O loader programme.

7

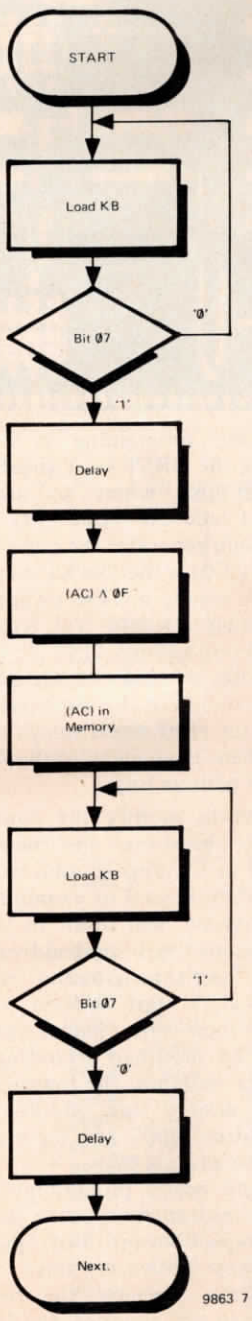


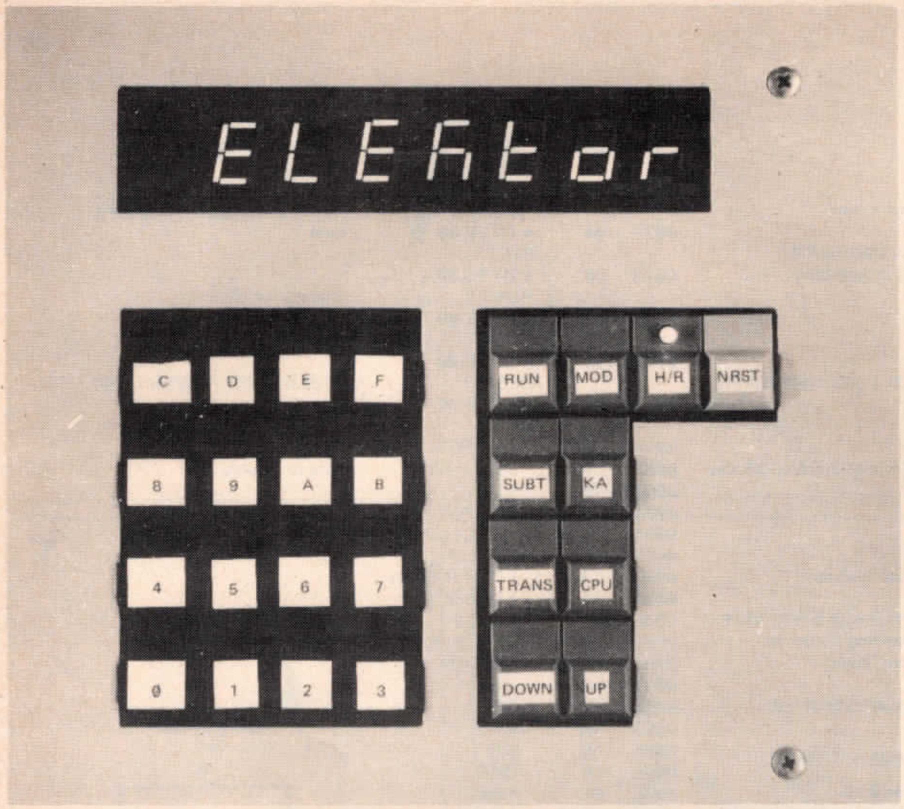
Table 4.

```
                    START = ØØØØ
ØØØØ    Ø8            NOP
ØØØ1    C4Ø8          LDI L (KB)
ØØØ3    31            XPAL 1              ; load PTR 1 with
ØØØ4    C417          LDI H (KB)          ; EA of 'KB'
ØØØ6    35            XPAH 1
ØØØ7    C4ØØ          LDI L (DISPL)
ØØØ9    32            XPAL 2
ØØØA    C417          LDI H (DISPL)       ; load PTR 2 with
ØØØC    36            XPAH 2              ; EA of 'DISPL'
                    LABEL 1:
ØØØD    C1ØØ          LD Ø (1)            ; load keyboard
ØØØF    94FC          JP LABEL 1          ; bit 7 = Ø, no key pressed
ØØ11    D4ØF          ANI ØF              ; mask bits Ø - 3
ØØ13    Ø1            XAE                 ; in E, indirect addressing
ØØ14    C427          LDI L (TAB)
ØØ16    33            XPAL 3              ; 'TAB' to PTR 3 (higher byte = ØØ)
ØØ17    C38Ø          LD-128 (3)          ; addressing via E
ØØ19    CEØ1          ST @ 1 (2)          ; display 7-segment 'code'
ØØ1B    8FØA          DLY ØA              ; delay approx 10 msec.
                    LABEL 2:
ØØ1D    C1ØØ          LD Ø (1)            ; wait until key is released
ØØ1F    94Ø2          JP DLY              ;
ØØ21    9ØFA          JMP LABEL 2
ØØ23    8FØA          DLY ØA              ; delay approx. 10 msec.
ØØ25    9ØE6          JMP LABEL 1
                    TAB:                  ; Table with 7-segment code.
ØØ27    3F          • BYTE 3F, Ø6, 5B, 4F, 66, 6D, 7D, Ø7
ØØ28    Ø6
ØØ29    5B
ØØ2A    4F
ØØ2B    66
ØØ2C    6D
ØØ2D    7D
ØØ2E    Ø7
ØØ2F    7F          • BYTE 7F, 6F, 77, 7C, 58, 5E, 79, 71
ØØ3Ø    6F
ØØ31    77
ØØ32    7C
ØØ33    58
ØØ34    5E
ØØ35    79
ØØ36    71
                    • END
```

Table 5.

```
                HEX I/O LOADER
ØØØØ   Ø8        NOP
ØØØ1   9ØØ5      JMP START        ; programme start
ØØØ3   Ø8        NOP
ØØØ4   Ø8        NOP              ; space for start routine
ØØØ5   Ø8        NOP              ; of user's programme
ØØØ6   Ø8        NOP
ØØØ7   Ø8        NOP
                START:
ØØØ8   C4ØØ      LDI L (ADR)
ØØØA   31        XPAL 1           ; load PTR 1 with start address
ØØØB   C41E      LDI H (ADR)
ØØØD   35        XPAH 1
ØØØE   C4Ø2      LDI L (DISPL)
ØØ1Ø   32        XPAL 2           ; load PTR 2 with EA of display
ØØ11   C417      LDI H (DISPL)    ; digit 2
ØØ13   36        XPAH 2
                $1:
ØØ14   C4Ø2      LDI Ø2
ØØ16   C866      ST COUNT         ; load cycle counter
ØØ18   C4C9      LDI L (TAB)
ØØ1A   33        XPAL 3           ; load PTR 3 with EA of table
ØØ1B   C4ØØ      LDI H (TAB)      ; for converting binary to
ØØ1D   37        XPAH 3           ; 7-segment 'code'
ØØ1E   31        XPAL 1
ØØ1F   C859      ST RAM           ; fetch lower byte of 'adr'
ØØ21   31        XPAL 1
                $2:
ØØ22   C4ØF      LDI ØF
ØØ24   DØ54      AND RAM          ; mask bits Ø - 3
ØØ26   Ø1        XAE              ; load E for indirect addressing
ØØ27   C38Ø      LD – 128 (3)     ; fetch 7-segment 'code' and
ØØ29   CEØ1      ST @ 1 (2)       ; store in display, digit 2
ØØ2B   CØ4D      LD RAM
ØØ2D   1C        SR               ; bits 4 - 7 of lower byte of
ØØ2E   1C        SR               ; PTR 1 (higher byte in second
ØØ2F   1C        SR               ; cycle) = 'adr', are shifted
ØØ3Ø   1C        SR               ; into bits Ø - 3
ØØ31   Ø1        XAE
ØØ32   C38Ø      LD – 128 (3)     ; fetch 7-segment code
ØØ34   CEØ1      ST @ 1 (2)
ØØ36   B846      DLD COUNT
ØØ38   98Ø8      JZ $3            ; 2 cycles completed?
ØØ3A   35        XPAH 1
ØØ3B   C83D      ST RAM           ; fetch higher 'adr' byte
ØØ3D   35        XPAH 1
ØØ3E   9ØE2      JMP $2           ; jump for second cycle
                $3:
ØØ4Ø   C4ØØ      LDI ØØ
ØØ42   CEØ1      ST @ 1 (2)       ; clear displays 6 and 7
ØØ44   CEØ1      ST @ 1 (2)
ØØ46   C6FA      LD @ – 6 (2)     ; reset PTR 2
ØØ48   C1ØØ      LD Ø (1)         ; load contents of 'adr'
ØØ4A   D4ØF      ANI ØF
ØØ4C   Ø1        XAE
ØØ4D   C38Ø      LD – 128 (3)     ; fetch 7-segment code
ØØ4F   CAFE      ST – 2 (2)       ; store in display 'Ø'
ØØ51   C1ØØ      LD Ø (1)
ØØ53   1C        SR               ; shift (adr) 4 bits to the right
ØØ54   1C        SR
ØØ55   1C        SR
ØØ56   1C        SR
ØØ57   Ø1        XAE
ØØ58   C38Ø      LD – 128 (3)     ; fetch 7-segment code
ØØ5A   CAFF      ST – 1 (2)       ; store in display '1'
ØØ5C   31        XPAL 1
ØØ5D   C81D      ST P1L           ; store (PTR 1)
ØØ5F   35        XPAH 1
ØØ6Ø   C81B      ST P1H
ØØ62   C4A2      LDI L (LDKB) – 1
ØØ64   33        XPAL 3
ØØ65   C4ØØ      LDI H (LDKB)     ; start 'LDKB' (= keyboard
ØØ67   37        XPAH 3           ; routine)
ØØ68   3F        XPPC 3
ØØ69   CØ1Ø      LD KBOARD
ØØ6B   E48Ø      XRI X'8Ø
ØØ6D   9C1Ø      JNZ $4
                $11
ØØ6F   CØØB      LD P1L
ØØ71   31        XPAL 1           ; reload PTR 1
ØØ72   CØØ9      LD P1H
ØØ74   35        XPAH 1
ØØ75   C5Ø1      LD @ 1 (1)       ; increment PTR 1
ØØ77   9Ø9B      JMP $1           ; jump back without loading
                RAM:              ; into 'adr'
ØØ79   ØØ       • BYTE ØØ
                KBOARD:           ; memory storage for keyboard
ØØ7A   ØØ       • BYTE ØØ         ; data
                P1L:
ØØ7B   ØØ       • BYTE ØØ
                P1H:              ; 2 bytes for (PTR 1)
ØØ7C   ØØ       • BYTE ØØ
                COUNT:
ØØ7D   ØØ       • BYTE ØØ
                SEGM 7:           ; RAM-byte for 7-segment code
ØØ7E   ØØ       • BYTE ØØ
                $4:
ØØ7F   CØFE      LD SEGM 7
ØØ81   CAFF      ST – 1 (2)         data to display '1'
ØØ83   4Ø        LDE
ØØ84   C8F4      ST RAM
ØØ86   C4A2      LDI L (LDKB) – 1
ØØ88   33        XPAL 3
ØØ89   C4ØØ      LDI H (LDKB)
ØØ8B   37        XPAH 3
ØØ8C   3F        XPPC 3           ; to keyboard routine
ØØ8D   CAFE      ST – 2 (2)       ; data to display 'Ø'
ØØ8F   CØE9      LD RAM
ØØ91   1E        RR
ØØ92   1E        RR
ØØ93   1E        RR
ØØ94   1E        RR
ØØ95   58        ORE
ØØ96   Ø1        XAE
ØØ97   CØE3      LD P1L
ØØ99   31        XPAL 1
ØØ9A   CØE1      LD P1H           ; reload PTR 1 with
ØØ9C   35        XPAH 1           ; previous contents
ØØ9D   4Ø        LDE
ØØ9E   C9ØØ      ST Ø (1)
ØØAØ   3F        XPPC             ; in 'LDKB' wait for key = '1'
ØØA1   9ØCC      JMP $11
                LDKB:             ; load keyboard routine
ØØA3   C4Ø8      LDI L (KB)
ØØA5   31        XPAL 1
ØØA6   C417      LDI H (KB)
ØØA8   35        XPAH 1           ; load PTR 1 with EA of keyboard
                $5:
ØØA9   C1ØØ      LD Ø (1)
ØØAB   94FC      JP $5            ; wait for key to be pressed
ØØAD   C8CC      ST KBOARD        ; keyboard code to memory
ØØAF   D4ØF      ANI ØF           ; mask bits Ø - 3
ØØB1   Ø1        XAE
                $6:
ØØB2   8FØA      DLY ØA           ; delay approx. 10 msec.
ØØB4   C1ØØ      LD Ø (1)
ØØB6   94Ø2      JP $7
ØØB8   9ØFA      JMP $6           ; wait until key is released
                $7:
ØØBA   8FØA      DLY ØA           ; delay approx. 10 msec.
ØØBC   C4C9      LDI L (TAB)
ØØBE   31        XPAL 1
ØØBF   C4ØØ      LDI H (TAB)      ; load PTR 1 with EA of 'tab'
ØØC1   35        XPAH 1
ØØC2   C18Ø      LD – 128 (1)     ; fetch 7-segment code
ØØC4   C8B9      ST SEGM 7
ØØC6   3F        XPPC 3           ; jump back for
ØØC7   9ØDA      JMP LDKB         ; new start
                TAB:
ØØC9   3F       • BYTE 3F, Ø6, 5B, 4F, 66, 6D, 7D, Ø7
ØØCA   Ø6
ØØCB   5B
ØØCC   4F
ØØCD   66
ØØCE   6D
ØØCF   7D
ØØDØ   Ø7
ØØD1   7F       • BYTE 7F, 6F, 77, 7C, 58, 5E, 79, 71
ØØD2   6F
ØØD3   77
ØØD4   7C
ØØD5   58
ØØD6   5E
ØØD7   79
ØØD8   71

                • END
```



8

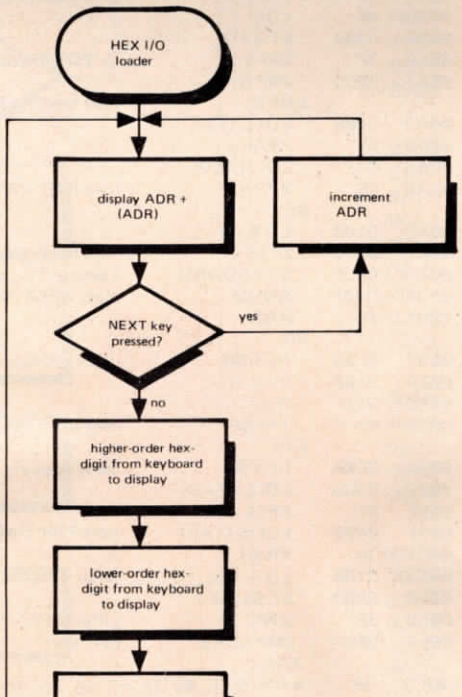


Figure 8. The flow-diagram for the HEX I/O loader programme. See table 5.

Table 6. The start routine.

Table 6.

```
0000   08     NOP
0001   C400   LDI 00
0003   33     XPAL 3
0004   C41E   LDI X'1E
0006   37     XPAH 3
0007   3F     XPPC 3
```

to user's programme'. A suitable start routine is shown in table 6; this routine must be entered (using the data switches) before operating the NRST key to start the user's programme. If a start address other than 1E00 is used, the start routine must be modified accordingly.
In order to be able to use the HEX I/O loader programme again, the beginning of this programme must restored to its original state.
It must be admitted that the start procedure for the user's programme is slightly awkward. However the alternative would involve further lengthening of the 200-byte loader programme, and without a cassette interface this is not really practical. Once the system is equipped with a cassette interface however, a programme of this length need be keyed into memory once only, since it can then be stored permanently on tape. Details of the cassette interface for the SC/MP will follow in a subsequent article. ■

Missing link

Experience has shown that the SC/MP II will not always work reliably in combination with the RAM I/O card. The reason is that the SC/MP II has a lower fan-out than the older SC/MP, for which the system was originally designed.
The problem can usually be solved by replacing IC4, IC5 and IC14 on the RAM I/O card by their 'low-power' equivalents: 74LS75 for IC4 and IC5, and 74LS00 for IC14.
If the problem persists after this modification, several ICs on the CPU card can also be replaced by their low-power equivalents:
IC7 = 74LS00 and
IC9 . . IC12 = 74LS125.

An extra decoupling capacitor, C7, has been added on the memory card (EPS 9863) — as can be seen in the top right-hand corner of the component layout shown in figure 5 (part 3). The value should be 150 n.