BASIC microcomputer

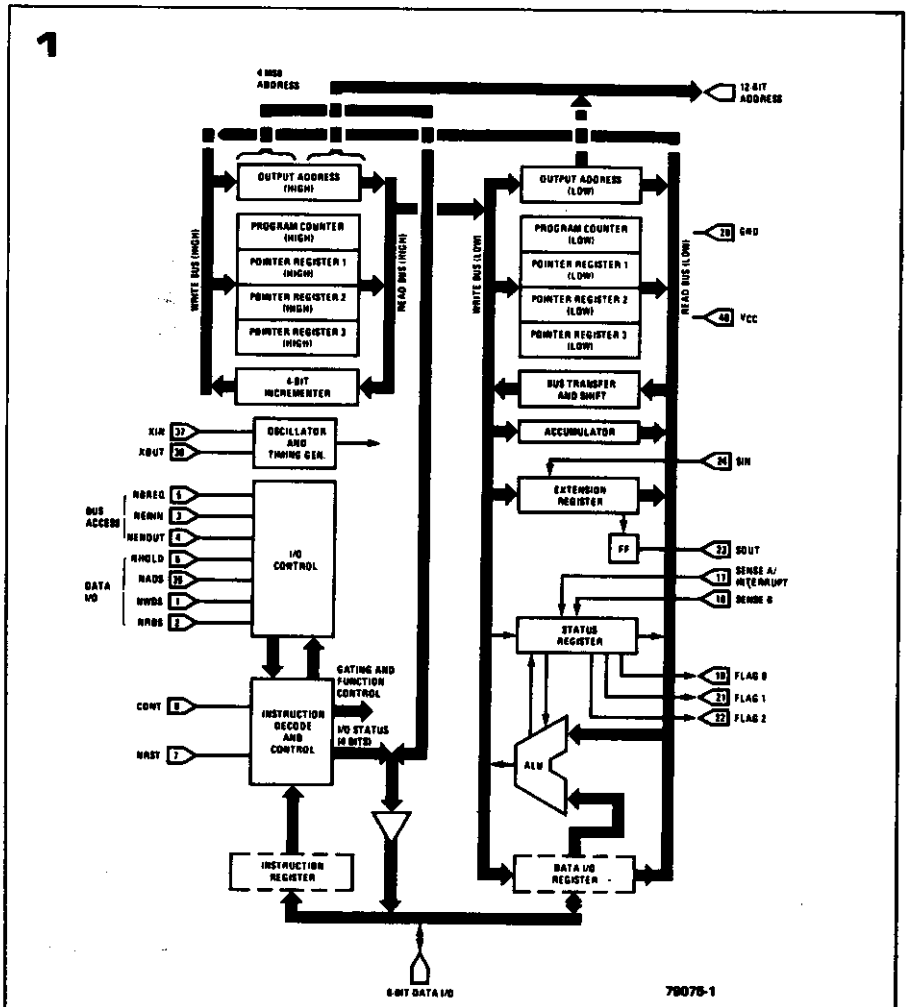
A SC/MP μP with BASIC interpreter

It seems safe to assume that the 'BASIC microcomputer' is the cheapest home-construction computer ever described that can be programmed using a higher programming language.
The SC/MP is a popular and readily-available microprocessor. Two further good reasons for using it in this microcomputer are that it can readily be incorporated into the Elektor SC/MP system, and that a Tiny BASIC interpreter for this μP is available in ROM (Read Only Memory).
The BASIC computer card described in this article contains three circuits that can be used as more or less independent units. The processor section is a fully buffered and self-contained 'CPU card' with provisions for DMA (Direct Memory Access) and multiprocessing.
The memory section is also fully independent, and contains the BASIC interpreter (NIBL-ROM) and the address decoder.
Communication with the 'outside world' (the Elekterminal, for instance) is taken care of by the third section: the interface.
To be fully operational, the computer requires at least one 4K RAM card (RAM = Random Access Memory), as described in Elektor, March 1978. The basic BASIC computer therefore consists of not more than two Eurocard-sized printed circuit boards!

The main advantage of a higher programming language is that there is no need to know the exact details of how the 'inside' of the computer works. A minor disadvantage is that a more sophisticated in- and output unit ('terminal') is required, with an alphanumeric keyboard. In other words, a keyboard that is similar to that of a typewriter. Furthermore, a serial data flow ('bit by bit'!) between computer and terminal is normally required. The Elekterminal with ASCII keyboard (Elektor, November and December 1978) meets these requirements, and this unit or a

Figure 1. Functional block diagram of the INS 8060.

Figure 2. Block diagram of the BASIC microcomputer/CPU card.

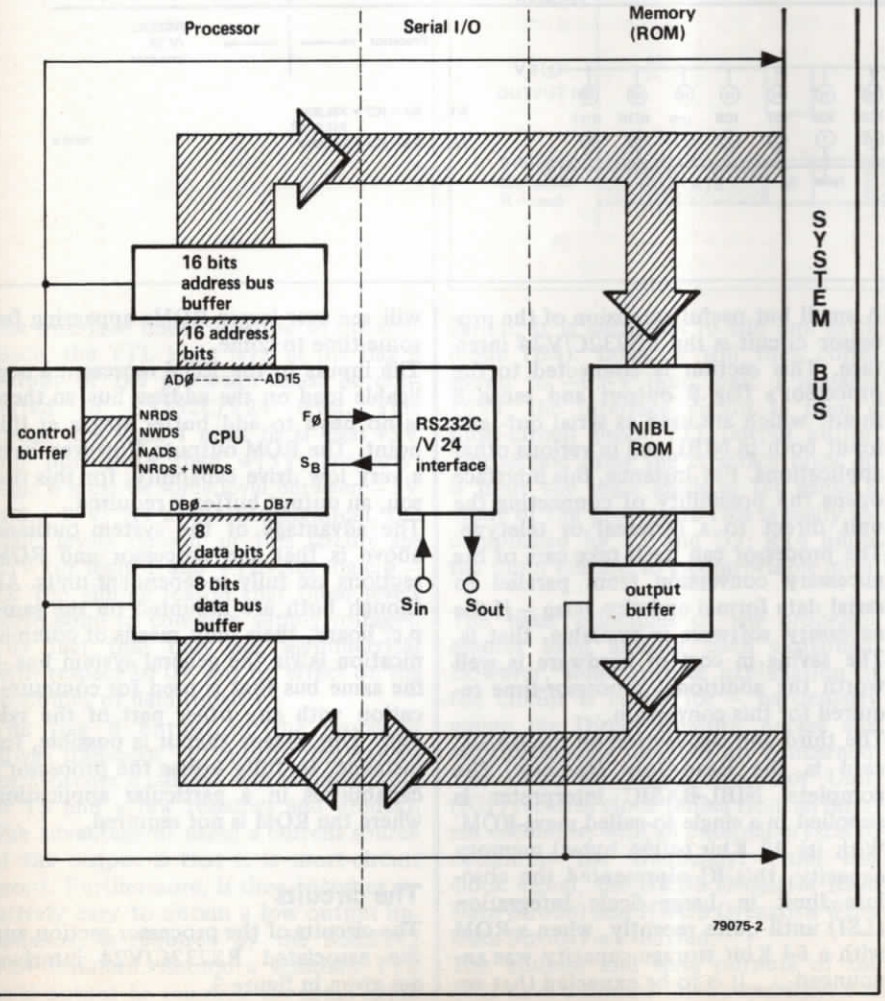


similar terminal must be used in conjunction with the BASIC computer.
Programming in BASIC is easily learned, but it is not so easy to explain all the details in a few pages. For this reason, no attempt will be made in this article to explain how to program in NIBL (National's Industrial BASIC Language). The BASIC course, which started in the recent March issue of Elektor, must suffice. It explains BASIC in general and, as required, deals with NIBL in particular. Obviously, it was written with this BASIC microcomputer in mind!
For this article, software is a side issue. The primary concern is the microcomputer hardware.
However, as stated at the outset: if the aim is to program in BASIC, there is no real need to know how the computer works. Most of the following article would therefore appear to be superfluous: certainly if one has some experience in programming in BASIC, the components can simply be mounted on the board and, (after a quick glance at the summary of NIBL statements and commands) everything's ready to roll. However, NIBL not only offers the possibility of programming in (Tiny) BASIC; it also provides for immediate addressing of the hardware. For this reason, it can be useful to know a little bit about the actual circuit...

Bird's-eye view of the CPU

The SC/MP (Simple Cost-effective Micro Processor) is an 8-bit μP, with all essential functions integrated on a single chip. As is apparent from the block diagram (figure 1), the SC/MP (type number INS 8060) contains four 16-bit registers: the program counter and three pointer registers. These 'pointers' play an important part in the (auto-) indexed addressing of the memory and input/output units.
The (8-bit) extension register is of particular interest, since it offers a serial in- and output facility with a minimum of fuss. The cassette interface in the Elektor SC/MP system makes full use of this possibility. A UART (Universal Asynchronous Receiver/Transmitter), as used in the Elekterminal, can also be made redundant by utilising the SIN and SOUT connections.
The status register can also be used for serial transfer of data. The three 'flag' connections can be used as outputs; 'sense A' and 'sense B' are both serial inputs. In fact, NIBL uses Flag 0 and Sense B as serial data out- and input respectively.
The INS 8060 can address 64k bytes of memory. This requires 16 address lines, 12 of which are brought out direct via pins of the IC. The four remaining MSB's (Most Significant Bits) are applied to four lines on the databus during the NADS (Negative Address Data Strobe, on pin 39). If these four bits are left unused, the SC/MP can address only 4096 bytes of memory. This 4K memory is called a 'page'; the four MSB's can therefore be used to address 16 pages of memory. The SC/MP will not, of its own accord, 'turn to a new page'. This requires an explicit instruction in the program. When programming in BASIC, nothing could be simpler: for example, the 'statement' PAGE = PAGE + 1 causes the μP to proceed to the next page.

2



DMA and multiprocessing

The SC/MP has an extremely useful facility, absent in many other μPs: all the outputs used for writing into memory etc. employ so-called Tri-state logic. This means that they can not only be made 'hard' logic 1 or 0; a third state is also possible, where the outputs are 'floating' with a high output impedance. In this third state, the processor no longer has any effect on the address- and databus: as far as any other units are concerned it is no longer 'on-line'! Another microprocessor can then take over (multiprocessing), or a terminal can be used for immediate access to the memory. The latter option is normally referred to as DMA, for Direct Memory Access. It is not really the intention that the (human) operator should proceed to 'walk around inside the memory'; the main advantage of DMA is that it can save a considerable amount of (computer-) time when transferring large blocks of data from the memory to peripherals – floppy disc, for instance.

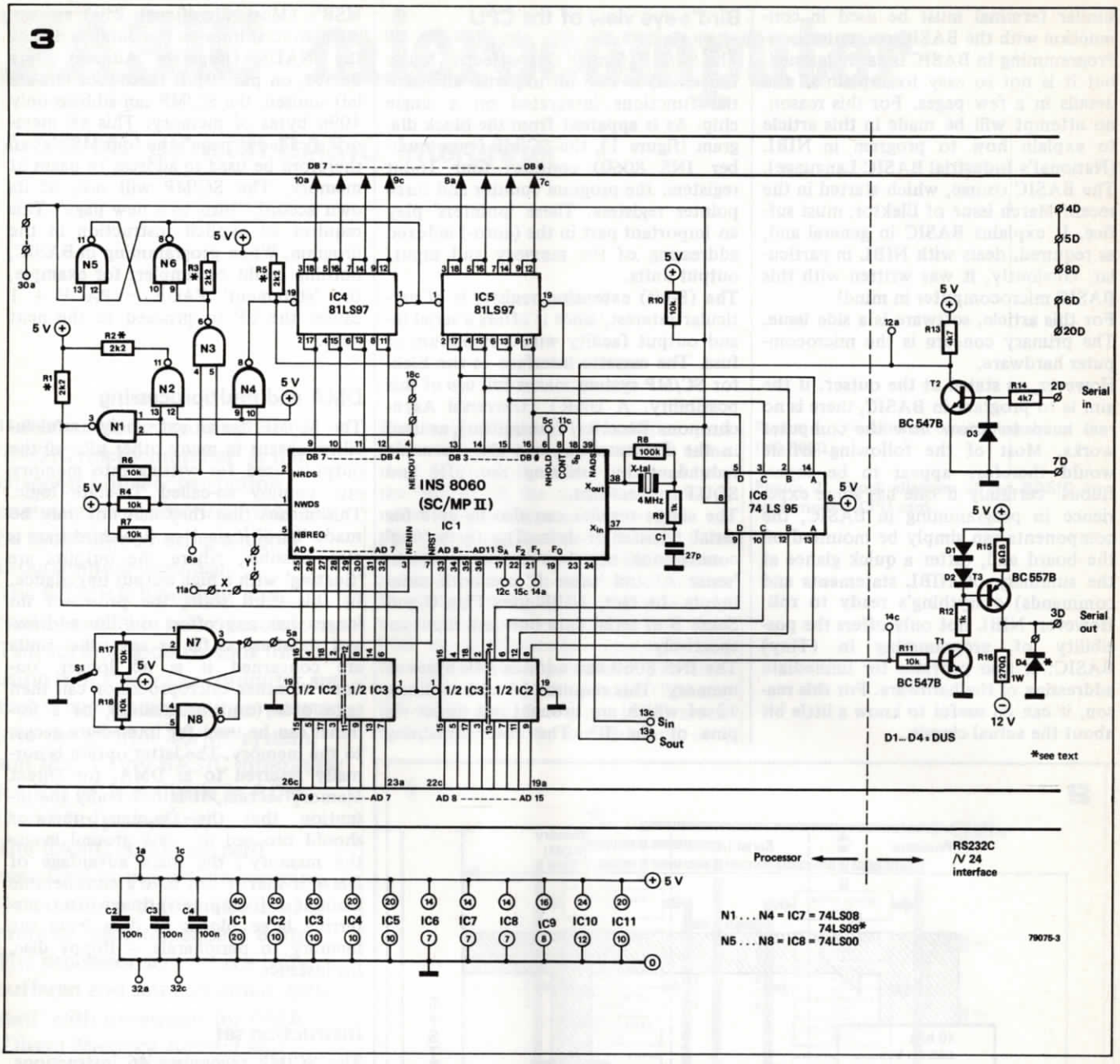
Instruction set

The SC/MP recognises 46 instructions, divided into nine groups; these instructions can be used in up to five different addressing modes. A detailed description of the complete instruction set, with all its variation capabilities, is way outside the scope of this article. It would require pages and pages (both magazine and human memory) and, moreover, it would be rather pointless. After all, this computer can be programmed in BASIC!
Detailed information is provided by the manufacturer, in the documentation listed at the end of this article. This not only explains the instruction set, but also contains full details on how to program in machine language and provides detailed technical information.

Block diagram

The BASIC card consists of three relatively independent sections. In fact, it doesn't really do justice to this design to call it a 'BASIC card', since its uses are by no means limited to a mere BASIC

3



computer. Right from the start, the intention was to produce a design with a minimum component count and maximum flexibility for different applications. The final result is all that we had hoped for.

The BASIC card is virtually a complete microcomputer: only the program memory must be added. The minimum memory requirement is 2048 bytes (sufficient for approximately sixty program lines), or half a 4K RAM card (EPS 9885). Obviously, any other 'memory' with the same capacity (or more) will do instead.

As illustrated in the block diagram (figure 2), the p.c. board contains three distinct sections. The most important of these is the processor section, consisting of the CPU and associated buffer circuits for the address bus, data bus and the main control signals. These buffer circuits make it possible for the CPU to work with extensive memory and peripheral systems. In short, this section is the ideal heart of a larger system.

A small but useful extension of the processor circuit is the RS232C/V24 interface. This section is connected to the processor's flag Ø output and sense B input, which are used as serial out- and input both in NIBL and in various other applications. For instance, this interface opens the possibility of connecting the unit direct to a terminal or teletype. The processor can itself take care of the necessary conversion from parallel to serial data format and vice versa – if the necessary software is available, that is. The saving in cost of hardware is well worth the additional processor-time required for this conversion.

The third and last section on the BASIC card is the Read-Only Memory. The complete NIBL-BASIC interpreter is supplied in a single so-called maxi-ROM. With its 32 Kbit (4096 bytes) memory capacity, this IC represented the absolute limit in Large Scale Integration (LSI) until quite recently, when a ROM with a 64 Kbit storage capacity was announced . . . It is to be expected that we will see ever larger ROMs appearing for some time to come.

The inputs to the ROM represent a negligable load on the address bus, so there is no need to add buffer stages at this point. The ROM outputs, however, have a very low drive capability; for this reason, an output buffer is required.

The advantage of the system outlined above is that the processor and ROM sections are fully independent units. Although both are mounted on the same p.c. board, their only means of communication is via the general system bus – the same bus that is used for communication with any other part of the system. This means that it is possible, for instance, to fully utilise the processor's capabilities in a particular application where the ROM is not required.

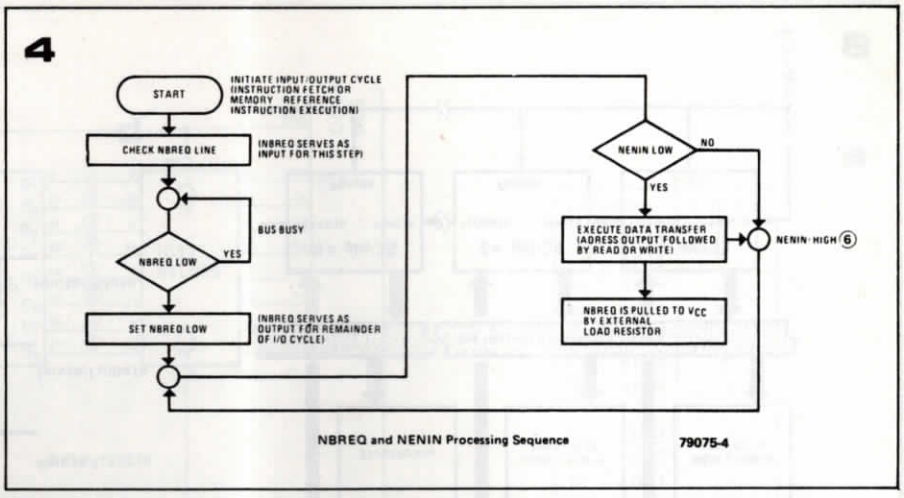
The circuits

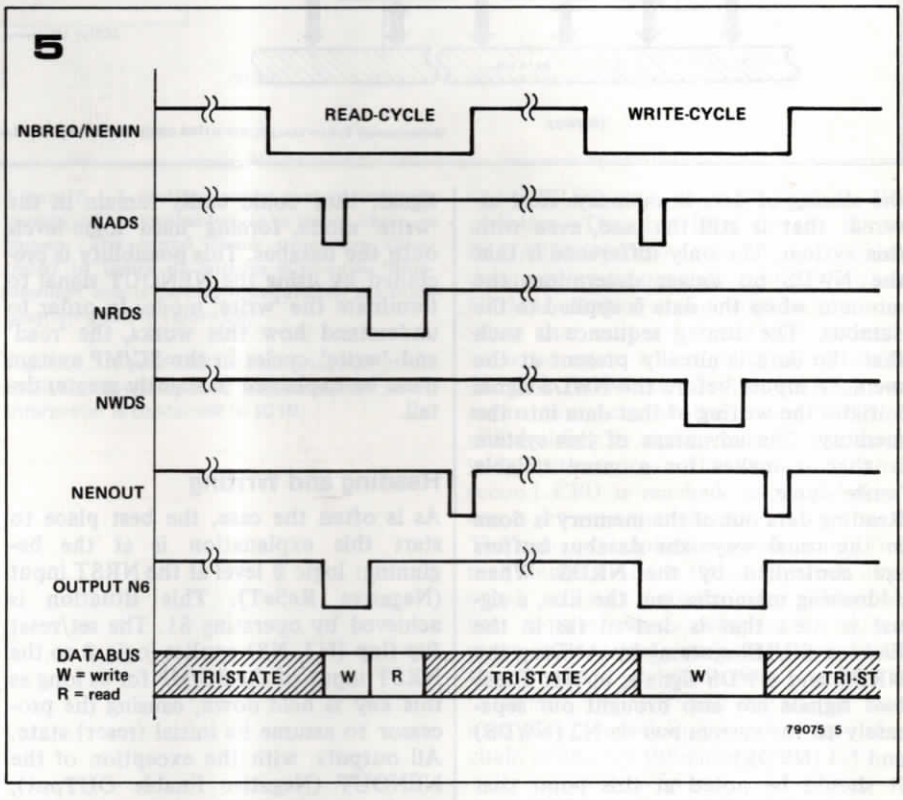
The circuits of the processor section and the associated RS232C/V24 interface are given in figure 3.

Figure 3. The processor section with input/output interface. This section can also be used as fully buffered CPU card.

Figure 4. Flow diagram of the initial check procedure that precedes each read or write cycle.

Figure 5. Pulse diagram of the main control signals within the BASIC microcomputer.





The *interface* does two jobs. In the first place, the TTL logic level at the flag Ø output of the processor must be converted to RS232C/V24 level. This means that logic 1 must be at least +5V and not more than +15V; similarly, logic Ø must be some level between −5V and −15V. As in the Elekterminal, the logic levels chosen in this circuit are +5V for logic 1 and −12V for logic Ø – for the simple reason that these levels correspond to common supply voltages. The fact that they are asymmetrical with respect to 0V has no effect on the realibility of data transfer.

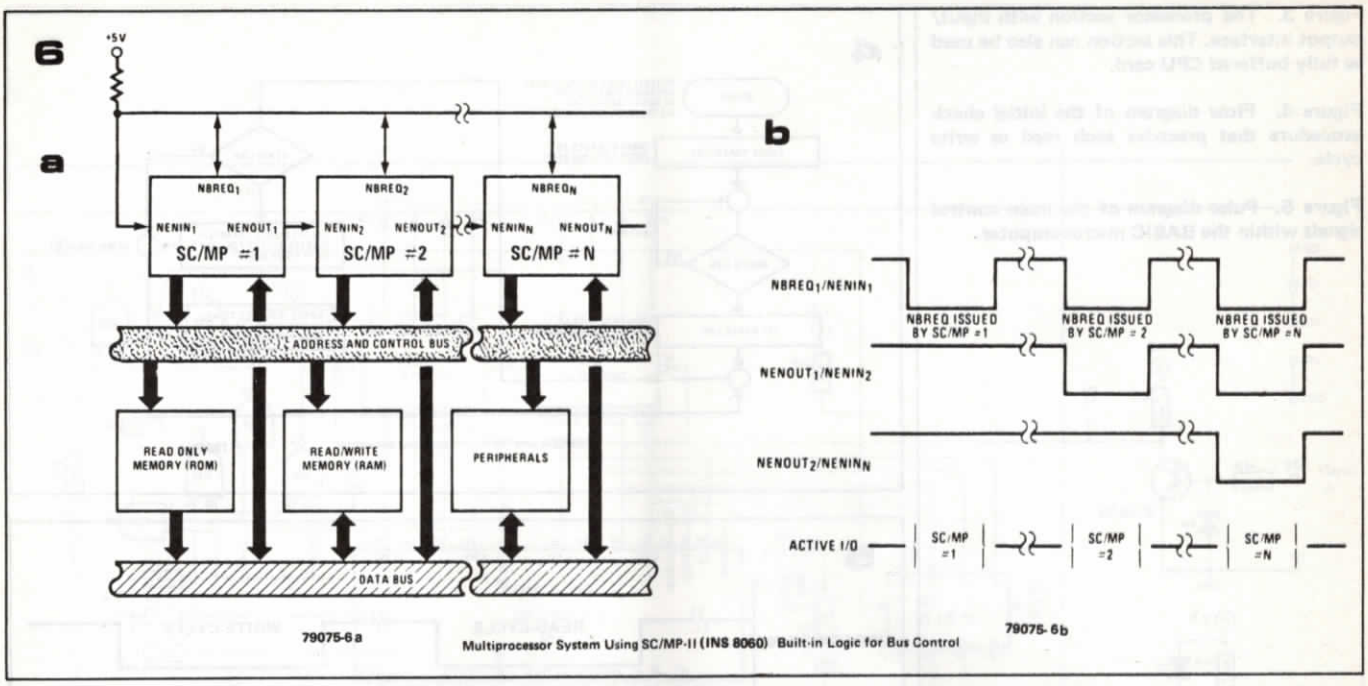
The flag Ø output of the processor drives transistor T1; in turn, this transistor switches a current source (consisting of T3 and a few resistors and diodes). The advantage of using a current source at the output is that it is short-circuit proof. Furthermore, it then becomes relatively easy to obtain a low output impedance, as required by the RS232C/V24 standard. Should a standard TTL level output be required for some application, it is sufficient to add one extra diode (D4). Logic Ø will then correspond to −0.6V (and logic 1 remains +5V); the interface circuit is then a short-circuit proof TTL output buffer. The second thing the interface must do is limit the logic levels at the sense B input of the processor. This is easily accomplished by T2 and D3; R14 limits the input current to a comfortable level.

The basic principles of the *processor section* have already been explained. However, some further explanation of the circuit is called for – particularly where the Direct Memory Access and multiprocessing facilities are concerned. The CPU, or Central Processor Unit, (IC1) receives clock pulses from an internal oscillator, with an external crystal to determine the frequency. From this clock signal, the NRDS (Negative Read Data Strobe) and NWDS (Negative Write Data Strobe) are derived.

The address and data outputs of the CPU have a limited drive capability. For this reason, the address bus is buffered by IC2 and IC3; similarly, IC4 and IC5 are included as databus buffer. These four ICs have an interesting feature: the input circuits incorporate PNP transistors in such a way that the input current is limited to 100 μA.

A shift register (IC6) is used as a buffer memory for the four highest address bits (MSBs). Using the 74LS95 at this point has the advantage that the NADS (Negative Address Strobe) can be used, without need for an inverter, to read in the four MSBs to the register.

The NADS is also used to control the databus buffers, in conjunction with the NRDS and NENOUT signals (Negative Read Data Strobe and Negative ENable OUTput, respectively). This combination may seem rather strange to those of our readers who have previously studied the Elektor SC/MP system. One would expect that the NWDS (Negative Write Data Strobe) would also be involved in the control of the databus buffers. After all, the NWDS is supposed to control



Multiprocessor System Using SC/MP-II (INS 8060) Built-in Logic for Bus Control

the storing of data in memory. Rest assured: that is still the case, even with this system. The only difference is that the NWDS no longer determines the moment when the data is applied to the databus. The timing sequence is such that the data is already present at the memory inputs before the NWDS signal initiates the writing of that data into the memory. The advantage of this system is that it makes for a more reliable 'write' cycle.

Reading data out of the memory is done in the usual way: the databus buffers are controlled by the NRDS. When addressing memories and the like, a signal is used that is derived (as in the Elektor SC/MP system) by ANDing the NRDS and NWDS signals, in N1. These two signals are also brought out separately to the system bus via N2 (NWDS) and N4 (NRDS).

It should be noted at this point that both the 74(LS)08 and the 74(LS)09 can be used as output buffers; the 09 is only required in DMA or multiprocessor systems. The reason for this is that the 74(LS)09 has so-called open-collector outputs, so that several of these ICs can be connected in parallel (with one common set of pull-up resistors) without 'biting' each other. If only a simple system is contemplated, with one CPU and without DMA, the 74(LS)08 can be used instead; the pull-up resistors R1, R2, R3 and R5 can then be omitted.

There is a further reason for controlling the databus buffers by means of a combination of the NADS, NRDS and NENOUT signals – quite apart from the increase in speed and reliability when writing data into the memory. In systems where the SC/MP is used without output buffers, DMA and multiprocessing present few problems, since its tri-state outputs can easily be set in the 'floating' mode. However, in the buffered system described here, the output buffers are not controlled by the NWDS signal; they could easily remain in the 'write' mode, forcing 'hard' logic levels onto the databus. This possibility is precluded by using the NENOUT signal to terminate the 'write' mode. In order to understand how this works, the 'read' and 'write' cycles in the SC/MP system must be explained in slightly greater detail.

Reading and Writing

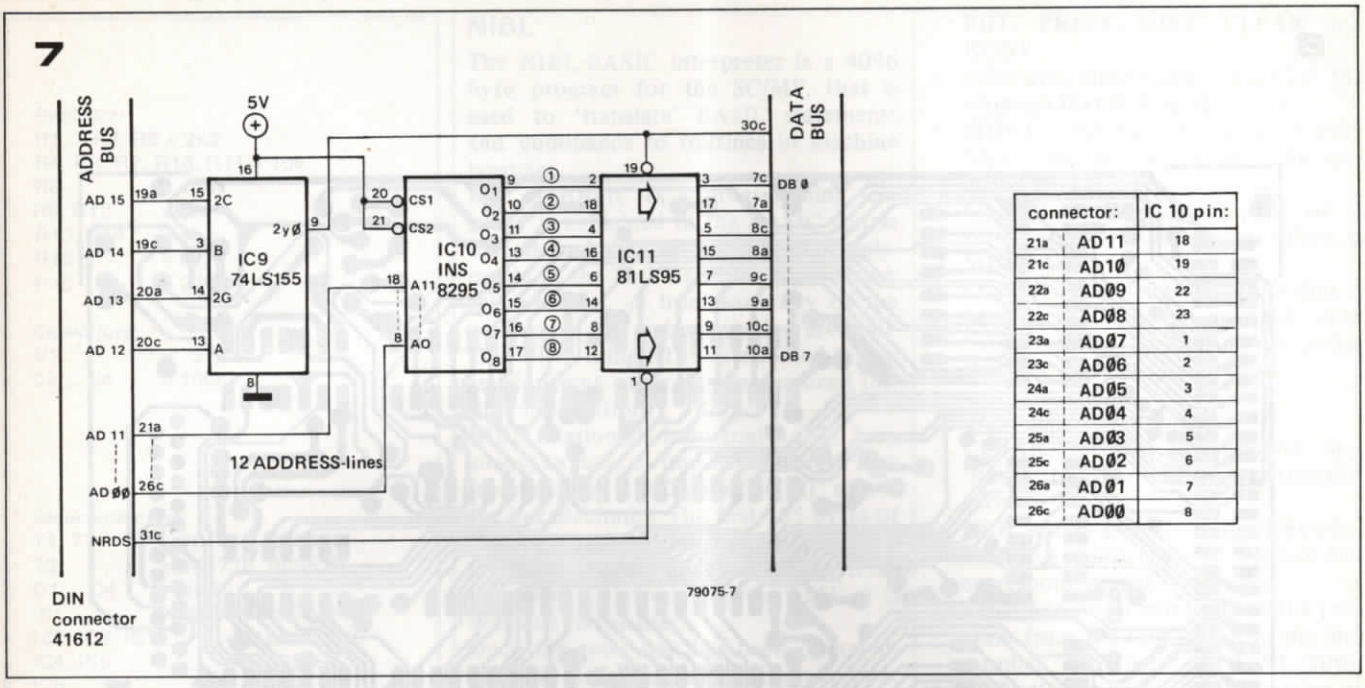
As is often the case, the best place to start this explanation is at the beginning: logic Ø level at the NRST input (Negative ReSeT). This situation is achieved by operating S1. The set/reset flip-flop (N7, N8) applies logic Ø to the NRST input of the SC/MP for as long as this key is held down, causing the processor to assume its initial (reset) state. All outputs, with the exception of the NENOUT (Negative Enable OUTput), are then in the floating (tri-state) mode. The pull-up resistors R4, R6 and R10 hold the NWDS, NRDS and NADS outputs at a defined logic level (logic 1), so that nothing untoward can occur...

When S1 is released, the SC/MP will check for a logic Ø level at the NBREQ and NENIN inputs (Negative Bus REQuest and Negative ENable INput, respectively). Figure 4 illustrates this procedure. In a basic single-processor system without DMA facility, R7 will always pull the NBREQ *input* high. As soon as the processor detects this logic 1 level, it will proceed to use the same connection as NBREQ *output*. Since the logic 1 level signifies that no other part of the system is using the bus at present (obviously, in a simple system without DMA this is always the case, since there is only the one CPU), the processor proceeds to stake its claim to the bus by making the NBREQ output logic Ø. Having done this, it tests the logic level at the NENIN input. Since this input is connected to the NBREQ output (by means of the link shown as a dotted line in figure 3) it will also be at logic Ø level. With both necessary conditions now fulfilled, the SC/MP will proceed to fetch its first instruction.

This first 'read' cycle is illustrated in figure 5. Shortly after the NBREQ output goes to logic Ø, the NADS signal appears. The shift register (IC6 in figure 3) takes this as its cue to store the four MSBs of the address; simultaneously flip-flop N5/N6 is set, switching the databus buffers into the write mode. However, when the NRDS signal appears it will reset this flip-flop and switch the databus buffers into the 'read' mode. The read cycle is terminated by a brief pulse on the NENOUT connection. In this case, the NENOUT pulse has no effect on the buffers – they had already been switched back to the floating state at the end of the NRDS pulse, as shown in figure 5.

The sequence of operations during the write cycle is similar, with one major difference: the output of N6 holds the databus buffers in the write mode for a much longer period. In fact, the NWDS signal falls well inside this period. The result is that the data to be stored are present at the memory input well before the NWDS signal appears, and remain there for a short time after this pulse is terminated. Finally, the NENOUT pulse causes the buffers to revert to the floating state.

The advantage of the system outlined above will become apparent from a closer look at the multiprocessor facilities that the SC/MP has to offer. Figure 6a gives a rough outline of a microcomputer system in which several SC/MPs are used. The first of these is connected in the same way as in the single-processor system described so far. For all the following SC/MPs, however, there is a minor modification to the circuit: the NENIN input of each is connected to the NENOUT of its predecessor in the



connector:	IC 10 pin:
21a AD11	18
21c ADlØ	19
22a ADØ9	22
22c ADØ8	23
23a ADØ7	1
23c ADØ6	2
24a ADØ5	3
24c ADØ4	4
25a ADØ3	5
25c ADØ2	6
26a ADØ1	7
26c ADØØ	8

chain.
After the initial reset, the situation for the first processor is exactly as outlined above. All other processors, however, must wait for their turn: as long as one CPU is using the bus, all others must keep off. The principle is clear from figure 4: each time a CPU wants to 'get on the bus', it will first check the logic level on its NBREQ *input*. A logic ϕ at this point signifies that one of the other SC/MPs is performing a read or write cycle at that moment, so that the bus is busy.
The interplay between the various CPUs is further determined by the NENIN and NENOUT signals. The rules of play are as follows. When a processor is using the system bus, its NENOUT is always at logic 1; if it is not on the bus, its NENOUT assumes the same logic level as that present at its NENIN. Bearing in mind that the NENIN must be at logic Ø before the actual read or write cycle can be initiated, the sequence of events is as follows.
Assume that a CPU somewhere in the middle of the chain wants to store some data in memory. Testing the NBREQ line, it discovers that this is at logic Ø and so it is forced to sit back and wait its turn. As soon as the NBREQ line goes high, the CPU quickly jumps in and pulls this line low again, staking its claim. This pulls the NENIN of the first SC/MP low and, assuming that this CPU is not interested in the busses, its NENOUT will follow – passing the logic Ø level on to number 2. The low NENOUT/NENIN level is passed down the chain in this way until it reaches the CPU that requested entry to the bus.
This unit takes this signal as a sign of approval, maintains its own NENOUT connection at a logic 1 level and proceeds to store the data.
It is, of course, conceivable that two CPUs jump in simultaneously when one other goes off the line – both pulling the NBREQ line down to logic Ø. No problem. The low level on the NENOUT/NENIN connections is passed down the chain until the first of the two CPUs is reached – and stops here! Only when that unit is finished with its read or write cycle will it produce a logic Ø level at its NENOUT (the NBREQ remains low because the second CPU is still holding it down); this signal then goes further down the chain until the second CPU is reached, and only then can it get onto the busses.
The same principles are involved in a Direct Memory Access (DMA) system: any other units (a terminal, for instance) that require direct access to the busses must include logic gating that provides the same relationships between 'NBREQ', 'NENIN' and 'NENOUT' signals. They can then be linked into the chain in exactly the same way.

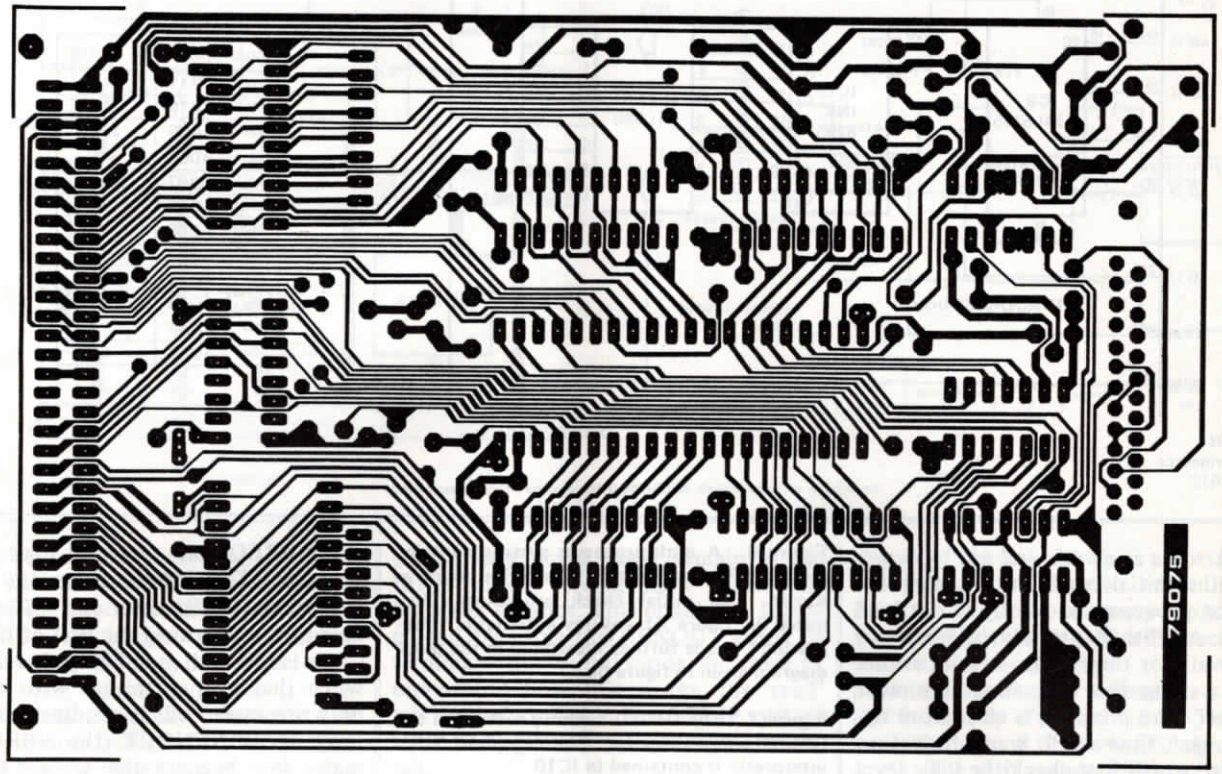
Figure 6. A multiprocessor system contains several CPUs connected in a series chain as shown. The initial check procedure (illustrated in figure 4) ensures automatic 'time-sharing'; this is further illustrated in the pulse diagram given in figure 6b.

Figure 7. The (ROM) memory section of the BASIC microcomputer. The complete NIBL interpreter is contained in IC10.

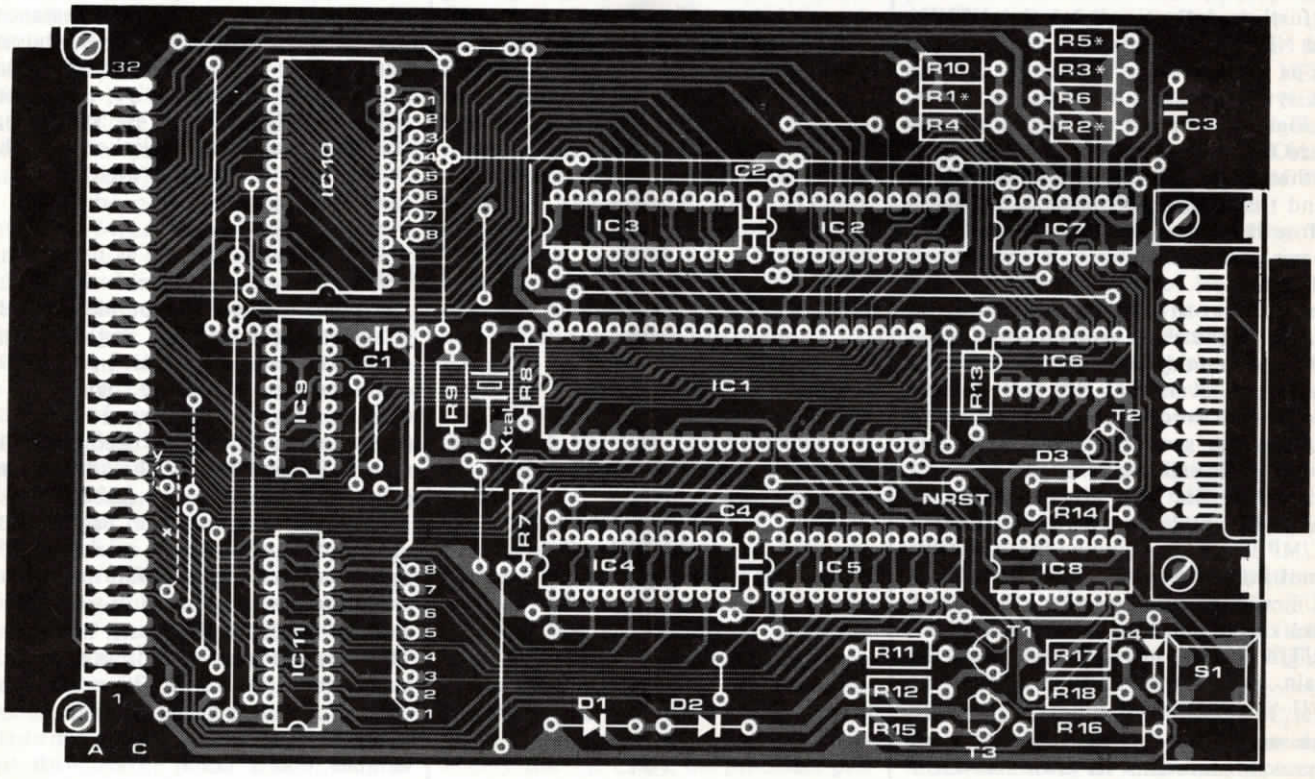
Memory

As stated earlier, the complete BASIC interpreter is stored in a single IC. This makes the memory circuit in the NIBL computer simplicity itself (figure 7).
One integrated circuit, a 74LS155 (IC9), is used as address decoder. It detects the four MSBs of the address, and it is wired in such a way that the NIBL-ROM (IC10) is located on page 0 in the memory. The remaining twelve address lines go direct to the ROM; the outputs from the memory are buffered (IC11) and applied to the databus.
The output from the address decoder is also brought out to pin 30c of the edge connector. In the Elektor SC/MP system, this line is used for control of the databus buffer (EPS 9972). With this extra connection, the BASIC microcomputer is suitable for use as a replacement for the original CPU card in an existing Elektor SC/MP system with or without databus buffering.

8


79075

9


32
IC10
IC9
IC11
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
C1
C2
C3
C4
R1*
R2*
R3*
R4
R5*
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
Xtal
NRST
D1
D2
D3
D4
T1
T2
T3
S1
A
C

Resistors:
R1 ... R3, R5 = 2k2
R4, R6, R7, R10, R11 = 10k
R8 = 100k
R9, R12 = 1k
R13, R14 = 4k7
R15 = 6Ω8
R16 = 270Ω

Capacitors:
C1 = 27p
C2 ... C4 = 100n

Semiconductors:
T1, T2 = BC107B, BC547B
T3 = BC177B, BC557B
D1 ... D4 = DUS
IC1 = INS8060 (SC/MP II)
IC2, IC3, IC11 = 81LS95
IC4, IC5 = 81LS97
IC6 = 74LS95
IC7 = 74LS08
IC8 = 74LS00
IC9 = 74LS155
IC10 = INS8295N

Sundries:
1 x 64-pin connector DIN41612 (male)
1 x 25-pin connector 90° MIN D (female)

Figure 8. Printed circuit board for the complete BASIC microcomputer (EPS 79075).

Figure 9. Component layout for the p.c. board. Note that some wire links are only required in certain applications.

NIBL

The NIBL-BASIC interpreter is a 4096 byte program for the SC/MP, that is used to 'translate' BASIC statements and commands to routines in machine language.
Use of BASIC as a programming language is explained in the BASIC course that is included as a series of supplements in the Elektor issues from March this year on. A brief summary of the commands and statements that are available when using NIBL is included in this article; some other details also need further clarification.
NIBL (National's Industrial BASIC Language) expects to find RAM storage area from address 1000_H on (the 'H' stands for hexadecimal). The first 285 bytes of this memory are used by NIBL for storing data. All remaining memory from there on (i.e. from address $111E_H$) is available to the user.
Once the reset key has been operated, NIBL is ready to receive program lines. Single statements can be entered without a program line number, if required; in that case they will be carried out immediately (so-called 'direct' or 'immediate' mode). This can be particularly useful when testing a (section of) program. In the direct mode, variables can be given certain values so that the program can be started from a well-defined initial situation.
A program can be entered in two ways: from the keyboard of a terminal, or by means of a paper-tape reader or some similar device. In the latter case, the reader-relay should be controlled by the flag 1 output. However, relatively few people have access to a paper-tape reader and the associated 'hole-puncher', so a tape or cassette recorder will normally be used instead. For this, a cassette interface is required, as well as some additional 'software'.
The NIBL statements and commands are based on Tiny BASIC. However, NIBL contains several additional features. The most important of these are the DO ... UNTIL routine, which is derived from 'PASCAL', and the 'Indirect Operator'. The latter replaces the PEEK and POKE statements in other BASIC dialects: it can be used for direct addressing of the memory and I/O (Input/Output devices). Of lesser importance – although it can be useful – is the possibility of using so-called 'text variables'.

NIBL statements and commands

Program entry (program lines)
- a line without a line number is carried out immediately.
- a line with a line number is inserted in the program in the correct (numerical) position.
- line numbers from 0 to 32767 = $2^{15} - 1$) can be used.
- spaces are not permitted within 'Key words' (LET, IF, THEN, GOTO, GOSUB, GO, TO, SUB, RETURN, INPUT, PRINT, LIST, CLEAR and RUN).
- otherwise, spaces can be added in the program text as desired.
- SHIFT/0 (or back-arrow on a teletype) deletes the letter that was typed in last.
- CONTROL/H (or backspace on a video terminal) has the same effect as SHIFT/O.
- CONTROL/U deletes the line that is being typed in at that moment, without affecting the data stored under that line number in memory.

Program control (commands)
- CLEAR returns all variables and 'stacks' to their initial state (usually zero).
- NEW erases page 1 in the memory.
- NEW n (where $2 \leqslant n \leqslant 7$) erases the corresponding page in memory.
- LIST initiates a print-out of the program from the first line *or* of the line number specified (e.g. LIST 2ØØ).
- RUN starts the program (starting at the first line).
- GOTO n (where $0 \leqslant n \leqslant 32767$) starts the program at the line number specified, without resetting the variables and stacks.

Variables, constants, operators
- 26 variables can be used: the letters A to Z.
- all operations ('expressions') are carried out using 16-bit 'two's complement' numbers.
- arithmetical operators: +, –, *, /.
- comparison symbols: <, >, =, <=, >=, <>.
- logic operators: AND, OR, NOT.
- decimal constants must remain within the range from –32767 to +32767.
- hexadecimal constants are recognised as such when preceded by the symbol #. Not more than four digits (16 bits) are permitted.
- program lines may contain more than one statement, provided the statements are separated by a colon (:).

Functions
- RND (a, b) generates a random number within the range from a to b.
- MOD (a, b) gives the remainder after the division a/b.
- STAT calls up the contents of the status register in the SC/MP.
- PAGE calls up the number of the page in memory that is currently in use.
- TOP calls up the upper boundary of the NIBL program, as a decimal address.

INPUT/OUTPUT statements
- INPUT X
- INPUT X, Y, Z
- PRINT "THIS IS NIBL"
- PRINT "F = ", M * A
- PRINT "SKIP", X, "PAGES";

Note that the semicolon (;) suppresses the automatic CR/LF (Carriage Return/Line Feed) after a print statement.

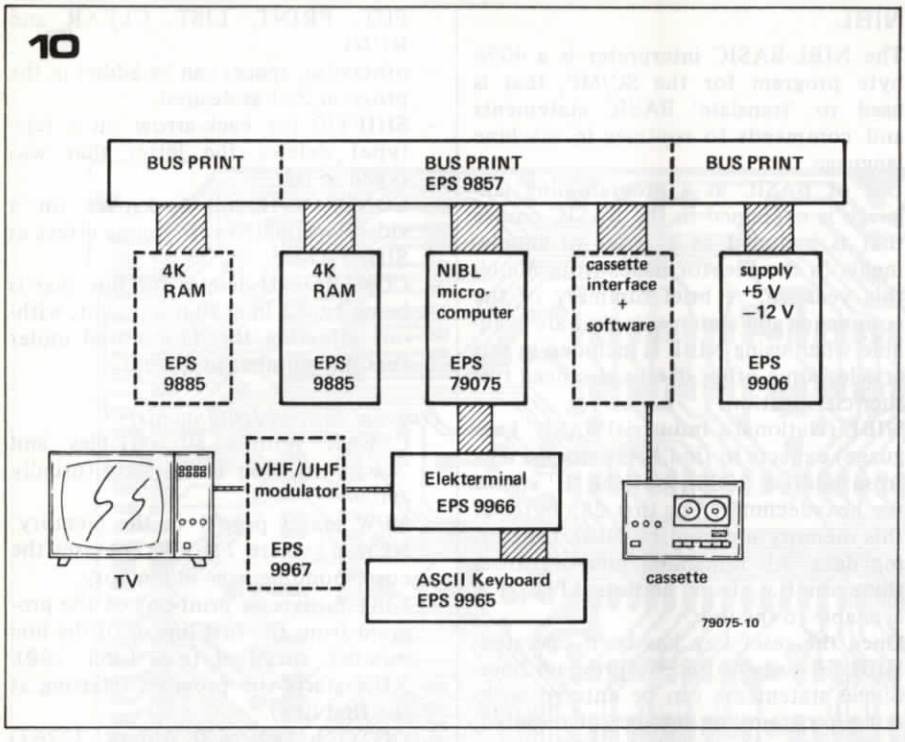


Figure 10. Block diagram of a complete BASIC microcomputer system with extension facilities, based on the main board described in this article.

Assignment statements
- LET X = 7
- E = I * R
- STAT = # 70
- PAGE = PAGE + 1
- LET @ A = 255
- @ (T + 36) = # FF
- B = @ (TOP + 5)

Control statements
- GO TO 15 or GOTO 15
- GOTO X + 5
- GO SUB 100 or GOSUB 100
- RETURN
- IF X + Y > # 1A GOTO 15
- IF A = B LET A = B–C
- FOR I = 10 TO Ø STEP –2
- NEXT I
- FOR K = 1 TO 5
- DO: X = X + 1: UNTIL (X = 10) OR (@X = 13)

Indirect operator
- the @ symbol can be used for immediate addressing of a location in memory; for instance: V = # 2000: LET @ V = 100 results in the decimal number 100 being stored at memory location 2000_H. Similarly, LET W = @ V gives the variable W the value stored in memory location V.

String handling (text facilities)
- $A = "ONE LINE OF TEXT"
- PRINT $T, $(TOP + 72)
- INPUT $(U + 20)
- U = $ (TOP + 2 * 36)

Sundries
- LINK (address): The program is continued in machine language, from the address indicated. The address must be given as a decimal number.
- REM offers the possibility of adding explanatory text (comments, reminders) to the program.
- END: this statement is used to conclude a program, and to add 'break points'.

Error indications

As soon as a program is started, error indications may appear as a result of incorrect or incomplete use of NIBL. The general error indication format is as follows:
... ERROR AT ... .
The first four characters indicate the type of error; the final characters (up to five) give the line number. For example, incorrect use of a statement at line number 4500 would result in the print-out:
STMT ERROR AT 4500
Error indications in NIBL all use 'words' of up to four letters. The following indications are possible:

AREA The memory space available on the chosen page is exceded.
CHAR Redundant or incorrect characters in or following a statement.
DIVØ Division by zero.
END" No quotation marks after text to be printed.
FOR FOR is not followed by NEXT.
NEST Subroutine possibilities are exceded.
NEXT NEXT used without FOR.
NOGO The line number specified in a GOTO or GOSUB statement does not exist.
RTRN RETURN was not preceded by GOSUB.
SNTX Incorrect syntax.
STMT Incorrect use of a statement.
UNTL UNTIL is used without DO.
VALU Incorrect constant or number outside the range.

The p.c. board

The complete circuit can be mounted on the p.c. board shown in figure 8. The board size corresponds to that used in the Elektor SC/MP system: it is Eurocard format, and the edge connector corresponds to the system bus. A second connector is included on the other end of the board; this is intended for connecting a teletype or videoterminal according to the RS232C/V24 standard. This 25-pin connector is variously referred to as a 'female modem connector' and as a 'D connector'.
Where possible, the component layout shown in figure 9 indicates which wire links should be included for a particular application. Reference to figure 3 should further clarify matters.

A complete microcomputer

The unit described here obviously requires a few additional circuits to be fully operational. A minimum system would consist of one bus board, a power supply board, one 4K RAM card and the BASIC computer card described in this article. The system can be extended by adding up to six memory cards.
An obvious choice for in- and output unit is the Elekterminal. The complete Elektor BASIC microcomputer would then consist of the units shown in figure 10.

Lit.
1. *SC/MP data sheet, pub. no. 420305227-001A*
2. *SC/MP instruction guide, pub. no. 4200110A.*
3. *SC/MP technical description, pub. no. 4200079A.*
4. *SC/MP microprocessor applications handbook, pub. no. 420305239-001A.*
5. *SC/MP programming and assembler manual, pub. no. 4200094B.*
6. *Elektor E31 ... E36 (November 1977 ... April 1978).*