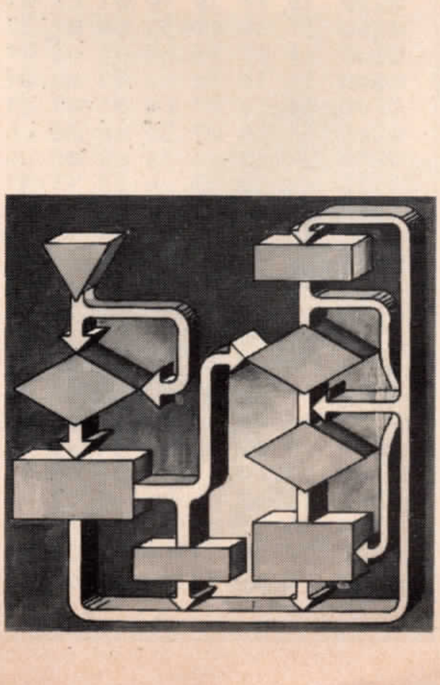
experimenting with the SC/MP (5)

Elbug

Elbug is the monitor software for the Elektor SC/MP microcomputer.
By monitor software is meant a programme, usually resident in ROM, which provides the user with the control functions needed to operate the system satisfactorily. A monitor programme typically contains a number of routines which perform such chores as programme loading, debugging and general housekeeping.
As has already been emphasised, before Elbug can be used, it is essential that all the system software which has been published so far should function without fault. A further preliminary to using Elbug is the transfer of the CPU card from page '1xxx' to page 'Øxxx' of memory. This is done by changing the position of the appropriate wire links on the memory extension board (see Elektor 33) to 'CPU Øxxx'. The page address of the RAM I/O (as long as it remains in use) then becomes 1ØØØ.
The Elbug programme is fairly long, and occupies all the 1½ k of memory which was reserved for this purpose. The memory hardware consists of three EPROMs, two of which (IC3 and IC2) are mounted on the CPU card, with the third (IC14) on the memory extension card.
The listing for the monitor programme is given in a condensed form in tables 1, 2 and 3. Only the machine code is listed. The first column in each of the tables consists of addresses, whilst all the remaining figures represent data. For example, the data byte Ø8 is contained in the location with address ØØØØ. The next figure, i.e. C4, is the contents of the following address, i.e. ØØØ1. The programme is written into the three EPROMs such that tables 1 and 2 represent the contents of IC3 and IC2 respectively on the CPU card, whilst table 3 is written into IC14 on the memory extension card.
In order to write data into an EPROM a special PROM-programmer is required, and, unfortunately, these are rather expensive. There are firms who operate a PROM-programming service, however that of course involves the prospective user recording the programme in question on papertape, cassette or PROM before it can be sent. For this reason it is the intention to make pre-programmed

This part of the series introduces 'Elbug', the monitor programme for the SC/MP system. It takes a close look at the various command routines provided by Elbug, as well as examining the software used to control the cassette interface.

H. Huschitt



EPROMs containing Elbug available direct from Elektor.
In addition to the three PROMs, Elbug also requires a section of RAM (STACK) in which to store variables. The section of memory from address ØFC9 up to and including ØFFF is reserved for this purpose. The rest of RAM present on the CPU and memory extension boards (address ØCØØ to ØFC9) and the ¼ k RAM on the RAM I/O card (address 1ØØØ up to and including 1ØFF) are left available for the user's programme. This amounts to a total of more than 1 k of RAM, which should prove more than sufficient for the 'apprentice' programmer. If desired, the memory capacity of the system can be expanded by incorporating one or more of the 4 k-RAM cards, details of which are contained elsewhere in this issue. However, before one sets about considerably expanding the memory of the system, one should ensure that the supply is capable of delivering sufficient current. A suitable power supply is also described elsewhere in this issue.

How to use Elbug

With the arrival of Elbug, the SC/MP system has acquired 'intelligence'. However before starting to use Elbug one must first know which command key initiates which control software routine and what the significance is of the various statements Elbug outputs on the displays.
Elbug splits the displays into three separate formats:

- Displays Ø and 1 (those furthest to the right) are reserved for data (data field)
- Displays 2 to 5 (the 4 middle displays) indicate addresses (address field)
- Display 6 and 7 (the 2 left-hand displays) indicate the instructions (command field)

The command keys enter the following control routines:

Key C7 (code FØ): RUN
Key C6 (code EØ): MODIFY
Key C5 (code DØ): SUBTRACTION
Key C4 (code CØ): CASSETTE
Key C3 (code BØ): BLOCK-TRANSFER
Key C2 (code AØ): CPU REGISTER
Key C1 (code 9Ø): DOWN
Key CØ (code 8Ø): UP

The UP- and DOWN keys are not in fact genuine command keys, but rather suffix keys, as will become clear further on.

MODIFY

Once Elbug has been installed, and the power turned on, pressing the Halt/Reset key should result in the word ' . . Elbug ' appearing on the displays. The two decimal points which should also light up indicate that the programme is waiting for one of the command keys to be pressed.

Once the MODIFY key has been pressed the text 'MO . . . . ' will appear on the since it enables the user to examine the contents of any location in memory, and if so desired, to directly alter the contents of that location (assuming it is RAM of course).

Once the MODIFY key has been pressed the text 'MO . . . . ' will appear on the displays, indicating that the programme is waiting for an address to be entered via the data keys (keys Ø . . . F). When the last (hexadecimal) digit of the address has been entered, the contents of that address will also appear on the displays. To give an example; if address ØCC9 is selected, the contents of which is A1, the displays will now read 'MOOCC9A1'. The user now has a number of different possibilities:

* The contents of the addressed location can be altered by using the data keys to enter the desired data-byte. When the first key is pressed the corresponding digit appears on display 1, the code generated by the second key appears on display Ø. It goes without saying that the above procedure is invalid if the address in question is located in PROM or does not reference memory at all. In the latter case the data field will display 'FF', whilst (as their name suggests) the contents of a location in PROM can only be read, and hence cannot be altered by means of the data keys.
* If the UP-key is pressed, the address and contents of the following location will appear on the displays. The contents can again be altered in the above-mentioned fashion.

If, whilst in the course of entering new data, a mistake occurs and the wrong key is pressed, this can be rectified simply by entering the correct byte immediately after the false data, since the latter is then lost.

* The contents of the previous address can also be examined by pressing the DOWN-key.
* The MODIFY routine can only be exited from by means of the NRST key. The word ' . . Elbug ' will then reappear on the displays.

Within the MODIFY routine it is only possible to access addresses on one page, i.e. if the UP-key is pressed at address 4FFF the programme simply returns to the top of the same page and address 4ØØØ appears. In order to 'turn the page' one must press the NRST, re-enter the MODIFY routine, then enter the address of the next page (5ØØØ).

Table 1.

0000	08	c4	15	c8	f1	c4	e0	c8	f7	c4	0f	c8	f2	c4	00	c8
0010	e9	c8	e9	90	3d	c0	e9	31	c0	e5	35	c5	01	c8	de	c5
0020	01	c8	db	c5	01	37	c5	01	33	c5	01	36	c5	01	32	c5
0030	01	c8	c4	c5	01	c8	c1	c5	01	07	c5	01	01	c5	01	c8
0040	b8	c0	b4	35	c8	b9	c0	b0	31	c8	b5	b8	ad	c0	aa	3f
0050	90	04	90	4d	90	bf	c8	a1	c0	a6	33	c8	9b	c0	a0	37
0060	c8	95	c4	ff	31	cf	fc	c4	0f	35	cf	ff	01	cb	03	06
0070	cb	02	c1	f9	cb	04	32	cf	ff	36	cf	ff	c1	f8	cf	ff
0080	c1	f7	cf	ff	c1	fe	cf	ff	c1	fd	cf	ff	37	c9	ff	c1
0090	fe	33	c9	00	a9	fa	e1	fb	9c	04	c4	ff	c9	fc	3f	90
00a0	b3	c4	00	31	c4	07	35	c4	e0	32	c4	0f	36	c4	2f	33
00b0	c4	01	37	c4	08	ca	0b	c7	01	cd	01	ba	0b	9c	f8	c4
00c0	0a	ca	1d	c4	02	ca	1c	c4	00	37	c4	55	33	3f	c4	80
00d0	cd	fd	cd	ff	cd	ff	cd	ff	c4	00	cd	ff	c2	08	01	40
00e0	e4	e0	98	53	40	e4	f0	9c	07	c4	01	37	c4	a0	33	3f
00f0	40	e4	d0	9c	07	c4	03	37	c4	ea	33	3f	40	e4	c0	9c
0100	07	c4	02	37	c4	f1	33	3f	40	e4	b0	9c	07	c4	05	37
0110	c4	49	33	3f	40	e4	a0	9c	88	c4	04	37	c4	35	33	3f
0120	06	5b	4f	66	6d	7d	07	7f	6f	77	7c	58	5e	79	71	00
0130	3d	1c	7c	38	79	80	80	c4	5c	c9	05	c4	54	c9	06	c4
0140	3e	ca	1d	3f	c2	01	33	c2	02	37	c3	00	ca	00	c4	a0
0150	ca	1d	c4	00	37	c4	55	33	3f	c4	0a	ca	1d	3f	c2	01
0160	33	c2	02	37	c2	08	e4	80	98	0a	e4	80	e4	90	9c	0e
0170	c7	ff	90	02	c7	01	33	ca	01	37	ca	02	90	c6	c2	07
0180	c9	00	c4	00	c9	ff	c2	09	1e	1e	1e	1e	01	c4	00	37
0190	c4	55	33	3f	c2	01	33	c2	02	37	c2	09	58	cb	00	90
01a0	a3	c4	50	c9	06	c4	1c	c9	05	c4	3e	ca	1d	c4	00	37
01b0	c4	55	33	3f	c4	0a	ca	1d	3f	c2	01	33	c2	02	37	c7
01c0	ff	c4	50	c9	00	c4	1c	c9	ff	3f	c4	0f	37	c4	ff	33
01d0	3f	c2	15	1c	ca	14	c4	ff	01	19	40	94	02	90	f7	c4
01e0	ff	01	c2	14	ca	0a	ba	0a	9c	fc	c4	08	ca	08	c2	15
01f0	ca	09	c4	16	8f	00	ba	09	9c	fc	19	ba	08	9c	ef	c2

Table 2.

0200	15	ca	09	ba	09	9c	fc	40	3f	90	c6	c4	14	33	c4	00
0210	37	c4	01	31	c4	07	35	c4	e0	32	c4	0f	36	c1	08	94
0220	fc	8f	1e	c1	08	ca	08	d4	0f	ca	09	01	c1	08	94	02
0230	90	fa	8f	1e	c4	1f	31	c4	01	35	c1	80	ca	07	3f	c4
0240	06	31	c4	07	35	c4	e7	32	c4	0f	36	c4	04	ca	f9	c4
0250	55	33	c4	00	37	c4	0a	cb	a8	c4	02	cb	a7	3f	c4	e0
0260	33	c4	0f	37	c3	07	cd	ff	c4	00	c9	ff	c9	fe	c9	fd
0270	c9	fc	c9	fb	c3	09	ce	ff	bb	00	9c	d3	c4	80	c9	ff
0280	c9	fe	c3	06	1e	1e	1e	1e	01	c3	05	58	cb	02	c3	04
0290	1e	1e	1e	1e	01	c3	03	58	cb	01	c4	00	37	c4	14	33
02a0	3f	c4	e0	33	c4	0f	37	c4	e0	32	c4	0f	36	c4	e3	31
02b0	c4	0f	35	c4	03	cb	0f	c2	00	d4	0f	cd	01	c6	01	1c
02c0	1c	1c	1c	cd	01	bb	0f	9c	ee	c4	1f	31	c4	01	35	c4
02d0	06	cb	0f	c6	01	01	c1	80	ca	05	bb	0f	9c	f5	c4	00
02e0	31	c4	07	35	c4	06	cb	0f	c6	01	cd	01	bb	0f	9c	f8
02f0	90	a8	c4	39	c9	06	c4	5f	c9	05	01	19	c4	ff	ca	00
0300	c4	00	37	c4	55	33	3f	c4	5f	c9	00	c4	5e	c9	ff	c2
0310	08	e4	e0	9c	1e	c4	54	c9	00	c4	5c	c9	ff	c4	3e	ca
0320	1d	3f	c2	01	ca	15	c4	0a	ca	1d	3f	c4	5f	c9	00	c4
0330	5e	c9	ff	c2	08	e4	80	98	2c	c4	3e	ca	1d	3f	c2	01
0340	ca	0b	c2	02	ca	0c	3f	c4	0a	ca	1d	3f	c2	08	e4	80
0350	9c	04	ca	00	90	0f	e4	80	e4	90	98	02	90	50	c4	04
0360	37	c4	e3	33	3f	c4	1c	c9	00	c4	73	c9	ff	c4	d0	33
0370	c4	01	37	c2	00	98	0e	3f	ca	0c	3f	ca	0b	3f	ca	02
0380	3f	ca	01	90	04	3f	3f	3f	3f	c4	20	ca	05	c4	00	ca
0390	06	02	c2	0b	31	c2	0c	35	3f	c9	00	f2	06	ca	06	35
03a0	e2	02	9c	11	31	e2	01	9c	0c	3f	e2	06	9c	21	c4	0f
03b0	37	c4	ff	33	3f	06	01	02	c2	0b	f4	01	ca	0b	c2	0c
03c0	f4	00	ca	0c	40	07	ba	05	9c	c8	3f	e2	06	98	ba	c4
03d0	01	31	c4	07	35	c4	00	c9	04	c4	79	c9	03	c4	50	c9
03e0	02	c9	01	c9	ff	c4	5c	c9	00	90	fe	c4	6d	c9	06	c4
03f0	76	c9	05	c4	3e	ca	1d	c4	00	37	c4	55	33	3f	c4	40

Table 3.

0400	c9	00	c4	00	c9	ff	c9	06	c9	05	c2	02	ca	14	c2	01
0410	ca	13	3f	03	c2	13	fa	01	ca	01	c2	14	fa	02	ca	02
0420	c4	0a	ca	1d	3f	c4	a0	ca	1d	3f	c4	00	c9	ff	c9	00
0430	c4	48	c9	05	90	fe	c4	39	c9	06	c4	73	c9	05	c4	3e
0440	ca	1d	c4	00	37	c4	55	33	3f	c2	01	ca	0e	c2	02	ca
0450	0d	3f	c2	01	31	c2	02	35	c4	3f	c9	00	c4	71	ca	1d
0460	c4	04	ca	1c	c2	0e	01	c2	0d	36	40	32	c6	ff	c4	55
0470	33	3e	c4	e0	32	c4	0f	36	c4	d5	ca	1f	c4	0a	ca	1d
0480	c4	02	ca	1c	c4	00	37	c4	55	33	3f	c2	08	01	c4	a0
0490	ca	1d	c4	01	31	c4	07	35	40	e4	fa	98	16	40	e4	fe
04a0	98	15	40	e4	f5	98	14	40	e4	f1	98	13	40	e4	f2	98
04b0	15	90	bf	c2	ff	90	1c	c2	fe	90	18	c2	fd	90	14	c2
04c0	fc	01	c2	fb	90	05	c2	fa	01	c2	f9	ca	02	40	ca	01
04d0	3f	90	09	ca	01	3f	c4	00	c9	03	c9	04	c4	00	c9	00
04e0	c9	ff	90	cd	c4	5e	c9	00	c4	5c	c9	ff	c2	0b	31	c2
04f0	0c	35	c4	d7	33	c4	05	37	c2	0c	3f	c2	0b	3f	c2	02
0500	3f	c2	01	3f	c4	20	ca	05	c4	00	ca	06	02	c1	00	01
0510	c2	06	70	ca	06	40	3f	35	e2	02	01	40	e2	02	35	40
0520	9c	08	31	e2	01	98	19	e2	01	31	06	01	02	31	f4	01
0530	31	35	f4	00	35	40	07	ba	05	9c	d2	c2	06	3f	90	c4
0540	c2	06	3f	c4	0f	37	c4	ff	33	3f	c4	7c	c9	06	c4	38
0550	c9	05	c4	3e	ca	1d	c4	00	37	c4	55	33	3f	c2	01	ca
0560	10	c2	02	ca	0f	3f	c2	01	ca	0e	c2	02	ca	0d	3f	c4
0570	0a	ca	1d	3f	03	c2	01	fa	10	ca	0c	c2	02	fa	0f	ca
0580	0b	94	29	c2	10	31	c2	0f	35	c2	01	33	c2	02	37	c5
0590	01	cf	01	c5	ff	31	e2	0e	01	40	e2	0e	31	40	9c	08
05a0	35	e2	0d	98	9e	e2	0d	35	c5	01	90	e3	c2	0e	31	c2
05b0	0d	35	c5	01	03	c2	0e	f2	0c	33	c2	0d	f2	0b	37	c5
05c0	ff	cf	ff	31	e2	10	01	40	e2	10	31	40	9c	f1	35	e2
05d0	0f	98	d0	e2	0f	35	90	e7	ca	07	c4	0b	ca	08	c4	00
05e0	01	19	01	ba	20	c2	07	01	c4	0b	8f	00	c2	15	ca	09
05f0	ba	09	9c	fc	19	40	dc	80	01	ba	08	9c	eb	3f	90	d8

RUN

Once one or more user's programmes have been stored in memory using the MODIFY routine, the RUN-key allows the user to select and then run one of these programmes.

After pressing the NRST and RUN keys the text 'RU . . . . ' will appear on the displays. The decimal points again indicate that Elbug is waiting for data to be entered, in this case the start address of the user's programme. Once this has been done programme execution can be commenced by pressing one of the data- or command keys. The displays will then continue to show 'RU address RU' provided that the user's programme does not require any data to be displayed.

If an XPPC 3 instruction occurs in the user's programme before pointer 3 has been reloaded, then the SC/MP will leave the user's programme and return to Elbug. The displays will indicate this fact by once more showing ' . . Elbug', thereby telling the user that the MPU is awaiting fresh instructions. This feature can be quite useful, since an interrupt call causes the SC/MP to automatically execute an XPPC 3 (3F).

SUBTRACTION

This routine can be used to calculate displacement values when developing one's own programmes. After pressing the SUB-key, ' SH . . . . ' (=Subtraction Hexadecimal) will appear on the displays. A four-digit hexadecimal number should then be entered. When the last digit of this number has been punched in, displays 6 and 7 are blanked whilst a '–' sign appears on display 1. The subtrahend of the first number (which should also be a four-digit number) should then be entered. The result of the subtraction is obtained by pressing one of the data- or command keys, the difference will then appear on the display preceded by an ' = ' sign. If the difference is a negative number then it is expressed as the two's complement.

After a difference has been calculated the CPU enters a loop which can only be exited from via NRST.

Tables 1, 2 and 3 give the condensed listing for Elbug, the monitor software for the Elektor SC/MP system.

Photo 1 shows two of the three EPRoms in which Elbug is stored.

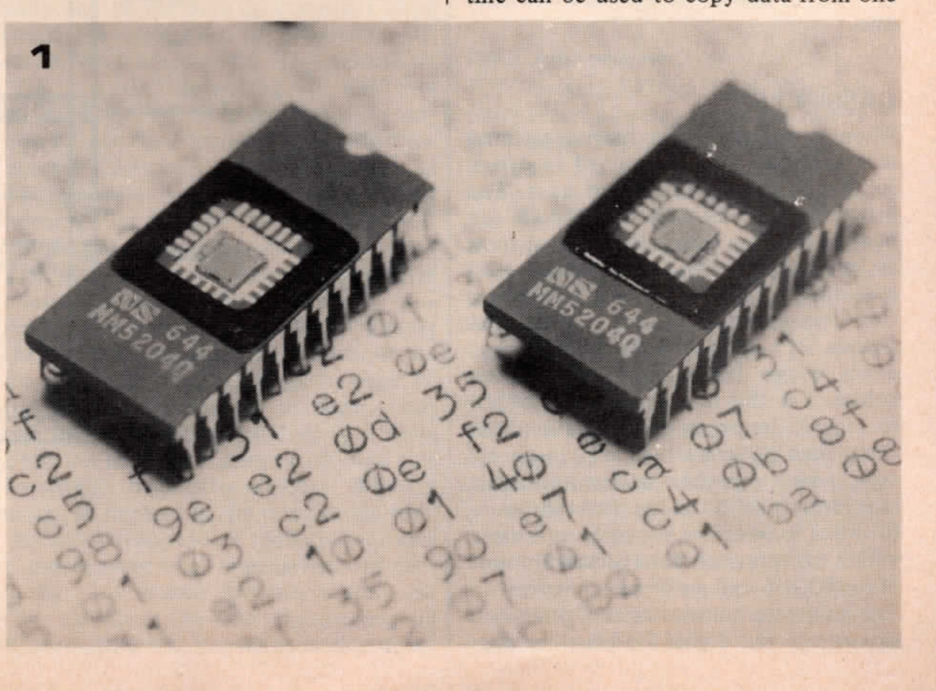
BLOCK-TRANSFER

When developing one's own software it will often occur that several instructions have to be inserted into the middle of a longish programme. In such an event it is possible to spare oneself the chore of having to re-enter great chunks of programme by employing the BLOCK-TRANSFER routine.

This routine allows the contents of any section of memory to be transferred elsewhere in memory. After an NRST the transfer key is pressed, causing the text ' BL . . . . ' to appear on the displays. The first address of the block of data to be transferred should then be entered (the displays then read ' BLxxxx . . '), immediately followed by the last address (' BLyyyy . . '). Once that is done, the initial address of the section of memory to which the data is being transferred can be entered (' BLzzzz . . '). Pressing one of the data- or command keys then results in the BLOCK-TRANSFER being executed. The word ' . . Elbug' will reappear on the displays to indicate that the transfer has been completed.

As long as neither the block of data to be transferred nor the position to which it is to be transferred contains a page boundary, then the block-transfer routine can be used to copy data from one

1



page to another. In the event of a page boundary occuring in the data block or the new position the transfer will not be executed. This is only to be expected since programmes which contain such a page boundary cannot be executed directly, but must first contain a number of extra instructions (XPPC) which make the page jump possible.

CPU-REGISTER COMMAND

This is another routine which offers considerable assistance when developing, and, in particular, debugging one's own programmes. It allows the contents of the CPU registers to be examined on the displays at any stage during the user's programme. The only proviso is that the programme in question must be stored in a section of RAM.
After the NRST- and CPU-keys have been pressed, the text ' CP . . . . ' will appear on the displays. The start address of the programme to be checked should then be entered (the display will read ' CPxxxx . . '), followed by the address which immediately succeeds the last instruction to be executed (' CPyyyy . . '). After pressing any of the data or command keys the user may then examine the contents of the accumulator by pressing key A ('CP (A) '), the contents of the extension register (' CP (E) ') by means of key E, the status register by key 5, and pointer registers 1 and 2 by keys 1 and 2 respectively. The above registers can be examined in any order and as often as desired. This section of the CPU-routine can only be left via an NRST instruction.
Since this routine itself utilises pointer 3, care should be taken to ensure that the contents of this pointer are not altered in the section of programme under test, otherwise the CPU-routine will fail to function. In addition, the contents of the stop address following the section of programme being checked must be returned to their original state. This can be done by means of the MODIFY routine.

CASSETTE

When a programme has been completed it is normally first committed to paper, so that it can be re-entered later. The programme is, of course, loaded via the hexadecimal keyboard. However, as soon as one starts dealing with longish programmes, using a keyboard becomes a time-consuming business, whilst the chance of entering false data always exists where the human factor is involved. For this reason it is only natural to record the programme on a medium other than paper such as, e.g., magnetic tape, which can then be used to interface directly with the microcomputer.
In order to interface the microprocessor with a cassette recorder a certain amount of both hard- and software is required.
The hardware consists of two parts, one to convert the digital information supplied by the μP into signals which can be accepted by the cassette recorder, and another to do exactly the reverse. The cassette interface hardware for the SC/MP system (which can, however, also be used for any other microprocessor system) will be published in next month's article.
The cassette interface software, which ensures that data is read serially from and into memory is already contained in Elbug. This software allows a programme to be read from memory (PROM or RAM) onto a cassette, and programmes recorded on cassette to be written back into memory (RAM). Altough these routines cannot of course be used without the complementary hardware, they will nonetheless be discribed in this article on Elbug.

From memory to cassette

If a programme is to be transferred from memory to a cassette, first the NRST, then the CASSETTE key should be pressed. This results in 'CA . . . .' appearing on the displays. If any of the data or command keys, with the exception of the MODIFY-key, are then pressed, the displays will show ' CA . . . . AD ', indicating that the start address of the programme being read out can now be entered (' CAxxxx . . '), to be immediately followed by the stop address

1

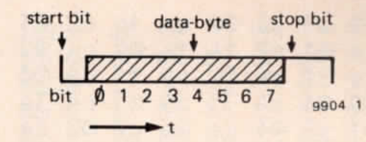


Figure 1. Whenever a byte is transferred from memory to a cassette it is prefaced by a single start bit and followed by two stop bits.

Figure 2. A diagrammatic illustration of the way in which Elbug reads a programme out of memory and onto a tape.

Table 4. This table shows how different transmission speeds can be obtained by entering the appropriate hexadecimal figures.

2

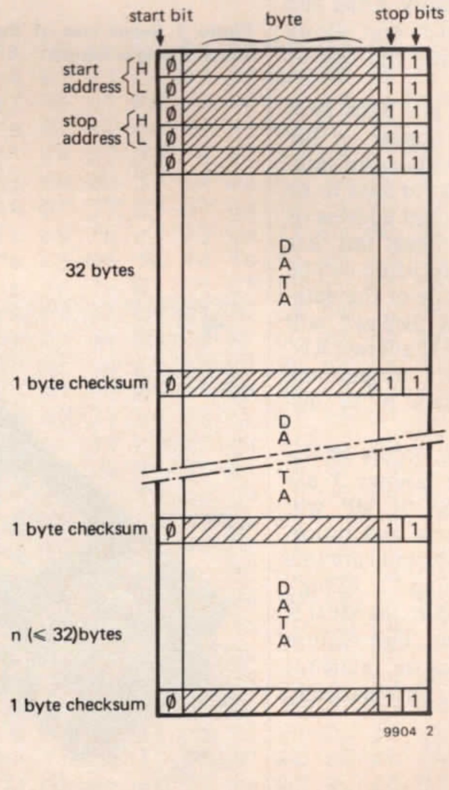


Table 4.

desired speed	number (hex)	calculation
24ØØ Baud	ØØØ2	1/2400 s = 417 μs ≈ 2 x 66 + 285 μs
12ØØ Baud	ØØØ8	1/1200 s = 833 μs ≈ 8 x 66 + 285 μs
6ØØ Baud	ØØ15	1/ 600 s = 1667 μs ≈ 21 x 66 + 285 μs
3ØØ Baud	ØØ2E	1/ 300 s = 3333 μs ≈ 46 x 66 + 285 μs
11Ø Baud	ØØ85	1/ 110 s = 9091 μs ≈ 133 x 66 + 285 μs

('CAyyyy . . '). The recorder is then connected up and started in the record mode.
The interface circuit will then produce a high-pitched tone which the recorder should be allowed to pick up for a short period (about 1 minute). The DOWN-key is then pressed, whereupon the μP will begin to output the desired programme. Each data byte which is read out is accompanied by one start bit (Ø) and two stop bits (1) (see figure 1). These control bits enable the data to be read back into memory again. When the process is complete the word Elbug will appear on the displays.
The start and stop addresses of the programme are also recorded on the cassette. In addition, after every 32 bytes the CPU calculates their arithmetical sum and records the last (i.e. lowest) eight bits of this sum (the checksum) on the tape. This procedure is illustrated diagrammatically in figure 2. At the end of every programme read out in this way there is a checksum of the last section of programme, regardless of whether it is shorter than 32 bytes.
There are no limits placed upon the size of programme which can be transferred to tape. Page boundaries can be crossed and it is even possible to read out the entire contents of memory from start address ØØØØ to stop address FFFF at one go.
Elbug outputs data for the cassette interface at a speed of 600 Baud, i.e. 600 bits (that includes control bits) per second. The data can also be read out at different speeds if so desired. In this case the operating procedure is slightly altered. After pressing the CASSETTE-key, the MODIFY-key should then be pressed, causing the text 'CA . . . . MO' to appear on the displays. The desired speed of transmission is now selected by entering the appropriate number ('CAXXXXMO'), whereupon after pressing any of the data- or command keys (with the exception of the MODIFY key) the transfer procedure can be executed in the above described fashion. The number which is entered to select the desired transmission speed can be found from table 4. The figures for 5 different transmission speeds are given in this table. Naturally enough other speeds can be obtained by entering different figures. Care should always be taken to ensure that the interface hardware is capable of handling the chosen transmission speed.

From cassette to RAM

To transfer a programme which is recorded on tape back into memory the procedure is as follows: One first presses NRST followed by CASSETTE. The recorder is started, and when the tone which is present at the beginning of each recorded programme can be heard, the UP-key is pressed. As soon as the programme begins it is written back into memory, commencing at the start address which was also recorded on the tape. When the transfer is complete '. . Elbug' reappears on the displays.
If anything should go wrong during the transmission, such as e.g. a damaged tape or a fault in the interface hardware, then the text 'CA ERROR' will appear on the displays; this means that the checksum calculated by the CPU during the transmission does not coincide with the checksum recorded on the tape. After an 'ERROR' statement the cassette routine can only be exited from by means of NRST. A second attempt can then be made to load the programme from cassette. It is, of course, also possible that an error occurred during the 'recording' of the programme onto tape, in which case a fresh recording must be made.
As already mentioned, since the start and stop addresses of the programme are also recorded onto the tape, the data is written back into the same section of memory from which it was read out. However there are cases where this may prove undesirable or (in the case of PROM) even impossible. In this instance the start and stop addresses must be entered which indicate the locations between which the data is to be stored. This is done as follows:
First NRST, then CASSETTE, and finally any of the data or command keys (with the exception of MODIFY, UP or DOWN) are pressed. This results in 'CA . . . . AD' appearing on the displays. The desired start and stop addresses are then keyed in one after another ('CAxxxx . . ' and ('CAyyyy . .'). The recorder is then started and after pressing the UP-key the processor will initiate the cassette routine.
The start and stop addresses should be chosen such that the number of addresses between them exactly equals the number of addresses contained in the programme recorded on the tape. If the amount of memory between these two addresses is any smaller, then only a section of the programme on tape will be written back into memory. What is more, at the end of the transfer the text 'Error' will appear on the displays, even if the transfer itself was carried out correctly. The reason for this is that the CPU calculates the sum of a number of bytes before the stop address and compares this with the last recorded byte, which will not normally be a checksum in this case.
Until now it was assumed that the programme was recorded onto the cassette at a speed of 600 Baud. If this is not the case, then, as when reading data out of memory, the desired speed can be selected by using the MODIFY routine.

missing link

Modifications to
Additions to
Improvements on
Corrections in
Circuits published in Elektor

Experimenting with the SC/MP

Part 4, E34 (February 1978), p. 2-28. On the HEX I/O printed circuit board, R42 is connected to supply common instead of positive supply. This error only becomes apparent if a 'low-power' 74LS00 is used for IC6. R42 can either be omitted, or else connected to +5 V. This can be accomplished by connecting the 'top' of R42 (the end nearest to the keyboard) to the R5/C7 junction.

Formant – the Elektor music synthesiser

Part 3, E29 (September 1977) p. 9-42. On the component layout for the power supply board shown in figure 12, the capacitor between IC1 and D1 should be C5 (not C15).
Part 5, E31 (November 1977), p. 11-41. Several readers have requested information on how to extend the frequency range of the VCOs above 10 kHz. There are two possibilities: R12 can be reduced to 47 k, or C2 can be reduced to 2n7 (or even 2n2).

CMOS noise generator

E33, January 1978, p. 1-05. The pink noise filter shown in figure 1 is not as accurate as it might be. A slope of 3 dB/oct. ± 0.2 dB can be achieved using the following component values:
Resistors: R2 = 10 k; R3 = 4k7; R4 = 2k2; R5 = 1 k; R6 = 470 Ω; R7 = 220 Ω; R8 = 100 Ω; R9 = 47 Ω; R10 = 22 Ω.
Capacitors: C2 = 1 μ; C3 = 470 n; C4 = 220 n; C5 = 100 n; C6 = 47 n; C7 = 22 n; C8 = 10 n; C9 = 4n7; C10 = 2n2; C12 = 1 n.
The same changes can, of course, be incorporated if the pink noise filter is used in conjunction with the TTL noise generator (E21, January 1977, p. 1-23, figure 7).
The characteristic of the filter mentioned in the 'Stop Press' at the end of the CMOS noise generator article is 3 dB/oct. ± 0.5 dB.