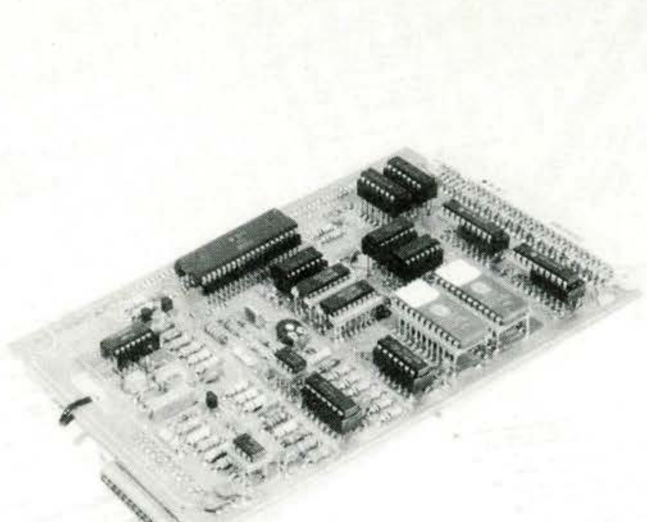


sophisticated software for the Junior Computer

two kilos of brain power, please!

Now that we've added to the Junior's hardware (see the relevant article elsewhere in this issue), we'll have to provide the accompanying software as well. For this Junior was sent to school where the machine assimilated two highly informative programs, 'TAPE MANAGEMENT' and 'PRINTER MONITOR'. The first of these enables data be read and written on tape and the second allows the Junior Computer to be connected to the Elekterminal or to a printer. Tape Management deals with function keys on the standard Junior Computer keyboard and Printer Monitor activates such keys on either the terminal or the printer keyboard. This article is nothing more than a short preview of the in-depth software description to be published in Book 3.

What is involved aren't exactly 'new' keys, but brand new functions that play a key role in the cassette interface. This means the 'old' keys must now bear yet another set of inscriptions. Now that the extension card has been added to the standard Junior Computer, programs can be entered within a much larger address range running from Ø2ØØ up to Ø7FF without interruption. That amounts to 1536 bytes which, thanks to the recording power of the cassette interface, only have to be entered once.

The cassette: a magnetic RAM with magnetism

The Junior Computer's horizons can be widened at the reasonable cost of a simple mono cassette recorder (from £10) and a couple of cassettes. If the C-60 type is used, about 25 minutes will be available per side, as allowance has to be made for breaks (3-4 minutes) and a short space at each end of the tape. This means that at a transmission speed of 50 bytes per second (we'll come back to that later) 25 x 60 x 40 = 73 kbytes (1 k = 1024 bytes) can be stored. That'll do for a start . . .

How is the data stored on tape?

Figure 1 looks rather like a train with a series of compartments in which different information is stored. The data does not necessarily belong to a complete program, it may also constitute a separate unit, a table or a piece of text, etc. In either case, however, a *data block* is involved.

Now let us look at the configuration in figure 1:

1. Considering the tape from left to right, the first 'block' contains 255 synchronisation characters. These filter out the actual beginning of a data block from other information that the Junior Computer would find hard to digest. In other words, don't try to make the machine 'eat' your baby's first babblings, your brother's trumpet voluntary or such remarks as: 'This is my first program on cassette' . . . the Junior Computer is not a parrot!!

N.B. Data is always in ASCII code, in 8 bit words when it is stored on tape. The furthest bit to the left is reserved for special functions and in this case will be zero. The bits of an ASCII byte are stored one behind the other, *serially*. The ASCII code of a synchronisation character is 16 in hexadecimal.

2. The initialisation character '*': its purpose is to signal when the series of synchronisation characters has gone by and the data itself has arrived (hexadecimal 2A in ASCII code).
3. The program number ID: this enables one program to be differentiated from another, thus ID really stands for ID here! Since 254 different program numbers are possible, that includes all the values between Ø1 and FE. Values ØØ and FF fulfil a specific task when data is read from tape.
4. The low order address byte SAL:

1

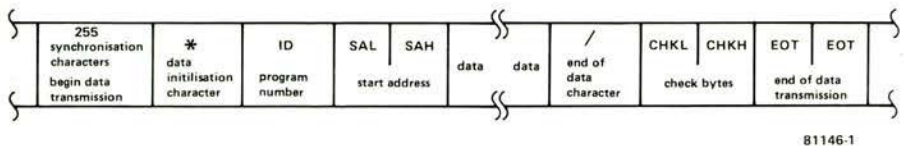


Figure 1. This is what a data transmission looks like on tape. It consists of a block preceded by 255 synchronisation characters, the data initialisation character, the program number and the start address, and followed by an end of data character, check bytes and two end of data transmission characters. This taping procedure is similar to that in the KIM computer, the only difference being that the Junior features 255 synchronisation characters, whereas the KIM has only 100.

corresponds to the first memory address belonging to the program or data block that is to be written on tape.

5. The high order address byte SAH.

6. The actual data block: each data byte is transcribed into two ASCII bytes, that is to say, one ASCII byte per nibble. The data byte is then stored on tape as a series of 16 bits. The write operation begins with the start address (SAH, SAL) and ends with the byte stored at address EAH, EAL minus one.

7. The end of data character "/" which indicates that all the data have been transmitted (ASCII: 2F).

8. The check bytes CHKL and CHKH: these make sure nothing was mislaid or misread during the recording. A cassette tape has a special gift for distorting information, inspite of the improved PLL (see the article on the hardware). It is therefore absolutely necessary to check whether one or more bits did not happen to be viciously mutilated in some invisible snare, the best method being to count them like the shepherd counts his sheep.

At the beginning of a write operation, the CHKH/CHKL bytes are zero. From SAL on, the bytes are all added to each other before being transcribed into the ASCII code. The program number (ID) is not taken into account here. Each time the contents of CHKL reach FF, that of CHKH is incremented once and when this too reaches FF, both are reset without causing any complications. When data is read on tape, the same procedure is used. All that has to be done then is to compare the result of the two counts: the sum of bytes during the write operation must equal that during the read operation, and vice versa. If so, there is reason to believe (and hope!) the transmission has been carried out correctly. If not, the tape will have to be tidied up here and there. We could of course provide you with a whole chapter of error statistics and other theoretical titbits . . . but at this stage it is better for readers to cross that bridge when they come to it.

With regard to the interface check system, let us compare it to a bank to illustrate its function. It is in the bank's (and our!) common interest to see that people's money is well looked after. It would therefore be disheartening, to say the least, for the employees to discover one morning that one of the clients had remained in the building after closing and had subsequently absconded with the cashbox. This can be avoided quite simply by counting all the clients who have come in and gone out during the day and then compare the two numbers. If they do not tally, then there is reason for worry. It could mean the counting system can't count, or that two persons passed the detector at exactly the same moment and were therefore registered as only one (what a primitive system, get a refund!), or even that a client cashed in on his Pool winnings and was shown out discretely by the manager through a door at the back. If two of the cases mentioned happened to coincide they would automatically cancel each other out and so would escape notice altogether.

As far as CHKL and CHKH are concerned, counting the bytes by adding them is, believe it or not, a perfectly foolproof method, inspite of all the peculiar circumstances that might arise.

9. Two EOT (End Of Transmission) characters which indicate that a data block has been transmitted (ASCII: Ø4).

How to write on tape

As we said before, data is transmitted bit by bit. In figure 2 pulse trains are shown with long wagons (relatively high frequency) and short wagons (relatively low frequency). A low level bit is made up of four half periods of 2400 Hz and a high level one consists of three half periods of 3600 Hz (figures 2 and 2a). A logic zero consists of six 3600 Hz half periods and two 2400 Hz half periods. The total length of the train remains unchanged, irrespective of whether the logic level is high or low. (This can be expressed as 9T where T equals the duration of a half period of 3600 Hz.) Furthermore, the train will always start at the higher frequency. The duration ratio for the high and low frequencies is either 2:1 or 1:2.

Figure 2b shows what happens in the KIM. The graph had to be spread across several lines, so please follow the arrow. Here a high logic level bit consists of nine full periods of 3700 Hz (which in this instance will be rounded off to 3600 Hz) plus twelve full periods of 2400 Hz. A low logic level bit, on the other hand, is made up of eighteen full 3600 Hz periods plus six 2400 Hz periods. From these figures it can be seen that a bit in the KIM lasts six times longer than one in the Junior Computer. The reading and writing speed is therefore much slower, but this has been remedied by the HYPERTAPE program written by Mr. J. Butterfield, which enables the transmission speed to be accelerated considerably, so that one KIM bit then equals one in the Junior Computer. The software in the Junior's DUMP/DUMPT writing routine differs from that of HYPERTAPE in a number of fundamental aspects. To look at this in detail would hardly suit the purpose of the present article, but not to worry, several pages are devoted to the subject in Book 3.

To get back to what we were saying, one bit of cassette software in the Junior Computer lasts 9 half periods of 3600 Hz, which is $9 \times 139 = 1250\ \mu s$. In other words, 800 bits (or 100 ASCII characters = 50 data bytes) per second.

Data is written on tape during the DUMP/DUMPT routine. During this write operation the six displays remain unlit. The parameters for a successful operation are:

a. indicate the program number ID (Ø1 . . . FE; ØØ and FF serve a special function, which we'll come back to later)

b. indicate the start addresses SA

c. indicate an end address EA

How to read on tape

To return to figure 2a, this also contains signals which can be found at the PLL output. The PLL is activated when data is read on tape, as was mentioned previously in the hardware discussion, where we also referred to the 1 to 3 and the 2 to 4 transition. The PLL's output signal clearly shows how the ratios 2:1 and 1:2 may be obtained.

According to duration of the PLL output's high and low levels, the software will filter out either high or low level bits from the PLL signal. It is not the durations themselves that matter, but their relation to each other: is the 3600 Hz period (high PLL) any longer than the 2400 Hz (low PLL), or vice versa? If the 3600 Hz lasts longer than the 2400 Hz, the bit in question will be logic zero, otherwise it will be logic one. Since the durations themselves are immaterial, the Junior Computer may be used to read back periods taped on the KIM, inspite of the pulse length differences between the two systems. Consequently, signals 3 and 4 in figure 2a are identical to those in figure 2b, even though the signals in the second figure (2b) are six times longer than those in 2a. This is a great advantage, even for readers who do not have occasion to use the KIM, as it means that variations in the reading or recording speed (usually

4.75 cm per second) will not affect the transmission quality — there will be no 'flutter', etc. In other words it does not matter which type of cassette recorder is used and it is possible to tape on one recorder and play back the cassette on a different one altogether. This is because the 2:1 (low logic level) and 1:2 (high logic level) ratios are too wide apart to cause confusion.
N.B. PLL jitter is not taken into account during a data reading (see the hardware article for more details).
Data is read with the aid of the RDTAPE routine, which is called during the TAPE MANAGEMENT program. The two right hand displays of the Junior Computer conveniently indicate what is going on during the reading. The remaining four will stay unlit.

The first drawing in figure 3 applies when:

a. the tape passing the reading head (cassette is on 'play') contains no digestible data (space between two data blocks, empty tape, Beethoven's 5th, etc.). D5 and D6 will be seen to flicker during this period.
b. the tape contains a data block which is being read, but the beginning is missing, or the ID number does not correspond to the one specified. D5 and D6 will no longer flash but will remain constant. This situation is in force when the computer is in the synchronisation phase. In other words, it is reading the synchronisation characters preceding a data block. The reading may well not quite be perfect with regard to the initial characters, so that the display pattern in the drawing will flash for about 1 second before becoming stable. There are 255 synchronisation characters on the tape. These ASCII bytes take about 2.5 seconds to read. The Junior Computer is able to detect the beginning of a data block as soon as it has read an uninterrupted sequence of 10 synchronisation characters. Since there are 255 altogether, the Junior Computer has an excellent chance (in fact at least 20 possibilities) of making an unambiguous detection. The KIM, on the other hand, only has 100 synchronisation characters and so things are far more likely to go wrong here.
The third situation as shown in figure 3 occurs when the ID number specified by the operator has been found and loaded into the Junior Computer memory.
Before data on tape can begin to be read, that is, before the jump to the RDTAPE routine, an identification number must be specified. Even though a tape may feature up to 254 different data blocks, entering the number of one of them before the reading is enough for the computer to be able to trace it. There is also another method. If either ØØ or FF is introduced as an ID number, the computer will load the first data block that happens to arrive. If, however, ØØ is specified, the data block's number will be ignored and the block will be stored in memory at address SA

2

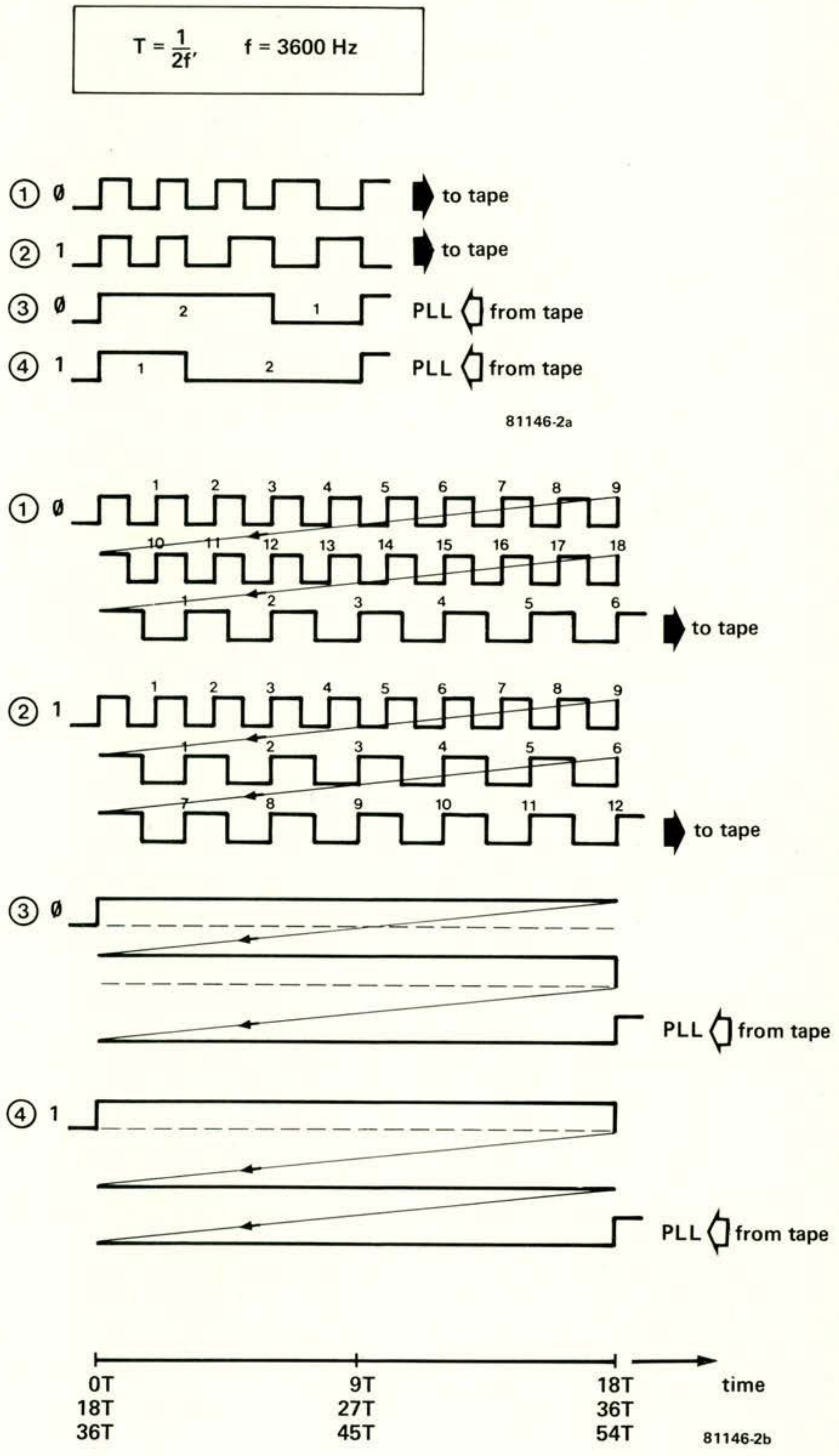


Figure 2. The signals that are transmitted on tape during the write operation correspond to a Ø bit ① (a) in the case of the Junior Computer and to a 1 bit ② in that of the KIM (b). The common time base at the bottom of figure 2 has been 'reset' twice to reduce the size of the drawings. The time base is split into 3600 Hz (T) half period units. The signals in ③ and ④ are produced when the tape is being read and the data has been processed by the PLL. After being inverted by the computer the signals are used to reconstruct the read data and store it in memory.

on the tape. If, on the other hand, FF is specified, both the data block number and the start address SA on the tape will be ignored. In that case, the data block will be stored in memory at an address selected by the operator at that particular moment.
This enables the data blocks to be moved around easily. The only consideration that needs to be taken into account is that whenever FF or ØØ is used as an ID number the 'first come, first serve' rule will be enforced, meaning that the first data block that happens to arrive in due form will be loaded in memory. In other words, the operator must know exactly which data block is to be transferred and where it is situated on the tape. The easiest solution is to use a cassette recorder which includes a counter.

DATA MANAGEMENT

The Tape Management program

The TAPE MANAGEMENT program (which we'll refer to as TM from now on) could also be called 'Tape Monitor'. It is designed to fulfil all the operator's wishes with respect to writing data from the Junior Computer onto cassette (= reading from the Junior Computer memory) and reading data from cassette into the Junior Computer (= writing into the Junior Computer memory). The program resides in a 2716 EPROM. Although this comprises more than 1024 bytes, it by no means occupies the full 2048 bytes, so that there is room for a few bytes which will come in handy later on.
TM ranges from Ø8ØØ to ØC7F. Nevertheless, its start address is not actually Ø8ØØ as might be expected, but Ø81Ø! Sometimes the TM is left by way of the editor (see figure 2) and sometimes it is exited from by depressing the RST key, after which the computer will be back in the standard monitor routine.
As soon as the computer is activated (AD Ø 8 1 Ø GO) the first situation in figure 4 will appear on the display. Then depressing the PAR key (= + key) will lead to the second situation in figure 4. Every time PAR is depressed, the next situation in the series will arise until, finally, PAR brings us back to the beginning again (drawing 1). As you have probably already guessed, 'PAR' stands for 'parameter'. This term is used to define the size of a particular data block and its whereabouts on cassette. As shown in figure 4, there are nine parameters altogether, one or several of which have to be specified (depending on which other function key(s) of the four that TM also acknowledges was/were selected). The parameters are as follows:

- ID (program or data block number)
- SAH
- SAL
- EAH
- EAL
- BEG(AD)H
- BEG(AD)L
- END(AD)H
- END(AD)L

It should be noted here that the high order address byte is specified first for once.
When the required parameter is shown on the display, two numeric keys are depressed, one after the other. The two key values (nibbles) move from right to left across the two left hand displays in exactly the same way as in the DA mode. After the start of TM, the nine parameters are reset (ØØ). This is precisely what is shown in figure 4. In this way, the PAR key allows data to be introduced which the Junior Computer needs to be able to successively carry out the read and/or write operation(s). This is very convenient for the operator, as he/she can now 'see' what is going on. If the more primitive system were to be used (AD Ø Ø E 2 DA X X + Y Y) the operator would not really know what was happening: ØØE2, is that BEGADL or BEGADH??...
N.B. Nine locations on page ØØ or 1A correspond to the nine initialisation parameters in figure 4. Locations 1A69 . . . 1A7F may not be overwritten during the cassette read operation!
What about those four function keys that were mentioned earlier? These are:

1. SAVE: this is a new name and a new function for the AD key. It enables data stored in the Junior Computer's memory to be saved by transferring them to cassette. Before the key is operated, the cassette is prepared for recording (depress 'rec' and 'play' simultaneously).
Again before this, however, an ID (neither ØØ nor FF) and the parameters SAH, SAL, EAH and EAL have to be indicated for the data block that is to be recorded. Take care! EAH and EAL form an address which is located one place behind the last address of the data block concerned. Thus, if the end of data address is Ø3FF, EAH = Ø4 and EAL = ØØ.
When the SAVE key is operated it calls the routine DUMP. The red LED D5 is lit, but the six displays will remain unlit. Once the data block has been recorded, the Junior Computer will announce this by displaying 'ID XX' (first drawing in figure 4), where X is instead of ØØ, being the number of the data block which has just been copied on cassette.
N.B. Operators will find it highly useful to jot down the ID, SA and EA, as well as the value displayed by the cassette recorder counter, on a piece of paper.
2. GET: this is a new function for the PC key. When it is operated the Junior Computer reads a certain data block stored on cassette and memorizes it. The cassette recorder will have been put on 'play' beforehand. As before, it is enough to specify the ID number. Only ID numbers included between Ø1 and FE will be stored on the cassette. If, before depressing GET, ØØ is introduced as an ID number, the Junior Computer will store the first data block it meets in memory, without taking any notice of its ID. Instead, the data block's SAH and SAL will determine where the block is to be placed in memory.
If, on the other hand, FF is introduced as an ID number before GET is operated, the Junior Computer will again

3

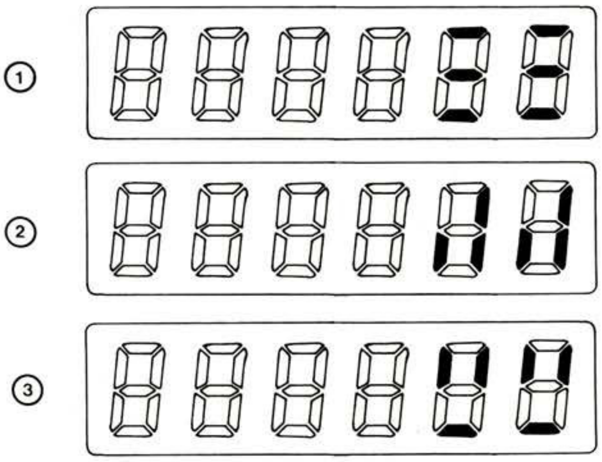


Figure 3. When a data transmission (data block) is being searched, detected and read on cassette, information will be shown on the two right hand displays of the Junior Computer. This is not the case in the KIM computer.

store the first data block it comes across in memory, regardless of its ID, but now the block will be stored in an address range specified by the operator and the SAH and SAL cassette will be ignored.
Depressing the GET key calls routine RDTAPE. Then the green LED D4 will light and one of the situations indicated in figure 3 will be shown on the display. Once the block is loaded, the Junior Computer will inform the operator by displaying 'ID XX' (situation 1 in figure 4, where XX is a number between ØØ and FF).
N.B. When the data block is loaded using FF as its ID, only the contents of ID and SAH/SAL will be correct. So don't expect to find the end of block address in EAH/EAL!

In addition, something else happens: the start address (contents of SAH/SAL) on display will constitute the end of data block address. This allows edited programs to be placed side by side without loss of space (the EOF characters are suppressed). Bearing this in mind, if any data blocks are to be placed side by side using ID = FF, EAL will have to be modified (incremented by 1) and, if necessary, so will EAH, before a new data block is loaded. This type of storage will obviously be necessary when a program is to be reconstructed from various sections dispersed here and there on the cassette. In that case, it is very important to know 'what's where'!

3. EDIT: this attributes a new function to the DA key. In actual fact, the function concerned is not entirely new, as activating EDIT is equivalent to operating AD 1 C B 5 GO, leading to a cold start of the editor. Prior to this, BEGAD and ENDAD are introduced by means of the PAR key (operate PAR until BEG(AD)H is found, enter the corresponding data, operate PAR and enter suitable data).

As you know, a cold start in the editor causes '77' to appear on the left hand displays. This is exactly what happens when EDIT is operated as well, so that it may well seem to be superfluous. Is is really necessary? The answer is yes, as will be understood from the following.

4. SEF: this is a new function attributed to GO. It has a special task to fulfil just before TM is started. SEF stands for Save Edited File; in other words, a data block that is *not completely edited*, and therefore *not yet assembled*, is transferred to cassette. Provided the EDIT key is depressed beforehand (cold start editor), all the program data from BEGAD onwards (like ENDAD, BEGAD will have been entered prior to depressing EDIT) up to the address to which the variable pointer CEND is pointing, will be copied on tape in the form of a single data block during the DUMP routine. For such a block to be transferred by means of the SEF key, it is absolutely imperative that BEGAD and CEND are defined, which of course is only possible once the program has been edited! Thus, before SEF is operated to tape an edited program on cassette, the editor must be left by way of the monitor and the computer then jumps to TM (during this time, the Junior Computer must under no circumstances be switched off – BEGAD and CEND are stored in RAM in page ØØ!!):
RST Ø 8 1 Ø GO
enter the ID
and operate SEF.
It is during the DUMP routine that data is written on tape. The display remains unlit until SEF is operated, but the red LED is lit. Once all the data has been transferred the first instruction of the edited program that has just been taped

4

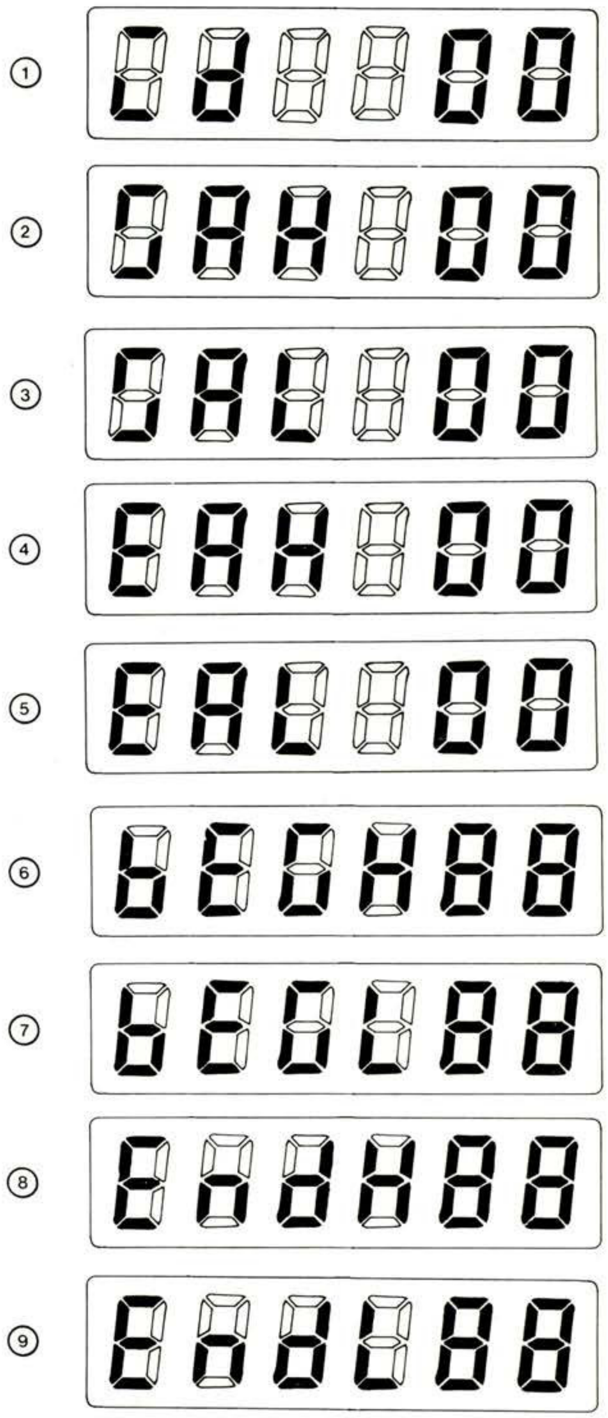


Figure 4. The nine parameters which play a decisive role when data is being transferred from or to the cassette, can all be brought into view during the TAPE MANAGEMENT program with the aid of the PAR = + key, so that data may then be keyed in (shown on two right hand displays).

will appear on the display. This is because after DUMP TM ensures a warm start entry into the editor. Previously, of course, a program number had to be entered and as for the start address, this is equal to BEGAD, the end address (the end of data block address plus one) being equal to CEND.

How can we benefit from SEF? This function allows us to store incomplete (non assembled) programs of any length by 'putting them on ice' so to speak. All that has to be done to fetch such data blocks is to depress the GET key and then activate the editor by means of a warm start entry. Let us see how this is done. The variable end of address pointer CEND is pointing at the first vacant location in memory at the end of the data block. Straight after the EOF character, in other words. If *your* memory is failing you on this point, it is a good idea to take another look at chapter 8 in Book 2. Since it is the CEND pointer that acts as the EA during an SEF operation, the last data to be read in the block will be 77, the EOF character. The operator does not have to enter BEGAD = SA and CEND = EA before depressing SEF, as this is done automatically by the machine. What must be specified, however, is the program ID number, which is as well to write down! Why? Well, because after the data block is reread:

RST or AD Ø 8 1 Ø GO X Y (XY = program number) the computer must be prepared to jump to the editor.

First of all, RST (return from TM to monitor) is depressed. Then the contents of BEGAD (L = ØØE2, H = ØØE3) are made to equal that of BEGAD = SA as was noted during the data block recording. After this the contents of CEND (L = ØØE8, H = ØØE9) are made to equal CEND = EA which will also have been noted previously. All that remains is to make CURAD's contents equal BEGAD's and then the computer may proceed with the warm start editor entry:

AD 1 C C A GO.

Since CURAD now equals BEGAD, the first instruction will appear on the display.

N.B.: The editor should *never* be activated by means of a *cold* start entry in this particular instance, by depressing EDIT, say. This would inadvertently cause 77 to appear instead of the first instruction, which would really put the cat among the pigeons.

N.B.: When several blocks of edited data are being read on tape and ID = FF, the value of CEND will be equivalent to the address of the last data block to be read.

An edited and taped data block (or several) may be re-addressed by re-reading it (them) with FF as the ID. It goes without saying that in that case the CEND and CURAD parameters must be adjusted accordingly.

```
(RST 1 Ø Ø Ø GO)

(CTRL+DEL=RUB)
JUNIOR

1A7E(SP)
1A7E 04 CF.
1A7F 00 14.
1A80 80 100(SP)
0100 5D 18.
0101 1D A9.
0102 1C 13.
0103 3D 69.
0104 3C 08.
0105 3C .
0106 2C +
0107 AC -
0106 2C -
0105 00 -
0104 08 -
0103 69 -
0102 13 -
0101 A9 -
0100 18 L
ACC: C0
Y  : CB
X  : CA
PC : F33C
SP : 01FF
PR : 00000100
     NV BDIZC 100(SP) (STEP:OFF)
0100 18 R
0107 AC L
ACC: 1B
Y  : CB
X  : CA
PC : 0107
SP : 01FF
PR : 00110100
     NV BDIZC F3 (SP)
00F3 1B 100(SP) (STEP:ON)
0100 18 R
0101 A9 L
ACC: 1B
Y  : CB
X  : CA
PC : 0101
SP : 01FF
PR : 00100100
     NV BDIZC P
0101 A9 R
0103 69 L
ACC: 13
Y  : CB
X  : CA
PC : 0103
SP : 01FF
PR : 00100100
     NV BDIZC P
0103 69 R
0105 00 L
ACC: 1B
Y  : CB
X  : CA
PC : 0105
SP : 01FF
PR : 00100100
     NV BDIZC P
0105 00 R
0107 AC L
ACC: 1B
Y  : CB
X  : CA
PC : 0107
SP : 01FF
PR : 00110100   (STEP:OFF!)
     NV BDIZC M
HEXDUMP: 100,105
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0100: 18 A9 13 69 08 00

JUNIOR

S11,100,106
READY
```

The Printer Monitor program

How the computer writes home to the operator

The PRINTER MONITOR program (which will be called PM from now on – no relation to the resident at no. 10!) will be seen to consume yards of paper when a printer is connected to the computer. To describe it in detail would take a few reams as well! The program is stored inside a 2716 EPROM and extends from address 1ØØØ to 14F3. Again, the EPROM has enough room left for other resident programs.

The PM features the following standard function keys: AD, DA, + and GO. This time, however, data will not be overwritten by other data on the six Junior Computer displays, but we will end up with a complete case history of everything we do. If a printer is used, the paper may roll on indefinitely and all the information will remain permanently available, which is certainly not the case with a video terminal, where the memory and display capacity is of course fairly limited.

The PM is started by way of the monitor:

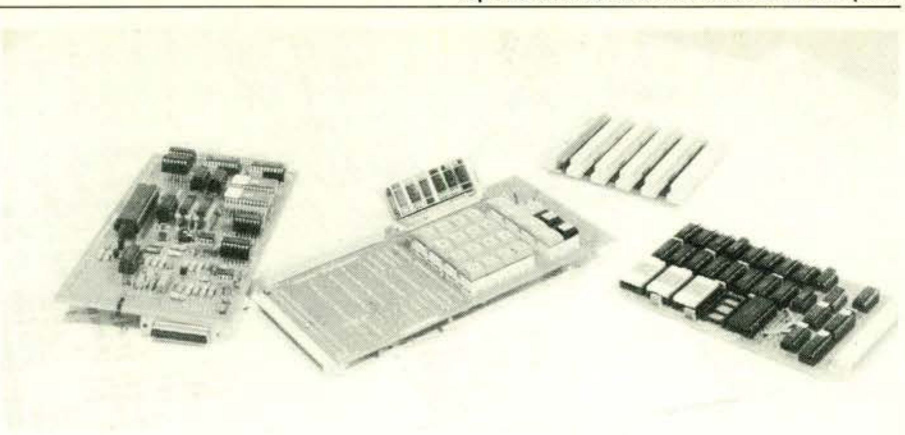
AD 1 Ø Ø Ø GO

Depressing the Elekterminal's RUB key (CTRL or DEL in other devices), will cause the Junior Computer to answer by displaying the word 'JUNIOR'. The ASCII keyboard keys Ø . . . 9 and A . . . F may then be used to enter a work address. Insignificant zeros may be omitted: thus, '2ØØ' stands for 'Ø2ØØ'. The work address appears with the contents of the corresponding memory location, once the SP (SPACE) key has been operated. If data is to be entered at this address, the first two keys (Ø . . . F) are pressed and then the "." key (full stop). The information is stored in the Junior Computer's memory which subsequently displays the following address and its contents. The same procedure is repeated.

Function keys

In addition to the auxiliary keys, RUB, CR etc., there are ten function keys:

1. The "–" key: this enables the address immediately preceding the work address to be printed; in other words the work address is decremented.
2. The "+" key: this increments the work address in the same way as the + key on the standard Junior Computer keyboard. The new address is shown with its corresponding memory contents.
3. The SPACE key: the specified work address appears together with its contents. This key is similar in every way to the AD key on the standard keyboard.
4. The "." key: the last data to be entered is stored at the work address. This operates just like the DA key in the standard Junior Computer.



5. The R key: "R" stands for "run" and operates like the GO key; the program is started from the last address to be printed (= work address).
6. The L key: "L" stands for "list". When this key is operated the contents of all the internal registers belonging to the 6502 μP, including ACC, Y, X, PC, SP and P, are listed. The P register is represented by eight bits, each one of which has an identification letter: N, V, (space), B, D, I, Z and C.
7. The P key: "P" stands for "print" and is depressed to show the PC (program counter) contents, as in the step by step mode just before the program is left (after an instruction was executed) to go to the PM. Thus, similar to the PC standard key, the P key makes sure the following instruction is prepared (depress R). Step by step programming is only possible in the PM when S24 is "ON" (the LED of the GO key will then be lit) and provided the circuit around IC10 on the main board is correct (see figure 1b in the 'Junior Cookbook' article in the April issue).
8. The M key: When this key is depressed, the text 'HEXDUMP' is printed. Then an address is entered (without the insignificant zeros), the "," (comma) key is depressed and a second address is entered. Depressing CR will cause the hexdump between two specified addresses to be listed. At the beginning of every row of 16 data the address of the first data in the line will be shown. Each data column is headed by a figure between Ø and F, enabling a certain data address to be found. The last line in the hexdump is not necessarily complete, as the number of addresses does not have to be a multiple factor of 16.
9. The G key: "G" stands for "GET". Depressing this, then the program number required (ID) and finally the CR key, will tell the Junior Computer to search the data block corresponding to the ID specified on the cassette tape, and copy it into memory (provided the recorder is on 'play'). Once the read operation has been completed, the word 'READY' will be printed. This means all has gone well. If the program number was specified as ØØ, the first data block the Junior Computer comes across will be read and stored in memory. If the ID was FF, "SA" will appear and this address will have to be entered, after which the first suitable data block will be searched and stored in the computer's memory at an address specified by the operator.
10. The S key: this allows a data block to be written on tape. This is what happens: The S key is depressed, then the required program number is entered, the "," key is depressed and the start address SA is entered. Next the "," key is operated again, the end address SA is entered (one address behind the last data block), the recorder is set on 'record' and is started. Finally, the CR key is operated. Once everything is correctly recorded, the Junior Computer will let you know by printing "READY".

That brings us to the end of our survey, and we would like to conclude by adding a few helpful hints.

Operators know how to use the numeric keys (ASCII Ø . . . 9 and A . . . F) and the others, such as "." and SPACE, to enter a work address, or to modify data. As soon as a function key is executed, the corresponding data buffers are reset. After this, the SPACE key is operated and the work address becomes ØØØØ. If "." is pressed at a certain work address, the contents of that particular address will become ØØ.

The PM program automatically specifies the NMI vector. As a rule, NMIL = CF (address 1A7A) and NMIH = 14 (address 1A7B). This relates to the step by step execution of a program. If a program ends in a BRK instruction, a direct jump may be made to PM, provided the IRQ vector is correctly positioned. As a rule, IRQL = CF)address 1A7E) and IRQH = 14 (address 1A7F). Data relating to the vector may be entered either before or after the PM has started.

After the start of a program (key R, with work address = start address) ending in a BRK instruction, the Junior Computer will again display or print the address and its contents. This address will be two addresses after the one containing the BRK instruction (provided the IRQ vector indicates 14CF). In the step by step mode, the Junior Computer will report back by displaying or printing the address containing the opcode of the following instruction. ◼