Micro-Soft Basic

Uwe Schroeder KIM Kenner 5 September 1978

Problem 1: You wrote a basic program, telling your basic you don't want sin,cos,atn. After that, you decide to use a sinus in your program. Loading the basic-program again, you tell it: Yes, I want sin,cos,atn, and then load the sourcetext by means of the LOAD-command. Making a LIST your program seems to be loaded incorrect, why? Asking basic: "PRINT SIN(1)" your basic is blown, why?

Answer: Micro-soft tried to make basic programs "relocatable" by writing the program on tape with a tape-identifier FF. Reloading the tape they ask the KIM cassette load program to load a program, tape identifier FF and therefore to put the content of the tape at the adres specified in 17F5-17F8 (KIM-user manual, ch. 4.2 using the tape recorder, loading data from audio tape point 8) .
From the flow-diagram it is clear why this does not work. The first question in the tape load program is: Are the tape-identifier at 17F9 and the tape the same? Because this is true if both are FF, the tape wil be loaded at the adress specified on tape, and <u>not</u> at the adress specified at 17F5,17F6. Changing the tape identifier on tape

into any value not equal FF results in loading of the tape at the adress specified at 17F5,17F6, and you can fix this by changing at adress 2743 the code A9 FF by A9 00 (9 digit version)

Problem 2 You want to use hypertape, a high speed paper tape, or ....
How?

Answer The dump-routine is called at 275C :JMP 1800 (4C 00 18). You are allowed to change this into: JSR "hypertape", returnig to the basic with RTS. The begin and end adresses of the text buffer are available at 17f5 to 17f8.
The load routine is called at 27A6: JMP 1873 (4C 73 18). Change this into JSR "tape load", you have to specify at 17ED and 17EE the end of the text loaded (that is: the adress of the last byte loaded + 1), or change at 27B8-27BD: LDX $17Ed, LDY $17EE to accomodate basic needs.

Problem 3 You own a video terminal and want to change the "rubout"(5f) into "back space". Even if you found where to fix this, it does not work.

Answer Change the "getline" routine at 2426, and you never will have problems with the rubout code.

```
2420  ca        br2420   dex
2421  10 05              bpl br2428

2423  20 bf 29  br2423   jsr crlf
2426  a2 00     getlin   ldx# $00
2428  20 56 24  br2428   jsr getch from kim
242b  c9 07              cmp# $07          ;bell is a valid char.
242d  f0 14              beq  br2443

242f  c9 0d              cmp# $0d          ;carriage return?
2431  f0 20              beq  br2453

2433  c9 08              cmp# $08          ;rubout?
2435  f0 e9              beq  br2420       ;yes, then skip previous char

2437  c9 7d              cmp# $7d          ;char>7d, then skip it
2439  b0 ed              bcs  br2428

243b  c9 40              cmp# $40          :cancel line?
243d  f0 e4              beq  br2423

243f  c9 20              cmp# $20          ;char<20, then skip it
2441  90 e5              bcc  br2428
```

(lines to be changed are underlined)

Problem 4 You want to include data in your basic program, maybe even change this data in runtime and want more information on how text is stored in the microsoft basic.

Answer Text in the textbuffer is stored in code. Each line of a basic program results in one line of code. The organisation of this line of code is:

1. The line starts with two bytes containing the memory adress of the next line in the buffer
2. Then two bytes with the (hex) line number
3. A code of one byte for each command or ASCII code (but not equal 0). A byte ≥80 hex represents a command, for instance 83 is equivalent with "DATA", 8E with "REM" .(83 and 8F in the PET). Numbers in your program are stored by means of their ASCII code.
4. A line is terminated by the byte 00
5. The number of bytes of one line may not exceed FF hex, resulting in max. 250 bytes real code and 5 bytes overhead.
6. The text buffer starts with the byte 00, followed by the program lines
7. The text buffer is terminated with the bytes 00,00; the adress of the first 00 is noted at 7A,7B in the Zero Page (Pointer to start of simple variable table).

Problem 5 You wrote your own monitor, after hitting reset, restarting the basic at 0000 4C you cannot RUN programs any longer.

Answer Entering the basic with a stack pointer FD (my monitor) in stead of FF (KIM monitor) results in an error on the stack if the RUN or CLEAR statments are executed.

Problem 6 Is the Micro-soft basic interruptable?

Answer Maybe Yes, I wrote an interrupt driven Telex output routine, and it works until now.

Problem 7 Your basic program is short, you donot use many variables and get the message :" out of memory". (try for instance the program: 10 k = k + 1: print k: gosub 10, after number 26 you get the message "out of memory". Microsoft told you: Gosub nesting is limited only by available memory, did they lie?

Answer On each gosub basic pushes the "return adress" on the stack, just like a JSR in machine language. The same is true for each for-statement Because the stack on the 6500 is limited, the number of gosubs and for-statements is also limited to 26 gosubs or 10 for-statements that are nested. It is a pitty that microsoft didnot seperate this "out of memory" from the "out of memory" of the text/variables buffer.

Problem 8 Is the basic promable?

Answer Yes, but you better rewrite the initialisation part of the basic, starting at $4065. By the way, I think it is not wise to put ROM at the adress 2000-3FFF because most programs are writen in that area. I use a memory-protect on my system and hypertape, so loading the basic is done quickly, and it is not destroiable just as ROM.

Problem 9 What does the GET statement do?

Answer I wish I know. Try the next program: (User input is underlined)

```
10 GET H$: PRINT "H$="; H$
20 INPUT H$: PRINT "H$="; H$
RUN
X ?? How strange, isn't it?
H$ = How strange, isn't it?
? How strange, isn't it?
EXTRA IGNORED
H$ = How strange
OK
```

Using GET H$ basic asks for input by the keyboard without telling it (using INPUT H$ basic first output a question mark). The first letter typed in however is skiped, answered with two questionmarks, and everything typed thereafter is put into H$, inclusi comma's. By the way, the input call for the GET statement is at 2AE5.

U.O.Schröder