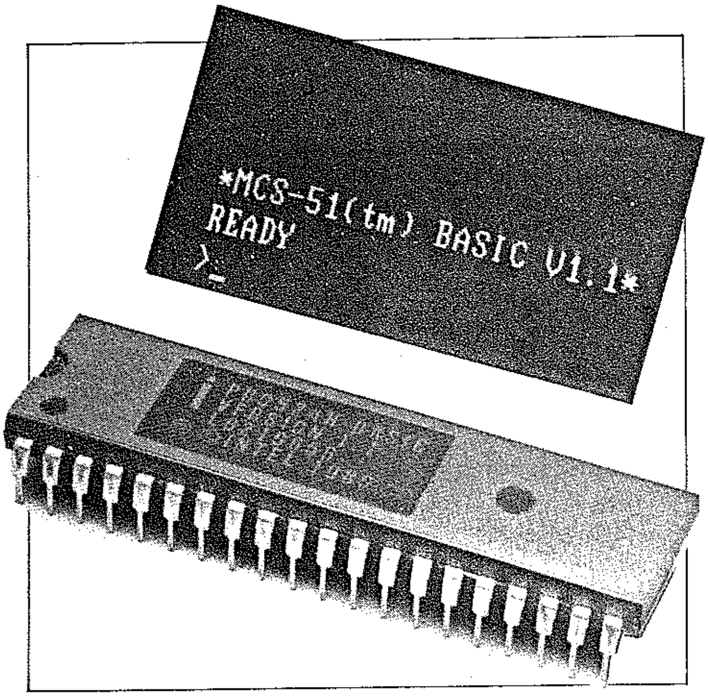
UPGRADE FOR MCS® BASIC-52 V1.1 (Part 1)

The 8052AH-BASIC from Intel is a versatile microcontroller with a powerful BASIC interpreter lurking in its on-board mask-programmed ROM. The authors, having worked with this IC for some time, discovered certain flaws in the BASIC interpreter, and set out to produce a better, faster version that can be run from EPROM.



by Dusan Mudric and Zoran Stojsavljevic

TO be able to make changes to the MCS-52 BASIC interpreter in the 8052AH-BASIC microcontroller, it is necessary to first unload it from the IC. This is done basically as described in an earlier article on the MCS BASIC-52 interpreter, Ref. 1. The result of reading the 8-KByte ROM is a file in Intel-Hex format that contains the machine code of the MCS BASIC-52 interpreter (version 1.1).

MCS BASIC-52 (Ref. 3), extracted from the 8052AH-BASIC V1.1 microcontroller, was disassembled and texts, tables and constants were extracted in order to produce an assembler version of the interpreter. The size of this assembler file was approximately 4,000 lines. Studying the program, we found that the operation of the interpreter could be improved by rewriting certain lines of assembler code. Subsequently, a number of algorithms were developed and substituted for the ones originally implemented by Intel. Furthermore, errors found in a number of routines were corrected.

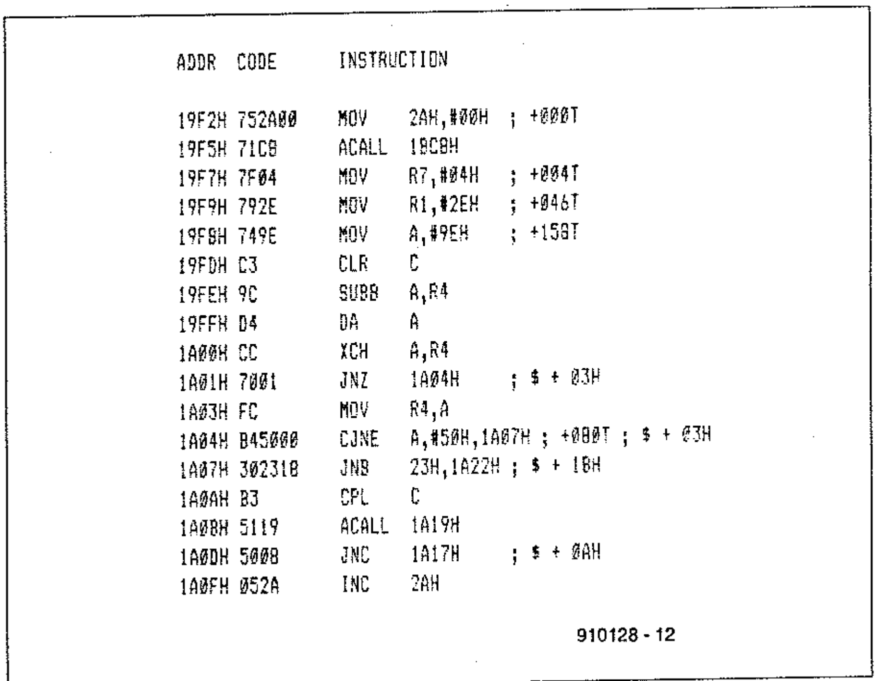
Floating point nucleus

One of the routines in the BASIC interpreter found to contain programming errors is the floating-point arithmetic nucleus. The errors can be demonstrated by running two small programs:

```
10   a=.10000001E30
20   b=.99999993E29
30   ?a–b
```

The result, 2.74E22, is erroneous, and should be 1.7E22. Similarly,

```
10   a=.10000001E30
20   b=.99999997E29
30   ?a–b
```



```
ADDR  CODE      INSTRUCTION

19F2H 752AØØ    MOV    2AH,#ØØH  ; +ØØØT
19F5H 71C8      ACALL  18C8H
19F7H 7FØ4      MOV    R7,#Ø4H   ; +ØØ4T
19F9H 792E      MOV    R1,#2EH   ; +Ø46T
19FBH 749E      MOV    A,#9EH    ; +158T
19FDH C3        CLR    C
19FEH 9C        SUBB   A,R4
19FFH D4        DA     A
1AØØH CC        XCH    A,R4
1AØ1H 7ØØ1      JNZ    1AØ4H     ; $ + Ø3H
1AØ3H FC        MOV    R4,A
1AØ4H B45ØØØ    CJNE   A,#5ØH,1AØ7H ; +Ø8ØT ; $ + Ø3H
1AØ7H 30231B    JNB    23H,1A22H ; $ + 1BH
1AØAH B3        CPL    C
1AØBH 5119      ACALL  1A19H
1AØDH 5ØØ8      JNC    1A17H     ; $ + ØAH
1AØFH Ø52A      INC    2AH
```

910128 - 12

Fig. 1. Original floating-point nucleus in the MCS BASIC-52 interpreter.

produces 1.34E22 instead of 1.3E22.

The disassembly listing of the original floating point nucleus developed by Intel is given in Fig. 1, and the version developed by the authors in Fig. 2. When implemented in the BASIC interpreter, the nucleus shown in Fig. 2 produces the correct answers to the above subtractions.

Other corrections

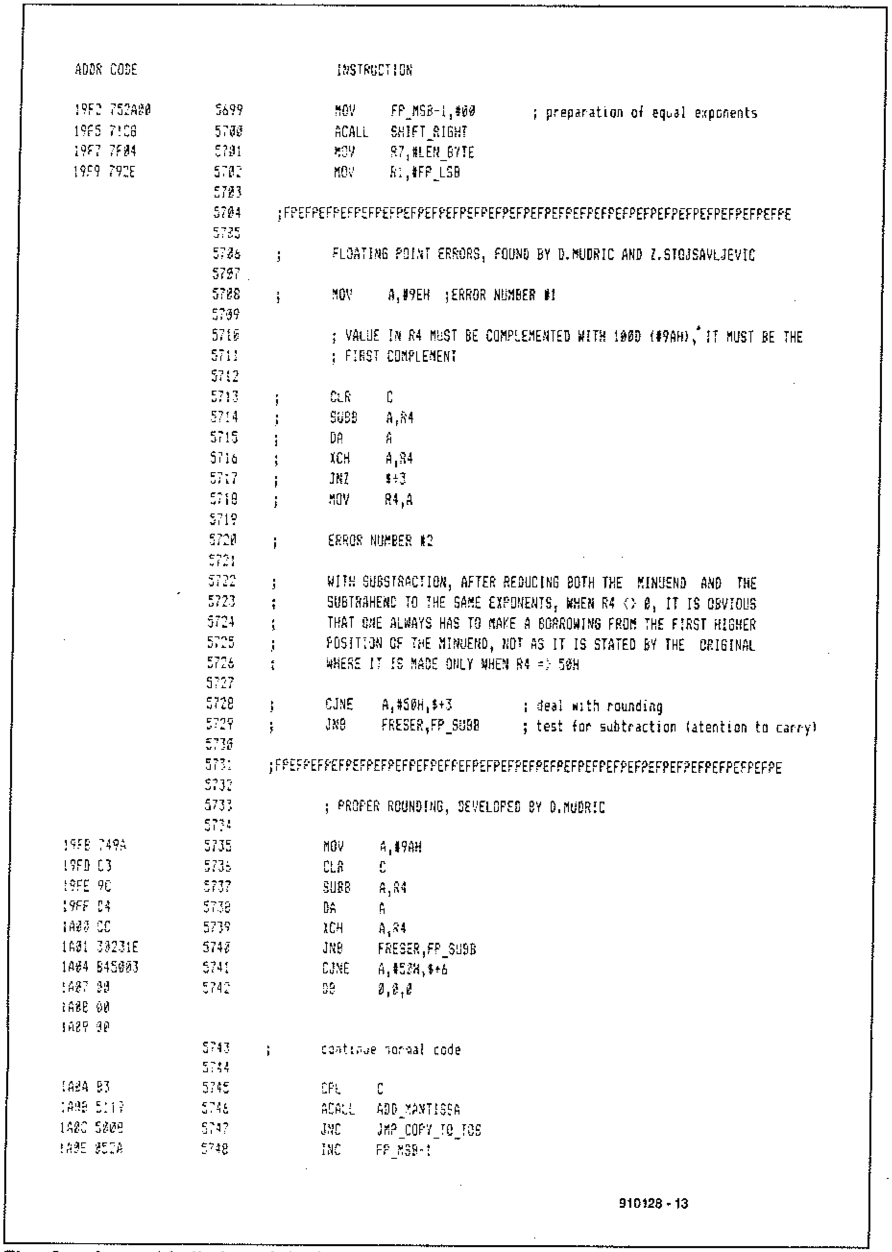
Further improvements were made to the hex-to-BCD conversion routine, both in regard of efficient programming and speed. For example, two approaches are possible for extracting BCD digits a, b, c, d, and e in

xyzwH = aD*10000D + bD*1000D + cD*100D + dD*10D + eD*1D

These possibilities are:

1. successive extraction of BCD digits starting with the most significant digit, a;
2. successive extraction of BCD digits starting with the least significant digit, e.

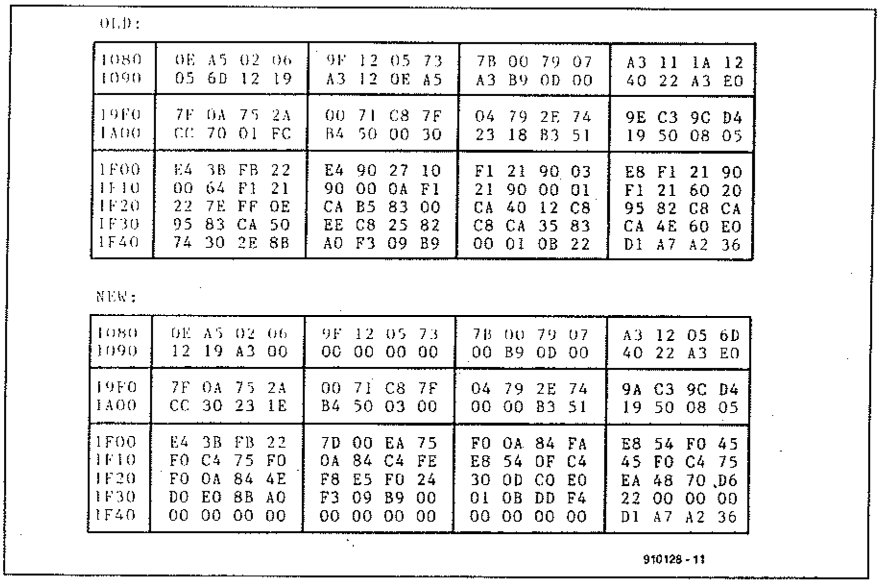
If the original version of the hex-to-BCD converter is studied, it is seen that the first procedure is employed. The DPTR is used as a 'weighted register', and the procedure is based on finding a suitably weighted subtraction number from a variable value.



```
ADDR CODE                    INSTRUCTION

19F2 752A00              MOV     FP_MSB-1,#00        ; preparation of equal exponents
19F5 7108                ACALL   SHIFT_RIGHT
19F7 7F04                MOV     R7,#LEN_BYTE
19F9 792E                MOV     R1,#FP_LSB

                   ;FPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPE

                   ;     FLOATING POINT ERRORS, FOUND BY D.MUDRIC AND Z.STOJSAVLJEVIC

                   ;     MOV     A,#9EH  ;ERROR NUMBER #1

                         ; VALUE IN R4 MUST BE COMPLEMENTED WITH 100D (#9AH), IT MUST BE THE
                         ; FIRST COMPLEMENT

                   ;     CLR     C
                   ;     SUBB    A,R4
                   ;     DA      A
                   ;     XCH     A,R4
                   ;     JNZ     $+3
                   ;     MOV     R4,A

                   ;     ERROS NUMBER #2

                   ;     WITH SUBSTRACTION, AFTER REDUCING BOTH THE  MINUEND  AND  THE
                   ;     SUBTRAHEND TO THE SAME EXPONENTS, WHEN R4 <> 0, IT IS OBVIOUS
                   ;     THAT ONE ALWAYS HAS TO MAKE A BORROWING FROM THE FIRST HIGHER
                   ;     POSITION OF THE MINUEND, NOT AS IT IS STATED BY THE  ORIGINAL
                   ;     WHERE IT IS MADE ONLY WHEN R4 => 50H

                   ;     CJNE    A,#50H,$+3          ; deal with rounding
                   ;     JNB     FRESER,FP_SUBB      ; test for subtraction (atention to carry)

                   ;FPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPEFPE

                         ; PROPER ROUNDING, DEVELOPED BY D.MUDRIC

19FB 749A                MOV     A,#9AH
19FD C3                  CLR     C
19FE 9C                  SUBB    A,R4
19FF D4                  DA      A
1A00 CC                  XCH     A,R4
1A01 30231E              JNB     FRESER,FP_SUBB
1A04 B45003              CJNE    A,#50H,$+6
1A07 00                  DB      0,0,0
1A08 00
1A09 00
                   ;     continue normal code

1A0A B3                  CPL     C
1A0B 5117                ACALL   ADD_MANTISSA
1A0D 5009                JNC     JMP_COPY_TO_TOS
1A0E 052A                INC     FP_MSB-1
```

910128 - 13

Fig. 2. Assembly listing of the improved floating-point nucleus.



OLD:

1080	0E A5 02 06	9F 12 05 73	7B 00 79 07	A3 11 1A 12
1090	05 6D 12 19	A3 12 0E A5	A3 B9 0D 00	40 22 A3 E0
19F0	7F 0A 75 2A	00 71 C8 7F	04 79 2E 74	9E C3 9C D4
1A00	CC 70 01 FC	B4 50 00 30	23 18 B3 51	19 50 08 05
1F00	E4 3B FB 22	E4 90 27 10	F1 21 90 03	E8 F1 21 90
1F10	00 64 F1 21	90 00 0A F1	21 90 00 01	F1 21 60 20
1F20	22 7E FF 0E	CA B5 83 00	CA 40 12 C8	95 82 C8 CA
1F30	95 83 CA 50	EE C8 25 82	C8 CA 35 83	CA 4E 60 E0
1F40	74 30 2E 8B	A0 F3 09 B9	00 01 0B 22	D1 A7 A2 36

NEW:

1080	0E A5 02 06	9F 12 05 73	7B 00 79 07	A3 12 05 6D
1090	12 19 A3 00	00 00 00 00	00 B9 0D 00	40 22 A3 E0
19F0	7F 0A 75 2A	00 71 C8 7F	04 79 2E 74	9A C3 9C D4
1A00	CC 30 23 1E	B4 50 03 00	00 00 B3 51	19 50 08 05
1F00	E4 3B FB 22	7D 00 EA 75	F0 0A 84 FA	E8 54 F0 45
1F10	F0 C4 75 F0	0A 84 C4 FE	E8 54 0F C4	45 F0 C4 75
1F20	F0 0A 84 4E	F8 E5 F0 24	30 0D C0 E0	EA 48 70 D6
1F30	D0 E0 8B A0	F3 09 B9 00	01 0B DD F4	22 00 00 00
1F40	00 00 00 00	00 00 00 00	00 00 00 00	D1 A7 A2 36

910128 - 11

Fig. 3. Overview of address locations to be modified in the 8-KByte interpreter. Remember, you can not overwrite data just like that in an EPROM — a 0 will remain a 0 unless you erase the EPROM using ultra-violet light. The modified version of the interpreter must, therefore, be loaded in a blank EPROM.

The second procedure is based on successive division of the variable value by 10, where, with each division, one BCD digit appears as the remainder.

By simplifying the conversion to successive division-by-10, the original code of 72 bytes is reduced to 57 bytes. The improved hex-to-BCD converter no longer requires the DPTR contents to be stored, returned and incremented after the conversion is finished. As a result, it is faster than the original converter implemented by Intel. Table 1 shows a few 'bench-mark' examples.

Table 1. Hex-to-decimal conversion time

P (hex)	P (dec)	t_{old} (ms)	t_{new} (ms)
0000	0000	141	56
0009	0009	276	56
4000	16384	507	268
EA5F	59999	820	268

Action!

If you are interested in the improvements made to the MCS BASIC-52 interpreter as described here, unload the interpreter from a 8052AH-BASIC, modify it, put it back into an EPROM and run it either with an 80C32, or as a turnkey-EPROM with an 8052AH-BASIC. All this is pretty straightforward and described at length in Refs. 1 and 2. First, copy the interpreter into an EPROM. Next, use the OLD function of the EPROM programmer on the BASIC computer (Refs. 4, 5) to make sure that an exact copy is available. Then load the contents of the EPROM into an EPROM programmer, and use the edit function to make the changes indicated in Fig. 3. Finally, program a new EPROM with the modified BASIC interpreter. The EPROM used may be a 27C64, a 27C128 or a 27C256. Alternatively, you may want to use an EEPROM Type 2864A. ❑

References:

1. "CMOS replacement for 8052AH-BASIC", Elektor Electronics January 1990.
2. "ROM-copy for 8052 BASIC computer",
3. "MCS BASIC-52 Users Manual", Intel Corp., 1986.
4. "BASIC computer", Elektor Electronics November 1987.
5. "8032/8052 Single-board computer", Elektor Electronics May 1991.

® MCS BASIC-52 is a registered trademark of Intel Corp.