In-Line Machine Language Subroutines for BASIC Interpreters

By John P. Newcomer

Robert Uiterwick's Floppy ROM* BASIC was an important addition to my SWTPC 6800 computer system, but I needed something more convenient than the USER function to permit machine-language operations. The need was met by what I call a PATsubroutine interpreter.

Floppy ROM BASIC doesn't care what you put after the PAT in PATCH so long as you don't overflow the input buffer. If you want to key in a sequence of machine-language instructions, you may. The trick, then, is to convert the ASCII character sequence into true machine code and execute it when the PATch command is executed by BASIC.

First you must intercept the jump to MIKBUG control (address E0E3) which results when BASIC sees a PATch command. This address is stored at location 08FE and must be changed to the address of the PATsubroutine interpreter. (You must also change the code at location 07F9 from 1200 to a value greater than the end address of the space required by your PATsubroutine interpreter.)

The interpreter as listed here stores the converted machine code beginning at location 1359, uses four bytes in the MIKBUG scratchpad for temporary storage of index register contents, calls a MIKBUG routine to aid in converting ASCII to hexadecimal, and jumps to MIKBUG control at E0E3 in case the conventional PATCH function is actually required.

Floppy ROM BASIC terminates the PATsubroutine with hexadecimal '1E', and at execution time location 34 contains the address of the first byte following the PATCH code in program storage (i.e., the first character of your PATsubroutine).

If you want to SAVE and LOAD a program containing a PATsubroutine, the maximum size of the subroutine is 62 characters or 31 bytes, which is adequate for many purposes. For other cases, successive PATsubroutines may be used, or a machine-language subroutine may be stored in high memory (say at location 1F00) using PATsubroutines and then invoked through the command PAT7E1F00.

Variables may be accessed by PATsubroutines by using the variable-storage pointer at location 2A. For example, if the first reference to a variable in your program is "A = 100", location 2A contains an address at which is stored the hexadecimal number 4120010000000003, which is the name of the variable "A" followed by the BCD mantissa and binary exponent of the value of A.

Following are some examples of PATsubroutines:

Store byte xx at location yyyy:

```
PAT86xxB7yyyy
```

Display on the terminal the contents of location yyyy:

```
PATCEyyyyBDE0CA
```

Test interrupt flag (bit 7 at location 801F in this example) to detect operator intervention request and reset flag:

```
PAT7D801F2A03BDE1ACB6801E
```

Add a key-entry value to variable A assuming that "A = 100" was the first reference to a variable in the program:

```
PATDE2ABDE1AC16270B4F8B011924026C025A26F6A703
```

Store a subroutine at 1F00:

```
PATCE1F00FF1311          (Change storage address)
PATaabbcc. . .           (Store subroutine)
PATCE1359FF1311          (Change storage address back)
    •
    •
    •
PAT7E1F00                (Execute subroutine)
```

A subroutine of any size can be stored in sections of 31 bytes or less by changing the storage address before each section is stored.

If you want to specify the conventional PATCH command, use "PAT". "PATCH" may not work unless you add a check for "CH" to the PATsubroutine interpreter listed here. If you are using MIKBUG to help with the conversion, "PATCH" will work by accident because the "H" in "PATCH", not being a valid hexadecimal digit, will cause a jump to CONTRL to signal an error condition.

It should be apparent that by using the scheme described above, complete machine-language flexibility is available as an integral part of your BASIC program without special provisions for loading and linking at run time. The concept should be adaptable to other BASIC interpreters and other CPUs, and it is even conceivable that an overlay structure could be effected by using PATsubroutines to load and execute subroutines as needed. □

*Floppy ROM is a trademark of INTERFACE AGE Magazine, Cerritos, CA 90701.

PROGRAM LISTING

```
00001                    NAM       PAT      SUBRT
00002                    *
00003                    * BY JOHN P NEWCOMER
00004                    * FOR FLOPPY ROM BASIC
00005                    *
00006                    OPT       NOG,P,NOO
00007      E0AC   INHEX  EQU       $E0AC
00008      E0E3   CNTRL  EQU       $E0E3
00009 1310               ORG       $1310
00010 1310 CE 1359       LDX       #$1359   SUBRT STORAGE ADD
00011 1313 FF A002       STX       $A002
00012 1316 DE 34         LDX       34H      1ST BYTE OF SUBRT
00013 1318 A6 00   ALPHA  LDA A     0,X
00014 131A 81 1E         CMP A     #$1E     END OF SUBRT ?
00015 131C 27 21         BEQ       BETA
00016 131E BD E0AC       JSR       INHEX    CONVERT
00017 1321 48            ASL A
00018 1322 48            ASL A              CONVERT TWO ASCII
00019 1323 48            ASL A              TO ONE HEX BYTE
00020 1324 48            ASL A
00021 1325 16            TAB
00022 1326 A6 01         LDA A     1,X
00023 1328 BD E0AC       JSR       INHEX
00024 132B 1B            ABA
00025 132C FF A004       STX       $A004
00026 132F FE A002       LDX       $A002
00027 1332 A7 00         STA A     0,X
00028 1334 08            INX
00029 1335 FF A002       STX       $A002
00030 1338 FE A004       LDX       $A004
00031 133B 08            INX
00032 133C 08            INX
00033 133D 20 D9         BRA       ALPHA
00034 133F 9C 34   BETA   CPX       34H      NULL SUB ?
00035 1341 26 03         BNE       CHECK    NO
00036 1343 7E E0E3       JMP       CNTRL    YES
00037 1346 FE A002 CHECK  LDX       $A002    PROVIDE FOR
00038 1349 86 BE         LDA A     #$BE     LOAD STACK
00039 134B A7 00         STA A     0,X
00040 134D 86 A0         LDA A     #$A0     POINTER AND
00041 134F A7 01         STA A     1,X      RETURN FROM INTERT
00043 1351 86 08         LDA A     #08
00044 1353 A7 02         STA A     2,X
00045 1355 86 3B         LDA A     #$3B
00046 1357 A7 03         STA A     3,X
00047                    END
```

```
S00B000050415420535542528B3
S11E1310CE1359FFA002DE34A600811E2721BDE0AC484848481A601BDE0ACD5
S11E132B1BFFA004FEA002A70008FFA002FEA0040808020D99C3426037EE0E310
S1161346FEA00286BEA700B6A0A7018608A702863BA70395
S9030000FC
```