BASIC5 strings

BY FR. THOMAS MCGAHEE

```
HI! WHAT'S YOUR NAME?? MARVIN GOLDFISH
NICE TO MEET YOU, MARVIN GOLDFISH
DO YOU HAVE ANY HOBBIES?? WHAT ARE THEY???
SWIMMING UNDERWATER
REALLY!!! I KNEW A GUY WHO LIKED SWIMMING UNDERWATER
BUT HE WASN'T TOO GOOD AT DOING ANYTHING.
WHO IS YOUR BEST FRIEND? JOHNNY FLOUNDER
DOES JOHNNY FLOUNDER LIKE SWIMMING UNDERWATER LIKE YOU?
WELL, MARVIN GOLDFISH IT'S BEEN NICE TALKING TO YOU.
I HOPE YOU COME BACK AND TALK WITH ME AGAIN SOMETIME.
BRING YOUR FRIEND, JOHNNY FLOUNDER WITH YOU.
```

Many SOL 20 owners have suffered along without string capabilities while waiting delivery of Processor Tech's 8K BASIC. But Father McGahee found time to write a string handler for BASIC5, so as to give his students capabilities for conversational-type programs such as the one illustrated on this page.

Our school recently purchased a SOL 20 from Processor Tech. I assembled it, and we are now using it in our computer course here at Don Bosco Tech. We have the 8K BASIC on order, but while we are waiting for that we have been happily programming away using BASIC5. One of the things that BASIC5 is missing is strings. Too bad, 'cause strings are lots of fun to use in programs to provide a more conversational feedback and `personal' sounding program.

I finally had a few free moments the other day (I teach electronics and computer programming at Don Bosco, and am kept fairly busy!!), and I wrote up this short string-handler which makes use of the machine language CALL instruction in BASIC5. It is by no means an optimum implementation, but provides a reasonable flexibility. I will be doing up a more useful version soon, but in the meantime I figured maybe the guys and gals at PCC might be interested in this first version. I guess there are a lot of SOLs out there with BASIC5, and not all of the users are capable of doing up their own string handlers... so they might like to try this one out until something better comes along.

I assembled my particular version starting at 4000 hex (16384 decimal). The assembler used was the ALS-8 from Processor Tech. I tried to keep things simple. To input an ASCII string the user does a CALL to ASCIN. This routine starts storage at the next available location in the text storage area, which is pointed to by LAST. It duplicates this address in BEG (for BEGINNING) for later use in setting the BC registers prior to a return to BASIC. I use the SOLOS input routine at 0C01F to get keyboard input, then I strip off the MSB (parity bit) since otherwise TTYs might give us codes different from some keyboards. The ASCII is then stored in memory and the current address updated to point to the next available location. At this time (before any echoing), a check is done to see if the ASCII character was a Line Feed (LF). I use the line feed as a terminator rather than Carriage Return (CR), because this allows the user to input extremely long strings, such as entire poems and the like!! If it was not a LF then the character is placed in the B register and echoed using the SOLOS routine at 0C019. Since the echo causes the A register to be changed, but B still has the ASCII code, we copy B into A so we can perform comparisons. A CR will result in a CR, LF, and one NULL being sent out. If the user has made a mistake, he may type in a DELETE, which will cause the program to back up the memory to the proper place. Input continues uninterrupted until a Line Feed is finally typed.

When input is done, the present address (next empty location) is stored in LAST so the next time ASCIN is used it will start off at the right place. The ORIGINAL BEGINNING of the present text string is then recovered from BEG and transferred to the B and C registers, since the BASIC CALL instruction uses these registers for transferring data between BASIC5 and the machine language routines. Then there is a RETurn to BASIC5. You will notice that there is a special entry point labeled INIT. Upon entry here the DONE portion of ASCIN is used to reset the address pointers to the beginning of the text storage area. This entry point can be used at the beginning of a BASIC program to 'clear' the string storage area. (Notice that it does not erase anything... it merely allows us to recycle storage space to conserve memory.)

The ASCII output routine operates by taking the address found in the B and C registers and setting that up as the current address for memory. (The B and C registers are loaded with the address prior to the BASIC CALL using the ARG instruction. . . see sample program for details). The program now starts extracting ASCII characters one at a time and printing them. A CR will again result in a CR, LF, and NULL, using the same subroutine used during input. When a Line Feed is finally encountered, there is a RETurn to BASIC5. The Line Feed is NOT printed.

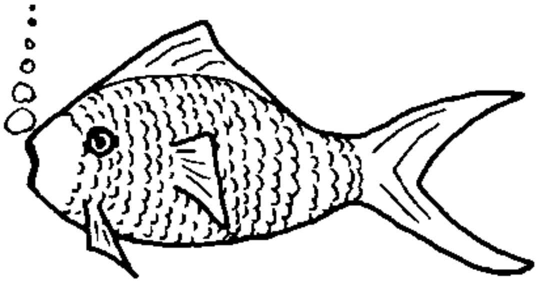
Three NOPs in the storage area are not necessary. I had them there to allow for quick `patches' should the need arise. It also prevents destruction of the program should too many DELETES be accidentally entered. One of the changes that I am making in the new version is a check to make sure the user does not delete beyond the BEGinning of the current string being input!!

The BASIC5 sample program listing shows one way of implementing strings using this machine language program and CALLs. The user must first load this string handler using SOLOS. What I am doing at present is have my students write three short subroutines in BASIC up at the high end, say at 10000, 20000, and 30000. These subroutines contain the necessary CALL and ARG statements to access the string handler. This way, instead of trying to remember the addresses needed for the CALL statements, all the student need remember is that GOSUB 10000 inputs a string, GOSUB 20000 extracts a string, and GOSUB 30000 resets the string storage area.

I have further chosen to arbitrarily use Z as the variable name under which all ARG and CALL transfers take place. This simplifies writing BASIC programs using the string handler, since there is only one variable name to be remembered. For example, to input a string which is to store a person's name, you can simply say: GOSUB 10000: N=Z. This inputs the string and stores the address of the string in variable N. To recover this specific string, simply: LET Z=N: GOSUB 20000 and the string is printed out!

One caution: no leading and trailing spaces are imbedded into the string unless the user enters them himself. What this means is that if you do not provide such spaces yourself inside the BASIC PRINT statements that may surround the output strings, you may find that the string is printed with no intervening spaces, and that looks messy. If you find this a bother, then modify the program to add such spaces automatically. On the other hand, I use the fact that there are no spaces to good advantage in a game where the user puts in a bunch of technical words, and then the program combines them in various ways to form some long technical-looking, mind-bending words.

In any case, the program is simple enough to be easily expanded. I can't wait to get my hands on Processor Tech's 8K BASIC, but in the meantime at least I have a limited string capability to play around with. Incidentally, I find the string handler useful for programs other than BASIC. As with anything, the uses are as broad as the user's imagination! So imagine to your heart's content, and have fun!



```
10 GOSUB 30000: REM * CLEAR STRING STORAGE AREA
20 PRINT "HI!  WHAT'S YOUR NAME?? ";: GOSUB 10000: N=Z
25 PRINT
30 PRINT "NICE TO MEET YOU. ";: Z=N: GOSUB 20000
35 PRINT
40 PRINT "DO YOU HAVE ANY HOBBIES?? WHAT ARE THEY???"
50 GOSUB 10000: H=Z
55 PRINT
60 PRINT "REALLY!!! I KNEW A GUY WHO LIKED ";: Z=H: GOSUB 20000
65 PRINT
70 PRINT "BUT HE WASN'T TOO GOOD AT DOING ANYTHING."
80 PRINT "WHO IS YOUR BEST FRIEND? ";: GOSUB 10000: F=Z
85 PRINT
98 PRINT "DOES ";: Z=F: GOSUB 20000: PRINT " LIKE ";
95 Z=H: GOSUB 20000: PRINT " LIKE YOU?"
100 PRINT "WELL. ";: Z=N: GOSUB 20000
110 PRINT " IT'S BEEN NICE TALKING TO YOU."
120 PRINT "I HOPE YOU COME BACK AND TALK WITH ME AGAIN SOMETIME."
130 PRINT "BRING YOUR FRIEND. ";: Z=F: GOSUB 20000: PRINT " WITH YOU."
140 PRINT : END
10000 Z=CALL(16384): RETURN
20000 Z=ARG(Z): Z=CALL(16442): RETURN
30000 Z=CALL(16430): RETURN
```

```
4000              0010 * MACHINE LANGUAGE ROUTINES TO ADD STRINGS
4000              0020 * TO BASIC5 VIA "CALL" INSTRUCTIONS.
4000              0025 * WRITTEN BY FR. THOMAS MCGAHEE
4000              0030 * ELECTRONICS AND COMPUTER INSTRUCTOR
4000              0035 * DON BOSCO TECH, PATERSON, NEW JERSEY 07502
4000              0040 *
4000              0100 *** ASCII INPUT WITH ECHO.
4000 2A 5A 40     0105 ASCIN  LHLD   LAST   RECOVER ADDRESS
4003 22 5C 40     0110        SHLD   BEG    STORE FOR LATER USE
4006 CD 1F C0     0115 INP    CALL   0C01FH  GET A CHARACTER
4009 CA 06 40     0120        JZ     INP    CHECK STATUS
400C E6 7F        0122        ANI    7FH    MASK PARITY BIT
400E 77           0125        MOV    M,A    STORE IN MEMORY
400E 23           0126        INX    H      UPDATE CURRENT ADDRESS
4010 FE 0A        0127        CPI    0AH    IF A LINE FEED...
4012 CA 31 40     0128        JZ     DONE   ...PREPARE TO RETURN
4015 47           0130        MOV    B,A    PUT IT IN B FOR SOLOS...
4016 CD 19 C0     0135        CALL   0C019H  ...SO IT CAN ECHO IT
4019 78           0140        MOV    A,B    IN "A" FOR COMPARES
401A FE 0D        0150        CPI    0DH    IF A CARRIAGE RETURN...
401C CC 4E 40     0155        CZ     CR     ...THEN DO LF AND NULL
401E FE 7F        0170        CPI    7FH    "DELETE" NEEDS HELP
```

```
4021 C2 06 40     0175         JNZ    INP    BACK FOR MORE!
4024 06 01        0185         MVI    B,01H  ...B HAS BACKSPACE...
4026 CD 19 C0     0190         CALL   0C019H  ...PRINT A BACKSPACE
4029 2B           0192         DCX    H      DOUBLE DECREMENT...
402A 2B           0193         DCX    H      ...CLEARS BAD DATA
4028 C3 06 40     0195         JMP    INP    ...AND GET MORE!
402E              0197 *
402E 21 63 40     0200 INIT   LXI    H,TXT  *RESET POINTERS
4031              0203 *
4031 22 5A 40     0205 DONE   SHLD   LAST   SAVE FOR NEXT TIME
4034 2A 5C 40     0210         LHLD   BEG    GET "ORIGINAL" ADDRESS..
4037 44           0215         MOV    B,H    ...AND STORE IN B,C
4038 4D           0220         MOV    C,L    ...FOR BASIC5 LINKAGE
4039 C9           0225         RET    BYE-BYE!
403A              0227 *
403A              0230 ***ROUTINE TO OUTPUT STORED ASCII STRINGS
403A 60           0235 ASCIO  MOV    H,8    TRANSFER ADDRESS ...
4038 69           0240         MOV    L,C    ...IN B,C TO H,L
403C 7E           0245 OUT    MOV    A,M    GET STORED CHARACTER
403D 47           0250         MOV    B,A    STORE IN B FOR NOW
403E FE 0A        0255         CPI    0AH    LF NOT PRINTED
4040 C8           0260         RZ     LF MEANS GO HOME!
4041 CD 19 C0     0265         CALL   0C019H  PRINT CHARACTER
4044 23           0270         INX    H      SET NEW ADDRESS
4045 78           0275         MOV    A,B    NEED IT IN "A"
4046 FE 0D        0280         CPI    0DH    CR NEEDS HELP
4048 CC 4E 40     0285         CZ     CR     SO HANDLE IT WITH CARE
4048 C3 3C 40     0290         JMP    OUT    GO FOR MORE OUTPUT
404E 06 0A        0295 CR     MVI    B,0AH  WITH A CR YOU GET...
4050 CD 19 C0     0300         CALL   0C019H  ...A FREE LINE FEED...
4053 06 00        0305         MVI    B,00H  ...AND A FREE NULL...
4055 CD 19 C0     0310         CALL   0C019H  ...TO ALLOW CLEAN I/O
4058 78           0320         MOV    A,B    NO TRASH, PLEASE
4059 C9           0325         RET    THAT'S ALL, FOLKS!
405A              0326 *
405A              0327 * STORAGE AREA FOLLOWS
405A 63 40        0330 LAST   DW     TXT    STORAGE
405C 63 40        0335 BEG    DW     TXT    STORAGE
405E 00           0340         NOP    FREE LOCATION
405E 00           0345         NOP    FREE LOCATION
4060 00           0350         NOP    FREE LOCATION
4061 00           0355         NOP    FREE LOCATION
4062 00           0360         NOP    FREE LOCATION
4063 00           0365 TXT    DB     00H    TEXT STORAGE BEGINS

ASCIN   4000
ASCIO   403A
BEG     405C     0110 0210
CR      404E     0155 0285
DONE    4031     0128
INIT    402E
INP     4006     0120 0175 0195
LAST    405A     0105 0205
OUT     403C     0290
TXT     4063     0200 0330 0335

4000: 2A 5A 40 22 5C 40 CD 1F C0 CA 06 40 E6 7F 77 23
4010: FE 0A CA 31 40 47 CD 19 C0 78 FE 0D CC 4E 40 FE
4020: 7F C2 06 40 06 01 CD 19 C0 2B 2B C3 06 40 21 63
4030: 40 22 5A 40 2A 5C 40 44 4D C9 60 69 7E 47 FE 0A
4040: C8 CD 19 C0 23 78 FE 0D CC 4E 40 C3 3C 40 06 0A
4050: CD 19 C0 06 00 CD 19 C0 78 C9 63 40 63 40 00 00
4060: 00 00 00 00
```