ML/BASIC Strings

String-Passing Between BASIC And Machine Language
by Rick Kephart 5/20/91

Here are my techniques for passing string variables from BASIC to ML, and from ML to BASIC. They are two entirely different methods, but they can be combined in the same program.

First, the means to pass a string variable from BASIC to ML:

First of all, make sure the variable you want to pass is the first variable in the program, which can be most easily done with the DIM statement (see below). If it's the very first variable, then you don't have to worry about hunting for the variable name, which greatly complicates things! This way, you'll know right where to find the information.

That information is stored immediately after the end of BASIC, which is at the location pointed to by zero page locations 45 & 46. So you can access the information using indirect addressing.

The first two bytes contain the variable name & type; you don't need these if you're only concerned about whatever variable is first. The third byte is the length of the string. The next two bytes are pointers to the location of the actual string in memory. There is no indirect indirect addressing, so you have to move those two bytes into particular zero-page locations so you can get to the string itself, then use indirect addressing from there to arrive at the string. Then you can use the length to count up to the length of the string (it makes things easier if this byte, too, is moved into some particular location).

Here is an example of code which will take a string variable, move it into location $2000 (it could be just about anywhere), mark the end of the string with a 0 byte, and then PRINT the variable:

```
C000  A0 02    LDY #$02
C002  B1 2D    LDA ($2D),Y ; get length of string
C004  85 FC    STA $FC     ; and store the length in $FC
C006  C8       INY
C007  B1 2D    LDA ($2D),Y ; get the low byte of string location
C009  85 FD    STA $FD     ; and move it to $FD
C00B  C8       INY
C00C  B1 2D    LDA ($2D),Y ; then move the high byte
C00E  85 FE    STA $FE

C010  A0 00    LDY #$00
```

```
C012  B1 FD    LDA ($FD),Y ; get the first byte of the string
C014  99 00 20 STA $2000,Y ; and move it into $2000
C017  C8       INY
C018  C4 FC    CPY $FC     ; check against string length
C01A  D0 F6    BNE $C012
C01C  A9 00    LDA #$00    ; mark the end of the string with 0
C01E  99 00 20 STA $2000,Y

C021  A0 00    LDY #$00
C023  B9 00 20 LDA $2000,Y ; get a byte of string from new area
C026  F0 06    BEQ $C02E   ; exit if 0 byte (end of string)
C028  20 D2 FF JSR $FFD2   ; PRINT
C02B  C8       INY
C02C  D0 F5    BNE $C023
C02E  60       RTS         ; back to BASIC
```

If the variable is the second or third variable defined by the program add multiples of 7 to the initial offset (i.e. the length of the second variable is 9 from the start-of-variables, so you would start with LDY #$09 instead of LDY #$02, for the third variable you'd use LDY #$10, etc.). This you could use if you need to move more than one variable at a time from BASIC to ML. Just make sure they're all defined at the beginning of the program, and in order. It's not worth all the effort it would take to hunt for the variable name which you would have to do if you didn't know where in the program the variable is first mentioned.

You can try it with this short program (with the ML in memory):

```
10 INPUT X$:SYS 49152
```

The string you INPUTed was moved to $2000, and then PRINTed.

Here is an easy way to move a string from ML to BASIC. To use my technique, you start by reserving an area of memory as a permanent place to put the string, the top 255 bytes (the maximum string length). To do this, make this your first line of BASIC (or at least before any variables have been used):

```
10 POKE 52,159:CLR:DIM A$
```

Location 52 normally contains 160 (52-52 points to the top of memory, normally 40960, but we're moving it down to 40704). The variable name in the DIM statement can be anything, but it must be a string variable, and whatever it is will be the variable into which the string will be placed.

Then, move the string in ML into 40704 and up. Put 159 and 0 into the variable pointers (at 45 & 46 + 3 and 45 & 46 + 4), and the length of the string into the BASIC variable string length (45 & 46 + 2), and the variable will contain your string!

Here is an example of code which will take a string in memory starting at 8192 (could be just about anywhere) and puts it into the first variable in any program:

```
C000  A0 03    LDY #$03     ; put the string location into
C002  A9 00    LDA #$00     ; the BASIC variable pointers
C004  91 2D    STA ($2D),Y
C006  C8       INY
C007  A9 9F    LDA #$9F     ; string will be at $9F00 (40704)
C009  91 2D    STA ($2D),Y

C00B  A0 00    LDY #$00
C00D  B9 00 20 LDA $2000,Y ; move a string from $2000 to $9F00
C010  F0 06    BEQ $C018   ; end of string was marked with 0 byte
C012  99 00 9F STA $9F00,Y
C015  C8       INY
C016  D0 F5    BNE $C00D

C018  98       TYA         ; Y still contains the length of the string
C019  A0 02    LDY #$02
C01B  91 2D    STA ($2D),Y ; so move it into the BASIC variable length
C01D  60       RTS
```

To try this out, start by putting any characters into memory starting at $2000 (such as with the M command of an ML monitor), ending with a 0 byte. Then try this:

```
10 POKE 52, 159:CLR:DIM X$
20 SYS 49152
30 PRINT X$
```

The string can be anywhere in memory, just make line C00D point to the beginning of the string.

The two can be combined. Start with the POKE 52 etc. line as above. Put the variable to be passed from BASIC into X$ (or whatever name you used in that line). The string will be passed from ML into that same variable name.

This is all for C-64 (or C-64 mode) only. It is much, much more complicated to do it in 128 mode, since it would involve bank switching.

You can write to me at .

website@lphrc.org