Combining BASIC And Machine-Language Programs On Tape

George Wells

This article describes a procedure to combine a machine-language program and a BASIC program into a single cassette tape file which can be LOADed and RUN without exiting BASIC. This procedure is specifically applied to a SYM-1, but the technique may be applicable to other machines, particularly Microsoft BASICs that store programs on tape in tokenized form exactly as they appear in memory.

General Discussion Of Technique

Whenever a BASIC programmer wants to jump to a machine-language program by way of the USR command, he has to decide where in RAM he is going to put the object code for the machine-language program. The usual place to put such code (assuming it is too big to squeeze into one of the unused areas on page zero or page one or some other place) is near the top of his contiguous RAM space which starts at page zero and includes at least 4K or 8K of memory. The method by which this is accomplished is to exit BASIC, load the object code from a file on tape, re-enter BASIC with an appropriate response to MEMORY SIZE? so BASIC will not use the memory allocated to the machine program and finally LOAD and RUN the BASIC program. In order to avoid this cumbersome procedure, we can put the two types of programs next to each other so that they can be LOADed together from one tape file into memory.

The technique to perform this is to make two tape files, the first one containing the BASIC program and the second one containing the machine code assembled somewhere in memory after the end of the BASIC program. Then all you have to do is enter BASIC, LOAD the BASIC program, LOAD the machine program, and SAVE the combined program. Now you have both programs on the same tape file which can be LOADed just like any other BASIC program. If you change the BASIC program, you will have to reLOAD the machine program and reSAVE the combined program. There are two pitfalls to be avoided when making changes. First, if the BASIC program expands to the point where it runs into the machine code, you will have to reassemble the machine program at a higher address, make a tape copy, modify the BASIC program to link properly to the new machine code, reLOAD the new machine code, and reSAVE the new combined program. Second, if you get a BAD LOAD error when trying to LOAD the machine code, your BASIC program will be deleted; so it's a good idea to SAVE the BASIC program after making any changes. In order to avoid these problems, you will probably want to assemble your machine-language program at the top of your RAM and check out your BASIC program as much as possible before combining the two programs together.

Specific Example On A SYM-1

This example will take a BASIC program that uses the trig functions and combine it with the machine code which the user must supply in order to use trig with the SYM-1 BASIC. It's a good idea to practice this technique on a simple BASIC program to get a feel for how it works before attempting a serious application.

STEP 1: Cold start to BASIC and enter the following program:

```
100 X = Y: REM CHANGE Y TO LAST
PAGE OF
TRIG.
110 POKE 196,104: POKE 197,X
120 PRINT SIN(1), COS(2), TAN(3),
ATN(4)
```

STEP 2: Save the BASIC program on tape with SAVE B.

STEP 3: Go to the monitor (by way of Reset) and look at memory locations $7D and $7E. These two values are the low and high bytes of the first available address after the BASIC program. The value of this address should be increased by at least 30 or 40 or even several hundred if extensive changes are expected in the BASIC program. In this example, we could safely start the machine code anywhere after address $0290.

STEP 4: Store the object code for the trig functions (from Synertek Systems, Inc. Technical Note 53) so that it ends at the end of page three.

STEP 5: Save the machine code on a second tape using an ID of $4D (ASCII "M") with the following command:

```
.S2 4D,2C7-3FF
```

STEP 6: Cold start back to BASIC and LOAD B to get the BASIC program.

STEP 7: Since we now know the location of the machine code, re-enter line 100:

```
100 X = 3: REM TRIG FUNCTIONS
END ON
PAGE 3.
```

STEP 8: Save the modified BASIC program on your first tape with SAVE B.

STEP 10: Enter LOAD M to load the machine code. If you get a LOADED message, go to STEP 12. If you get a BAD LOAD error message continue with STEP 11.

STEP 11. Reload the BASIC program with LOAD B and continue from STEP 10.

STEP 12: Save the combined program on a third tape with SAVE C. At this point, you can enter any valid BASIC command (try RUN and LIST) but when you get ready to modify the BASIC program continue from STEP 13.

STEP 13: Make as many changes as desired but DO NOT RUN the program.

STEP 14: Save the program on your first tape with SAVE B. This tape will now contain a valid BASIC program combined with invalid machine code. If you are sure that there is no danger of your BASIC program expanding into your machine code then continue from STEP 10. If you are not sure, continue with STEP 15.

STEP 15: There is no easy way to tell how big the BASIC part of the combined program is since the address at $7D, $7E is pointing somewhere near the end of the machine code. You could go to the monitor and manually search for three zero-bytes in a row which shouldn't be too hard if you have a general idea of where to look. Don't forget to insure that the system RAM is not write-protected after returning to BASIC. Another way to accomplish the same thing without leaving BASIC is to enter the following direct command (without spaces):

```
FORI = 515TO33333:IFPEEK(I-3) (   ) 0ORPEEK
(I-2) (   ) 0ORPEEK (I-1) (   ) 0THENNEXT
```

and wait for BASIC to respond with OK (it can take minutes). Then enter PRINT I and the computer will give the decimal equivalent of the first unused memory location after the BASIC program. If you run out of space between the two programs, reassemble the machine-language program at a higher address and continue at STEP 5. If you decide that you have sufficient space between the programs, you can continue at STEP 10.

NOTE: If at any time you suspect that the BASIC program has clobbered the machine program, you should reset your system, cold start to BASIC, LOAD B with the latest version of your program and continue at STEP 15.

NOTE: If you continue the trig functions with a BASIC program as in this example, you should take precaution to set the pointer at 196 and 197 back to its original value when leaving your program or avoid using any of the trig functions unless you properly re-attach the trig function object code. The original values of 196 and 197 are 2 and 208, respectively.

Theory Of Operation

The key to understanding how this technique works is in knowing the three ways that the Microsoft BASIC interpreter modifies the pointer to the start-of-variables ($7D and $7E in the SYM-1), and in realizing that the pointer to the start-of-program ($7B and $7C in the SYM-1) never gets modified once it is initialized by a cold start. In the SYM-1 the BASIC program always begins at location $0201 and there is a mandatory zero-byte at location $0200 which is put there only during cold start.

The first way that the interpreter modifies the start-of-variables pointer is through the NEW command which sets the pointer to a value that is equal to the start-of-program pointer plus two ($0203 in the SYM-1). This reduces the size of the BASIC program to two bytes which the NEW command clears to zeroes. In addition to being executed by a direct or indirect command or by a cold start, the NEW command is also automatically executed any time a tape LOAD command results in a BAD LOAD. This is why STEP 11 is required in the above example.

The second way the interpreter can modify the start-of-variables pointer is when a tape LOAD command results in the file being LOADED correctly. In this case, the pointer is set to one greater than the location of the last byte in the tape file and the other required pointers are updated with the NEW command. This is why it is possible to LOAD the machine code after the BASIC program and allow the interpreter to automatically adjust the pointers to allow you to SAVE and RUN the combined program. This is also why the Synertek Tech Note for using trig functions states that you must type either NEW or LOAD x after loading the file containing the trig object code into the top of your RAM space. If you didn't the variables would reside in non-existent RAM!

The third occasion in which the interpreter modifies the start-of-variables pointer is when a new line of a BASIC program is entered, although not in the way you might expect. After the interpreter finds the place in memory where the new line is to go, it calculates the change in the number of bytes that the new line will cause, either plus or minus. It then shifts memory by this amount beginning with the next line in the BASIC program and ending with the byte just before the start-of-variables. Next it updates the start-of-variables pointer by the same amount and then copies the new line into place. The important thing to note is that the interpreter is not influenced by the actual end of the BASIC program (the three zero-bytes) when it moves memory, so the machine code gets moved too. This is why it is necessary to reLOAD the machine code whenever a change is made in the BASIC program.

Conclusion

Now that you SYMmers know how easy it is to combine BASIC and machine-language programs, how about some neat utilities for BASIC? The rest of you can try this same technique on your own machines to see if it will work. Maybe someone with access to alot of different micros can publish a list of those that will and won't allow this technique to work. ©

SYM (AIM) Hi-Speed Tape Revisited

Gene Zumchak

Only a few days after I mailed in the article on SYM's high speed tape, and how loading might be improved by tweaking the value of HSPBDY, I received issue #3 of SYM-PHYSIS, the SYM Users Group newsletter. It contained an interesting note by Jay C. Sinnett, U.S. Environmental Protection Agency, South Ferry Road, Narragansett, R.I. 02882. He claimed that the volume range for loading SYM tapes could be expanded by making a hardware modification. Figure 1. shows his mod. He merely reconnected the clipping diodes so that clipping action does not occur until a diode drops above +5, and below ground. He explains that for many recorders, the amplitude of the positive and negative going peaks is not always equal, or constant. The diodes as connected allow charge to be trapped on C16 which changes the threshold point.

I made the change on my SYM and the results were amazing. Previously, I was only able to read in tapes with the volume level on my recorder at 7 plus or minus one-half. After the change, I could load from levels of 1 to 8. On another SYM, I was unable to load tapes at all. I made the change and was able to load tapes consistantly, and over a wide range of volume settings. Since the AIM and SYM tape circuits are similar, particularly in regard to the connection of the diodes, AIM users with marginal tape reading might also benefit from the mod.

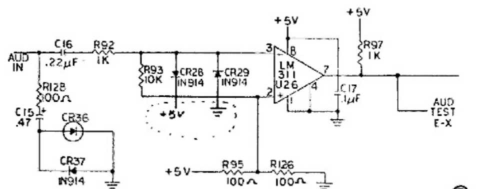

Figure 1. SYM Tape Hardware Mod ©