THE APPLE SCREEN

Getting Started With Subroutines

by Dan Ravey
127 Chukker Ct
San Mateo, CA 94403

If you've written even a few sizable programs, you probably already know how useful subroutines are. If you don't know, I'll bet your programs are overly long and complex. And if you're a beginner, well, just take my word for it!

The trouble is, many of us tend to create subroutines rather casually, assigning them line numbers that "seem right at the time," and we frequently use different constructs from one program to the next to accomplish the same results.

In this series of articles we will develop a method of building a library of standard subroutines with consistent line numbering and variables naming. The subroutines in the series will represent a substantial collection of fairly dramatic Screen Display controls to add professional zip to your programs.

Each subroutine will be saved as a Text File on diskette, so that in order to include any subroutine in your program, you may simply EXEC that file. This will insert the subroutine into your program — without the need for typing in all the subroutine lines. You can see right away that we must exercise some discipline about line numbering. Otherwise, we might replace existing lines of our program, or a previously inserted subroutine; or the inserted subroutine could end up in the middle of our program, even if it didn't replace lines with the same line numbers.

PLANNING SUBROUTINES

For reasonably short programs, or if speed of execution is unimportant, the placement of subroutines may be arbitrary, and perhaps the easiest plan is to assign line numbers much higher than we expect to use in our main program. But notice what the Applesoft Interpreter has to do when it encounters a GOSUB (or a GOTO, for that matter): after saving its current position, it goes back to the beginning of the program (normally, 2048, decimal) and chains its way up from the beginning of one line to the next, checking the line number each time, until it reaches the line number specified in the GOSUB or GOTO! So, if we place our subroutines early in the program, fewer lines will have to be checked to find the desired line.

We can't begin the program with a subroutine, of course, or it would execute whenever we RUN the program, (then "bomb out" when it finds the RETURN without there having first been a GOSUB). What we need to do is to jump over the subroutines with an unconditional GOTO, just before the subroutines.

THE GAME PLAN

Considering all of the above, here is the "Game Plan" I came up with: I use line numbers **under 100** for program I.D. REM statements, and any one-time initialization, such as DIMensioning arrays, defining Functions and constants, D$ = CHR$(4), etc. **Line 99** contains a **"GOTO 199" statement**, and 199 is a "REM" statement. Then **my main programs begin at 200** or higher. This leaves the line numbers 100 through 198 for subroutines. Any other systematic approach will yield equally good results. The main thing is **consistency**.

Once you have adopted a numbering scheme such as the above, the next step in building a subroutine library is to locate or write the subroutines you think you will have occasion to use, and look closely at the Applesoft code for each. You will be looking for **two things: (1)** the way each subroutine is line numbered; and **(2)** the names of the input and output variables.

HANDLING LINES

With respect to the line numbering, you will want your library to be expandable, so you should use as few line numbers per subroutine as possible. This means packing multiple statements per line number (being careful, however, of Applesoft IF statements, which skip the rest of the line if the argument is not true). Then, of course, you will want to assign the line numbers in the series you have chosen (100-198, in my case). If you have "PLE" or some other program editing capability, the job will be much easier.

STANDARD NAMES

I also recommend you try to standardize on the names of your input and output variables, to facilitate the use of the subroutines. You might always use **"V"** for a Real Value, and **"S$"** for a string, for example. Again, any letter will do, but BE CONSISTENT. Try to select names that you usually don't use elsewhere in your normal course of programming (such as X or N or A$).

CAPTURING SUBROUTINES

Okay, now, how do you put your subroutines into Text Files? You can use the "Capture" program in the DOS manual, but you'll have to rename each file from "LISTING" to whatever you want to call it. So, let's embellish "Capture" a bit, to make it handier for this purpose.

Listing 1 shows a modification that I call **CAPT SUBR**. The way this variation works is that it is really TWO programs disguised as one! When you RUN this, instructions are printed at the top of your screen. Then, in line 6, the command necessary to RUN the second part of the program (the one that actually captures the subroutine) is printed on the screen at line 11. Program line 8 then directs Applesoft to LIST line 20 on the screen at screen line 9. Yes, I know it says "VTAB 8", but the LIST command always prints a carriage return first, to insure that it starts on a fresh line.

So then we print "020" on the screen at line 9 — wait a minute! That's where the listing for line 20 goes! Yes, but we want the cursor to end up in a position where we can use the right arrow key to copy the listing of program line 20 — that's what the final VTAB 8 does, just before ENDing the program. But the Applesoft prompt symbol blots out the first character of the line number, SO — I had to "phoney" in the "20", beginning in column 2. (You're right — the first character doesn't have to be a "0", it can be anything at all.)

Okay, **the first part of the program ends at the end of line 8**, and the blinking cursor is right on the "2" of the listed line 20. If we **use the right arrow to copy through "20 LIST", then key in the line numbers that we want to capture in a Text File**, then press RETURN, line 20 (in the second part of the program) will be changed accordingly. Also, through CLEVER placement of the other printed line, the cursor will now be positioned so that you can copy through the command to run the rest of the program, beginning at line 10.

A DIGRESSION

Line 10, by the way, illustrates a more foolproof way of defining the DOS "waker-upper" (otherwise known as Control-D) than you may have seen used. To digress a moment: have you ever written a program that used Control-D DOS commands, and they wouldn't work, even though all the syntax looked correct? Try this:

```
10  D$ = CHR$(4)
20  PRINT "PRESS ANY KEY...";
30  GET A$
40  PRINT D$;"CATALOG"
50  END
```

That looks okay — but it won't work! Instead of cataloging your disk, it just prints "CATALOG" on the screen! How come? There are two culprits, either of which prevents DOS from recognizing the Control-D, which MUST be in the first column of a line. The semicolon at the end of the line 20 PRINT statement AND/OR the line 30 GET statement force the D$ of line 40 out of first place, as a PRINTed character. **A sure-fire cure is to always define D$ as a Carriage Return (ASCII 13) PLUS a Control-D (ASCII-4)**. The carriage return insures that Control-D will always be at the head of the "line". Change line 10, above, to be the same as line 10 in Listing 1, and — VOILA! It works!

BACK TO CAPTURE

Returning now to our Capture program, we prompt for the name to give to the Text File, then OPEN and enable WRITE to that file. Line 18 sets the text window so that extra spaces are not inserted in lines of over 40 characters. If you don't understand that, read about the text window in your Applesoft manual. Line 20 now contains whatever you input in the first part of the program, so the program now "LISTs" the subroutine, line numbers and all. But, since you enabled the DOS "WRITE", the lines are LISTed, not to the screen, but to your Text File. Finally, we CLOSE the file, return the text window to normal and signify completion.

Obviously, to be useful, we must have the subroutine lines in memory before we RUN this. So, the sequence for using this Capture program (once it has been entered, checked and SAVEd to disk) would be:

1. **LOAD CAPT SUBR**
2. **Enter lines of subroutine code, numbered somewhere in the 100 to 198 range (or your own choice)**
3. **RUN**
4. **Follow instructions to Run the second part**
5. **Repeat 2 through 4 as often as needed**

The next part in this series will present a number of subroutines you may want to have in your library, to EXEC into your programs. But to get you started, Listing 2 is the routine I assigned to line numbers 100-101, because it is used so frequently: "**Press Any Key To Continue**." As I do with all my Text File subroutines, I include the famous "99 GOTO 199" and "199 REM" lines, just to be sure that I don't forget to put them in my program.

My version of "Press Any Key" prints the prompt in inverse video on the bottom line of the screen, but if you don't want that, it is structured so that you may simply "GOSUB 101" instead of "GOSUB 100." If you are not familiar with the WAIT statement in line 101, look it up in your Applesoft manual. It's a useful statement that has been overlooked by many programmers.

The Text File names should be planned out, too. I indicate the subroutine line number in my file names, as well as a hint to its function. Thus, I have files named "SR100-PRESSKEY", "SR102-Y/N?", "SR110-CENTERPRINT", "SR113-BILLBOARD", "SR125-$ FORMAT", "SR140-FRACTIONS FORMAT", to name just a few that will appear in Parts 2 and 3 of this article.

```
]LIST

1  REM  * CAPTURE SUBROUTINE * BY
       DON RAVEY: COPYRIGHT (C)
       1982 BY MICRO-SPARC, INC.
       LINCOLN, MA. 01773
```

```
2  TEXT : HOME : INVERSE : PRINT
     " TO CAPTURE A SUBROUTINE IN
     A TEXT FILE,": NORMAL
4  PRINT "USING THE RIGHT ARROW K
     EY, COPY OVER THELINE NUMBER
      AND 'LIST', THEN ENTER THE"
     : PRINT "BEGINNING AND ENDIN
     G LINE NUMBERS OF THESUBROUT
     INE,  <CR>,  THEN COPY OVER
     THE": PRINT "'RUN 10' COMMAN
     D,  <CR>:"
6  VTAB 11: PRINT "]RUN 10"
8  VTAB 8: LIST 20: VTAB 9: PRINT
     "020": VTAB 8: END
10 D$ =  CHR$ (13) +  CHR$ (4)
12  TEXT : HOME : INPUT "ENTER FI
     LE NAME: ";FI$: PRINT : PRINT
     "DO NOT INTERRUPT WHILE WRIT
     ING FILE!"
14  PRINT D$;"OPEN";FI$
16  PRINT D$;"WRITE";FI$
18  POKE 33,30
20  LIST XXX,XXX
22  PRINT D$;"CLOSE";FI$
24  TEXT : PRINT "  DONE.": END
26  REM ------------------

]


]LIST

99  GOTO 199: REM  :: SUBROUTINES
      ::
100  VTAB 23: INVERSE : PRINT "
     PRESS ANY KEY TO CONTINUE..
     . ": NORMAL
101  WAIT  - 16384,128,1: POKE  -
     16368,0: RETURN
199  REM  :: MAIN PROGRAM ::

]
```