COMPUTE! ISSUE 54 / NOVEMBER 1984 / PAGE 163

Enhanced Commodore 64 DOS Support

Stephen S. Melsheimer

The wedge program that comes with every Commodore disk drive makes input/output much simpler. "Enhanced Commodore DOS Support" takes the wedge a few steps further by adding APPEND and VERIFY commands, allowing the use of wedge commands within a BASIC program, and several other features. There are also instructions on how you can further customize your wedge.

The Commodore 1541 disk drive used with the Commodore 64 and VIC-20 is *intelligent*—the disk unit contains a 6502 microprocessor, 16K of ROM holding the disk operating system (DOS), and 2K of RAM that serves as a buffer for information going to or from the disk. Thus, the computer is freed from the chore of managing the disk operations, and no computer memory is appropriated for a disk operating system when a disk drive is added.

Unfortunately, there are no simple commands in the *computer's* operating system to provide simple communication with the disk and its DOS. For example, there is no SCRATCH command to delete a file from the disk. Instead, you must use a cumbersome statement like:

```
OPEN 1,8,15, "S0 : filename" : CLOSE 1
```

To make up for this, Commodore provides a DOS support program named "DOS 5.1" (and a simpler "VIC-20 Wedge" for use with the VIC) to facilitate use of the disk. These programs appear on the *TEST/DEMO* disk which comes with the drive. This DOS wedge program is not needed to operate the disk, and adds no extra capabilities beyond those already present in the disk drive DOS ROM. What it provides is a set of shorthand commands that make things easier for the user. These commands do provide features that are vast improvements over what is possible directly from BASIC. In particular, the ability to display the disk directory without disturbing the program currently in memory is a great convenience.

The Theory Of Wedging

Programs like "DOS 5.1" are called *wedge* programs because they are wedged into the stream of BASIC interpreter processing. Central to the operation of BASIC is a subroutine called CHRGET (located at addresses $73-$8A in the 64 and VIC). This subroutine gets characters from a BASIC statement and delivers them to the interpreter. A wedge program intercepts each character and inspects it to see if it is a symbol recognized by the wedge. If not, control immediately returns to BASIC.

The "DOS 5.1" wedge also looks in the microprocessor stack and checks the return address of the BASIC routine which called CHRGET. If it is not an address which indicates the start of a new statement, control returns to BASIC. This allows the symbol characters to have their normal meaning in the middle of a statement. Finally, "DOS 5.1" checks whether BASIC is in direct or program mode, and exits to BASIC if in program mode. Obviously, all of this takes time—a couple of simple benchmark programs took about 15 percent longer to run when the

wedge was active, even though it has no useful effect in program mode. Thus, the wedge should be deactivated before running any program where execution speed is important. For those who are curious about the details of "DOS 5.1," Table 1 gives an abbreviated memory map to facilitate exploring it with a machine language monitor.

Extending "DOS 5.1"

"DOS 5.1" provides a table of symbols, and a list of associated vectors that point to the routines for the various functions. Functions can thus be added by altering a vector to point to the new function, and changing the corresponding symbol to the desired character. Since "DOS 5.1" has seven distinct functions, but provides eleven symbols (several are redundant), it is not necessary to delete any existing functions to add new ones. While "DOS 5.1" is very handy, I found that I wanted a few features that were not provided. The resulting "Enhanced DOS Support" program includes APPEND and VERIFY commands, provides a safety feature requiring user confirmation before erasing information on a disk, permits use of DOS commands within BASIC programs, and adds several other features. Still more commands could be added, and procedures for doing this are described below.

APPEND And VERIFY Commands

This revision of "DOS 5.1" originated because I wanted a simple procedure to append a BASIC program on disk to a BASIC program resident in memory. This would allow linking library subroutines or utility programs to other BASIC programs. Mark Niggemann presented an appending technique for the VIC-20 (COMPUTE!, March 1983) which is also applicable to the 64. All that is necessary to append a new program to an existing program is to determine the end address of the program in memory, and alter the starting address of the relocating loader accordingly. A LOAD command then executes as an APPEND. While this method is simple, it uses several direct-mode commands.

To automate the process, I made a patch in the LOAD section of the DOS support program at $CE26 (see the memory map, Table 1). The patch causes the jump to a new routine at $CF5F–$CF7E which alters the starting address in a manner similar to that presented by Niggemann. & was selected as the symbol to be replaced, so to append a program on disk with the name PROG 2 to a program in memory, one simply enters

```
&PROG2
```

It should be noted that this is not a *merge:* All statement numbers in the appended program must be higher than those in the program already in memory.

The ease with which APPEND was added encouraged me to add further enhancements to "DOS 5.1." Since the same 64 ROM routine handles both LOAD and VERIFY, it seemed reasonable to add a VERIFY command. The = character was assigned as the symbol. In order to produce the OK or VERIFY ERROR messages, a routine ($CE36–$CE6B) was written to check the error status byte for LOAD/VERIFY errors, and issue the appropriate messages. A typical application of VERIFY is

```
=PROG3
```

which compares PROG3 on disk with the program currently in memory.

The *disk* error status read by the @ command of "DOS 5.1" indicates errors detected within the 1541 disk drive and is completely independent

of the computer. Although it is essential to check the 64's status byte on VERIFY (and tape LOAD) operations, there is no great reason to check the status byte on disk LOAD operations, since most potential errors are monitored by the disk unit. However, Enhanced DOS Support reads the Commodore 64 error status on all LOAD/VERIFY operations, and does provide some useful messages even on disk LOADs (for example, BREAK ERROR).

Destruction Warnings

Certain disk commands may be disastrous if executed inadvertently—NEW erases an entire diskette, and SCRATCH deletes a file. BASIC 4.0 (used on larger Commodore PET/CBM computers) issues a warning on such commands, and requires user confirmation before executing them. Since "DOS 5.1" makes issuing commands to the disk very easy (and thus increases the prospect of careless errors), Enhanced DOS Support provides this safety feature for destructive commands issued via the DOS support program.

The portion of "DOS 5.1" ($CD48–$CD6E) that sends such commands to the disk was rewritten, and a new section was added at $CFBD–$CFEF that prints an ARE YOU SURE (Y/N)? message to the screen, and waits for a keyboard response. If N, the command is aborted. This revision also changed the scheme for decoding the commands for deactivating the wedge and for setting the device number. The effect was to free two vectors previously used, compensating for the two used for VERIFY and APPEND. In addition, the command for changing device numbers was simplified to @n rather than @#n.

Since SAVE&REPLACE also deletes a file, provision was made to issue the warning when the replace option is specified with SAVE operations. The revised SAVE routine is located at $CFF0–$CFFF.

Enabling Program Mode

While Enhanced DOS Support is mainly intended for direct-mode use, a few of the commands can be quite valuable in program mode. Thus, the portion of "DOS 5.1" ($CED1–$CED6) that prevented use of wedge commands within a program was deleted. Several other small changes were also needed to enable the commands to be used in program mode, and the LOAD/VERIFY routine ($CE36–$CE6B) was revised to facilitate program mode use of the % and & commands. With these changes some of the Enhanced DOS Support commands may be used within a program. The symbols =, ↑, and / are *not* allowed as DOS commands. In addition, the * and ? symbols cannot be used as wild cards in filenames while Enhanced DOS Support is active. In program mode these symbols are all tokenized by BASIC with codes other than the ASCII codes, and thus are not recognized by Enhanced DOS Support.

Of the commands that are operable in program mode, % is especially useful for loading machine language routines or screen images since it does not cause BASIC to restart as a LOAD command does. Thus, constructions like line 100 of Program 1 or line 10 of Program 2, which prevents repeated LOADing, are not needed. The & command was also designed to permit a program to automatically append BASIC subroutines, though using it is a little tricky. A procedure like

```
10 IF(PEEK(2)) < > 1THENPOKE2,1 : &SUBPROG
20 POKE2, 0
```

must be used since & restarts BASIC after it is executed, and also CLRs all BASIC variables. Obviously, this should normally be done at the very

beginning of the program. Memory location 2 is convenient to use as a flag since it is not used by the computer and has a value of zero on powerup. By using each bit of location 2 as a separate flag, up to eight subprograms could be appended in this manner. Another command that is valuable in program mode is @Q which deactivates the DOS wedge, thus speeding up program execution by about 15 percent. SYS52222 can activate the wedge from within a program, so it can be turned on only when needed.

Note that Enhanced DOS Support commands can be placed in multiple-statement lines, as illustrated in line 10 above. However, they must appear last on any line in which they are used, and thus only one DOS command can be placed on a line. Further, the Enhanced DOS Support symbol must be the first character of a statement for it to be recognized as such.

Other Changes

After SAVE operations, "DOS 5.1" reads the disk error status in order to verify a successful SAVE. A minor annoyance was that the disk status message appeared after the filename without any spacing or punctuation. Revision of $CD9C–$CDAE added a carriage return to provide a neater and cleaner error report.

The symbol table and list of vectors, located from $CC03 to $CC26, were revised considerably. The number of symbols was increased from 11 to 12, making the start of the symbol table $CC1B rather than $CC19 as given in the DOS 5.1 memory map of Table 1. Including the new & and = commands, seven distinct symbols have been used. For various reasons, > and ., both synonyms for (5), were added, thus using two more symbols. Since "Supermon64" (COMPUTE!, January 1983) was used extensively in developing Enhanced DOS Support, a symbol (!) was added that branches to Supermon64 (or to any other monitor that is entered via the break vector). Of course, the monitor must have previously been loaded. With ten symbols now assigned, two remain available for future use.

Command Summary

Table 2 lists the commands currently available in Enhanced DOS Support. Note that the . symbol is indicated for most disk command functions, while > is shown for device code changes and @ for reading the disk error status. This is strictly a matter of taste since all three (@, >, and .) are synonyms that perform exactly the same functions. Program 2 is a BASIC boot program that writes a command summary to the screen as Well as LOADing and activating Enhanced DOS Support (assuming you saved it with the program name used in line 10). Use Program 2 as you used "C-64 Wedge" to activate the original "DOS 5.1."

Entering The DOS 5.1 Enhancements

Program 1 will make all the necessary modifications to "DOS 5.1" to create the Enhanced DOS Support program. Before you run Program 1, make sure you have a disk containing the "DOS 5.1" program in the drive; otherwise, you will get the message FILE NOT FOUND ERROR IN 100. If any errors are detected in the DATA statements, the program will stop and report which lines contain errors. When Program 1 has finished, the cursor should be resting on a line of POKE statements. At this point you should insert the disk on which you want to store Enhanced DOS Support. Press RETURN, and the cursor should move to the line with the SAVE statement. Press RETURN again, and Enhanced DOS Support will be stored on the disk with the filename DOS 5.IE. At

this point you will need to turn the computer off and back on (or reset with SYS 64738) to return the system to its normal configuration.

If you want to take the easy way out, send a diskette containing "DOS 5.1" with $3 and a stamped, self-addressed mailer, and I will add the revisions for you.

Stephen S. Melsheimer
Clemson University
Department of Chemical Engineering
Clemson, SC 29631

To use Enhanced DOS Support, load it from disk by running Program 2. The original "C-64 Wedge" supplied with "DOS 5.1" can also be used, if the filename in line 10 is changed from DOS 5.1E to DOS 5.1.

Table 1: "DOS 5.1" Memory Map

$CC00	Wedge activation entry (52224)
CC03	symbol vectors
CC19	Symbol table
CC27	Text buffer
CC77	Device number
CC78	Filename suffix
CC7A	Current symbol
CC7B	"DOS MANAGER 5.1 …" text
CCE1	Wedge activation routine
CCF3	Normal entry point
CCF8	CHRGET call address validation ($A7E6 and $A48C are allowed)
CD0E	Check character against symbol table
CD30	Branch to execute routine if symbol matched
CD48	Execute @ commands
CD71	Send command string to disk (for example, @S0 : filename)
CD90	Read disk error status (@)
CDB2	List directory (@$)
CE22	Excute LOAD (including /, %, and ↑)
CE6C	Disable wedge (@Q)
CE79	Execute SAVE (←)
CE7E	Set device code (@#N)
CEA3	Process line into text buffer
CF4B	Print DOS MANAGER message
CF5B	End of "DOS 5.1"

Toble 2: Enhanced DOS Support Commands

$	Directory Listing
/filename	Load
↑filename	Load and run
& filename	Append
←filename	Save
←@0:filename	Save and replace
=filename	Verify
%filename	Absolute load
.S0 : filename	Scratch

.R0 : *newname = 0 : oldname*	Rename
.C0 : *newfile = 0 : oldfile, old2, … old4*	Copy/Concatenate up to four files
.N0 : diskname, id	Format disk
.I	Initialize disk
.V	Validate disk
@	Read disk error status
!	Break/Activate monitor
>n	Change disk device number to n
.Q	Disable DOS Support
SYS 52224	Reactivate DOS Support

(Note: ., >, and @ can be used interchangeably in any command beginning with one of these symbols.)

One caution: Commodore may in the future issue revisions of the DOS wedge. The enhancements given here may not work with versions different from the one on which it is based. If you get the message DOS MANAGER V5.1/071382 when you activate the original DOS 5.1, you have the correct version. If Program 1 checks out okay, but your program will not work, that may be the problem. The "easy out" mentioned above is still available in that case, of course.

Further Extensions

Two symbols remain unassigned in Enhanced DOS Support, and two of the three redundant command codes (@, >, and .) could be reassigned for other uses if needed. Thus, additional functions can easily be added to the program. One possibility is a command to link other utility programs (for example, a programmer's aid package) in a manner similar to the ! monitor link. Another handy addition might be a help routine that could display a summary of the Enhanced DOS Support commands on the screen without disrupting the program in memory. This could also include monitor or even BASIC commands as well.

To add a command, put the ASCII value of the desired symbol character in location $CC26. Then, put the high byte of the starting address of the new routine in $CC0E, and put the low byte of the target address *less one* in $CC1A. For a second command, decrement each of these addresses by one. The only other thing you must do is the hard part–writing the routine that will accomplish the new function. Locations $CF7F–$CFB1 are unused in DOS 5.IE, but extensive routines would have to be located outside the $CC00–$CFFF block. Remember that if a routine is to be used in program mode, the symbol must not be tokenized by BASIC (for example, do not use * as a symbol).

- Back to previous page
- See this article as it appeared in the magazine
- View this issue's table of contents