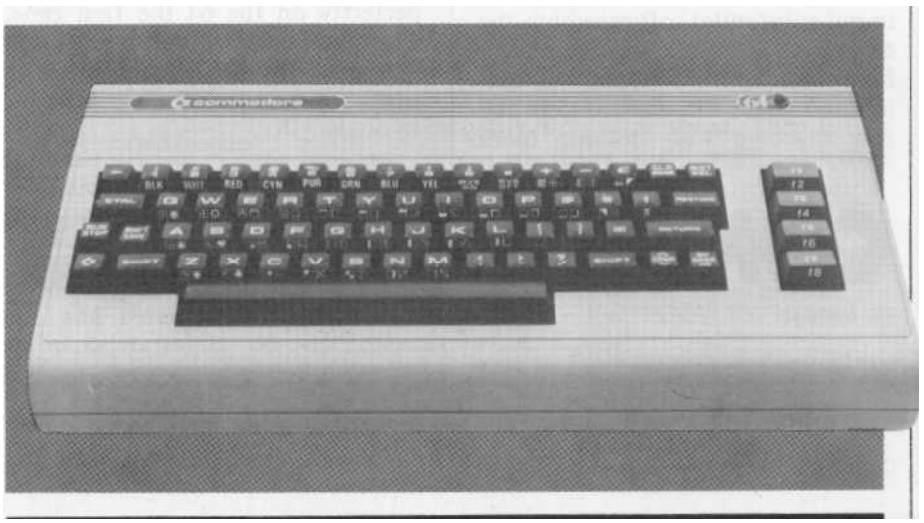
BASIC Programming on the COMMODORE 64



By the time you read this, the Commodore 64 will be on your dealer's shelf-if he or she can keep it there! As you know by now, it is a spectacular breakthrough in personal programming, offering more computer at less cost than anything Commodore has ever produced. And, needless to say, it is even more of an advance over the competition.

Programming the 64 should be a snap for anyone with experience on PET or VIC. For the most part, our familiar BASIC 2.0 will work exactly as programming for the older machines and for the Commodore 64. As we learn more, we'll make sure that we feed the information to you. The choice of BASIC 2.0 instead of 4.0 was made with some soul-searching, not just at random. The typical user of a 64 is not expected to need the direct disk commands as much as other extensions, and the amount of memory to be committed to BASIC was to be limited. We chose to leave expansion space for color and sound extensions instead of the disk features. As a result, you will have to handle the disk in the more cumbersome manner of the "old days." And, of course, we will have to put up with noticeable garbage collection, but that doesn't appear to be quite as much of a problem as it was in the older machines.

Of all the PEEKS and POKEs for the PET, the only one we need to be concerned with on the Commodore 64 is 59468. Of course, VIC-style code for literals and graphics is the preferred route, but if you want to be interoperable on the 64 and the "Fat 40" then you may want to POKE 53272,21 along with POKE 59468,12 and POKE 53272,23 along with POKE 59468,14. Neither poke hurts any machine, and the combination will do to any machine what 59468 did to the old ones.

Operating system changes may be important to the BASIC programmer, although many will be transparent to you. Two items that are apparent are the fact that files no longer routinely close on file errors and the RESTORE key can have anomalous results, especially when sound and sprites are in use. The file change will be helpful in debugging-your error channel will still be open after, for example, a "file not open." We're trying to be more specific about the RESTORE phenomenon, and will let you know when we do.

As many programmers have already learned one can program for relative files under BASIC 2.0, although it requires use of the command channel. We find that opening a relative file of record length n requires:

```
OPEN lf,8,sa,"0:REL FILE NAME,L," + CHR$(n)
```

instead of DOPEN#lf,"REL FILE NAME",Ln. Locating a record and byte under 2.0 requires separating the record number into high and low bytes (HI = int(#/256), LO = #-256*HI). Then the command channel (#15) is used to:

PRINT#15,"P" + CHR$(SA + 96) + CHRS$(60) + CHR$(HI) + CHR-$(BYTE) where SA is the secondary address of the file you're using and BYTE is the byte number you want (optional for byte 1). For reasons I don't understand, this set of commands seems trickier than those of 4.0 in handling record pointers. I recommend that you point to the

PROGRAMMER'S TIPS

record you want explicitly before and after every operation, just to be sure.

That's a summary of everything we've learned so far that we needed to put substantial software onto the 64 in BASIC. While we still have a lot to learn, it's encouraging that most software has moved up to the new machine with nothing more than these few items of change; most of it doesn't even need that much. In most cases, a 4040 disk or a cassette of your favorite PET program will probably load and run perfectly on the 64 the first time. Then, add color, sound, sprites, and whatever else you desire to move up to a whole new world of operation. C=

-Michael Richter