Commodore 16/Plus–4 Memory Maps

Jim Butterfield
Toronto, Ont.

I'm happy to see that Commodore have (I think for the first time) made an early publication of "official" RAM memory maps for their new machines. You can find them in the November/December 1984 issues of "Commodore Microcomputers" magazine, and they include something quite valuable: Commodore's internal "labels" by which they identify the locations. With these labels, an assembly language programmer can now use standard identifiers in writing a program – I'll certainly try to do so.

I've been picking apart the logic of the machines, and I'd like to offer my maps, too. They are similar in wording to previous maps I've published.

A few things that are noticeable about the new machines. They have a new architecture which calls for you to read the map more carefully.

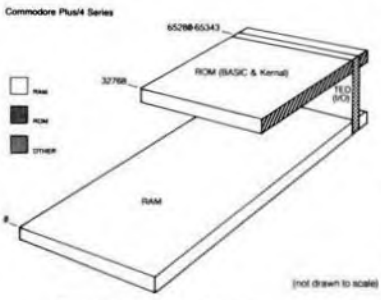

Diagram courtesy of Copp Clark Publishing

Since RAM lies partly beneath the ROM or ROMs above, we must understand that an address might refer to any of several "levels". For example: PRINTPEEK(32768) will give you a value from RAM at that location; but if you switch to the monitor and display the contents of hex 8000, you'll see the ROM and get quite a different value. For most applications (and for memory locations below 32768), you won't need to worry about all this. But when you get into the advanced stuff, you'll need to know how to work all the picky details.

A Few Differences

Inner space engineers will notice that much of zero page looks very much like that of the Commodore 64. The first big surprise is probably the CHRGET routine: it's missing from zero page. It turns out that CHRGET (which has relocated to page 4) needs to be used in a more complex way, since the calling routines must specify if they are looking for information from ROM or RAM.

And wonder of wonders: There's a whole block of spare memory in zero page, from at least $D8 to $E8. It's enough to disorient a programmer.

Extra space is put to a variety of uses. Page 1 is still mostly the system stack (see the note about the Basic stack, below). Page 2 is input areas as before, with space for working the new DOS commands. Page 3 contains links and vectors similar to those on the 64; watch these closely, since the absence of an NMI shortens up the table by one address.

Page 4 contains some communications buffering, and some replacements for the missing CHRGET routine. There's also a work area for PRINT USING and other activities. The current definitions of the programmable function keys take up much of page 5, and page 6 seems to be largely reserved for system expansion, such as speech synthesis.

There's a new stack mostly in page 7, the Basic stack, which holds loop and subroutine type information. FOR/NEXT, GOSUB/RETURN, and DO/LOOP with its WHILE and UNTIL provisions. It lets you write somewhat more complex programs, and leaves the machine stack less cluttered.

The screen is now accompanied by a sucessor to the "color nybble" table; it's called the "attribute" table. It's like the color nybbles, but contains extra information: not just color, but hue as well, and also a "flash" bit. It's in main memory now, residing at hex 0800 (decimal 2048), with the screen following it at hex 0C00 (decimal 3072). The extra space means that Basic starts higher than before, at hex 1000 or decimal 4096.

The ROM system is quite massive. At 32K, a map ends up lengthy even if it's abridged. A few surprises include: a built–in machine language monitor; graphics, error trapping and disk commands built into Basic; and a clever means of internally cataloging all the ROMs that happen to be fitted and making them available as desired.

The same old Kernal routines are still there and do the same job, but there's new stuff, including an "unofficial Jump Table" to handle some clever bank switching tasks.

It looks like a lot of fun. Good hunting. . ...