Apple II Computer System Information

ProDOS, the Apple III, and SOS

Source
ProDOS 8 Technical Reference Manual
http://linux.cis.monroeccc.edu/~paulrsm/6502/PDOS8TRM.HTM
28 May 2008

Appendix C
ProDOS, the Apple III, and SOS

This appendix explains the relationships between ProDOS, the Apple III, and SOS. It should be helpful to those already familiar with SOS and to those thinking about developing assembly-language programs concurrently for SOS and ProDOS.

C.1 - ProDOS, the Apple III, and SOS

As explained earlier in the manual, blocks 0 and 1 of a ProDOS-formatted disk contain the boot code -- the code that reads the operating system from the disk and runs it. Not explained was that this boot code runs on either an Apple II or an Apple III.

When you start up either an Apple II or an Apple III system with a ProDOS disk, the boot code is loaded at $800, and executed. The first thing it does is look to see whether it is running on an Apple II or Apple III. If it is running on an Apple II, it tries to load in the file PRODOS. If it is running on an Apple III, it tries to load in the file SOS.KERNEL. In either case, if the proper file is not found, it displays the appropriate error message.

This means that two versions of an application could be written, one for the Apple II, the other for the Apple III, and packaged together on the same disk. This single disk could be sold to both Apple II and Apple III owners.

C.2 - File Compatibility

SOS and ProDOS use the same directory structure: no exceptions. Every file on a ProDOS disk can be read by a SOS program and vice versa.

The file types that are used by both systems are directory files, text files, and binary files. These three types are adequate for the sharing of data between SOS and ProDOS versions of the same program.

File types that are intended for one system, but encountered on the other (as when you CATALOG a ProDOS disk using Business BASIC) are not inherently different from recognized file types; they just might cause a number to be displayed as their type instead of a name. The ProDOS BASIC system program, Filer, Conversion program, and Editor/Assembler all recognize and display names for all currently defined SOS file types. The abbreviations displayed when Apple III file types are encountered using ProDOS are shown in the quick reference section of this manual.

C.3 - Operating System Compatibility

Because of the larger amount of memory available to SOS, it is a much more complete operating system than is ProDOS. SOS has a complete and well defined file manager, device manager, memory manager, and interrupt and event handler. ProDOS has a file manager and simplified interrupt and memory calls.

C.3.1 - Comparison of Input/Output

SOS communicates with all devices -- the console, printers, disk drives, and so on -- by making open, read, write, and close calls to the appropriate device; writing to one device is essentially the same as writing to another. ProDOS can perform these operations on files only. Apple II peripherals generally have their driver code in ROM on the peripheral card. There is no consistent method for communicating with them. Thus the protocol for using any particular device must be known by the system program that is currently running.

C.3.2 - Comparison of Filing Calls

The set of calls to the ProDOS operating system is essentially a subset of the calls to SOS. All filing calls shared by the two systems have the same call number and nearly identical sets of parameters. Some differences are:

* With ProDOS you don't specify the file size when you create a file. Files are automatically extended when necessary.
* With SOS the GET_FILE_INFO call returns the size of the file in bytes (the EOF). In ProDOS you must OPEN the file and then use the GET_EOF call.
* The SOS VOLUME command corresponds to the ProDOS ON_LINE command. When given a device name, VOLUME returns the volume name for that device. When given a unit number (derived from the slot and drive), ON_LINE returns the volume name.
* For SOS, SET_MARK and SET_EOF can use a displacement from the current position. ProDOS uses only an absolute position in the file.

C.3.3 - Memory Handling Techniques

SOS has a fairly sophisticated memory manager: a system program requests memory from SOS, either by location or by amount needed. If the request can be satisfied, SOS grants it. That portion of memory is then the sole responsibility of the requestor until it is released.

A ProDOS system program is responsible for its own memory management. It must find free memory, and then allocate it by marking it off in a memory bit map. If a page of memory is marked in the bit map, ProDOS will not write data into that page. ProDOS can thus prevent users from destroying protected areas of memory (presumably all data is brought into memory using the ProDOS READ call).

C.3.4 - Comparison of Interrupts

In SOS, any device capable of generating an interrupt must have a device driver capable of handling the interrupt; the device driver and the interrupt handler are inseparable. ProDOS does not have device drivers; thus, interrupt handling routines are installed separately using the ALLOC_INTERRUPT call. Also, whereas SOS has a distinct interrupt priority for each device in the system, ProDOS must poll the routines one by one until someone claims the interrupt.

END