Apple II
File Type Notes

Developer Technical Support

File Type: $BD (189)
Auxiliary Type: All

Full Name: GS/OS File System Translator
Short Name: GS/OS File System Translator

Written by: Matt Deatherage September 1990

Files of this type and auxiliary type contain file system translators for GS/OS.

Files of type $BD contains file system translators, or FSTs. FSTs do not load if bit 15 of their auxiliary type is set.

GS/OS calls FSTs to interpret the physical file systems stored on block devices. By asking translation software to read the file system, GS/OS can read virtually any file system while having only an abstract file system assumed in the operating system code. Not all released file system translators are required, saving space on disk and in memory.

The format for FSTs is Apple confidential and subject to change with every system software release; Apple will release all future FSTs for GS/OS. Third-party developers may not create GS/OS FSTs—no documentation is available, and disassembly of the code for this purpose is not permitted. This is not an easy decision for Apple, which is a company that was built upon and operates with the goal to empower individuals through computing. Not revealing information isn't exactly consistent with this goal. There are, however, reasons for this policy.

First, FSTs are not as modular as they could be. Some GS/OS level changes require changes to all of the FSTs to be implemented. These changes range in magnitude from internal system service call changes to adding new parameters to existing calls. GS/OS is not tolerant of FSTs that do not know about such changes. The FST structure is straightforward, but it is also complex enough that disassembly of existing FSTs does not cover all the bases.

Second, it can create chaos for users. Two file system translators for a file system is far worse than none at all. No physical file system exactly matches the GS/OS abstract file system, so every FST must have file system specific behavior. Although some of these behaviors are well documented (parameters that do not fit in the abstract file system go in the `option_list`, for example), no two independently-designed FSTs for the same file system can possibly do such things identically.

For example, if there were two third-party DOS 3.3 FSTs available, each would have its own `FSTSpecific` subcalls, `option_list` parameters and other implementation differences.

Since there is only one `file_sys_ID` per file system, programs that create correct data structures for one DOS 3.3 FST may blow up with the other one.

If users somehow manage to figure this out, the only way to change FSTs is to enter the *:System:FSTs folder, deactivate one FST, activate another one and reboot, which is not acceptable. Even switching FSTs is unacceptable for archival and copying programs which may have stored `option_list` parameters embedded in files. Futhermore, if the file system is bootable, that makes boot blocks and a file system stub which are also tied to an FST, and users would have a horrible time changing those.

The best solution to these problems for Apple's customers (who are also your customers) is for Apple to maintain control over the development of file system translators. Apple will provide file system translators for other file systems. If you have requests for how certain features of any file system should be handled by future FSTs, please contact Developer Technical Support.

Further Reference

- *GS/OS Reference*