Apple II Miscellaneous
#14: Guidelines for Telecommunication Programs

Revised by: Matt Deatherage May 1992
Written by: Matt Deatherage July 1989

This Technical Note discusses recommended guidelines to ensure future compatibility and maintain workable standards for telecommunication programs.
Changes since July 1989: Rewritten to be more explicit in passages.

Telecommunication programs have always been a particularly troublesome area on the Apple II as far as standards are concerned. Exiting from terminal programs often leaves the system in an unbalanced state or leaves strange and unknown things upon the user's disks. Yet complying with standards would not only make life easier for the users, it's not that hard for developers to do. This Note lists the primary guidelines Apple II telecommunication program developers should keep foremost in their minds.

Talking to the Hardware

Communicating with the modem through the interface provided by the user isn't always the easiest task in the world. It often just can't be done at acceptable speeds when using high-level software routines, and sometimes it can't even be done at the firmware level. It's widely known that the Super Serial Card can't keep up with 9600 bps communication unless a low-level driver uses the 6551 chip on the card directly—the firmware just can't do it. The Apple IIGS serial port firmware can easily keep up with 9600 bps, but the GS/OS generated character drivers for those ports can't do single character I/O at that speed.

In general, programs **must** use the highest level interface available to them that functions to specifications. If dealing with speeds of less than 9600 baud in 16-bit mode, on the Apple IIGS, use the GS/OS drivers. That means if your terminal program uses both 4800 and 9600 baud, it should use the GS/OS drivers for 4800 baud and another method for 9600 baud—you cause more problems than you solve by using non-recommended methods for all speeds.

Remember that **any** GS/OS driver owns the slot or port it controls, and going around the drivers causes problems. High-speed, highly-configurable loaded drivers for the serial ports may ship with the System Software in the future, and it would be unfortunate if your terminal program was the one that caused the driver to break.

For speeds of 9600 bps or higher with System Software 6.0, the driver can't help you. It is necessary to go directly to the firmware or hardware and risk future incompatibility. Remember that the firmware must be called from bank zero emulation mode. If single character I/O isn't necessary, the driver can handle speeds of 9600 bps when used in multicharacter input or output.

Note: In the future, System Software may include loaded drivers for the serial ports. An application can tell whether a driver is generated or loaded by examining bit 14 of the `characteristics` word returned by the GS/OS `DInfo` call—a generated

driver has this bit set. A loaded driver may be able to handle 9600 bps single–character I/O, but a generated one may not.

File Transfer Considerations

Transferring files is probably the most important function of a telecommunication program. However, transferring the file's data itself is not always adequate. Telecommunication programs must find a way to transfer a file's **attributes** as well as a file's contents to keep things running smoothly.

File attributes include the file's type and auxiliary type (necessary fields for most applications to identify their data files), the size of the file, creation and modification dates and times, as well as information about how many forks the file has, what file system it came from, and how the file is stored on disk. In addition, when asked, GS/OS returns in its `option_list` information about the file that the native file system uses but GS/OS does not (information such as access privileges, native file types and creator types, parent directory IDs, extended attribute records and other information as important to the native file system as file type and auxiliary type are to GS/OS).

Any telecommunication program can devise a way to keep such attributes with a file when the file is transferred between two machines that are both running the program in question. It is a much trickier task to address the issue of keeping all file attributes with files **regardless** of the programs involved in the transfer. An industry-wide standard is necessary for such integration.

The Binary II standard, devised by Gary B. Little (and documented in the Apple II File Type Note for File Type $E0, Auxiliary Type $8000), has been accepted as a standard for maintaining these attributes for a number of years. Many major telecommunication programs already incorporate support for this standard; Apple urges those that don't to do so at their earliest convenience.

Binary II is designed to keep attributes with files on the fly—it is **not** an archival standard and should not be used as such. A standard like Binary II should **always** be used to keep attributes with a file; confusing it with an archival standard can result in files being transferred without their own attributes. Even archival files must be transferred with their attributes. It is **never** acceptable to transfer a file without it's attributes.

Archival considerations are a completely separate issue. An archival format and program must be carefully designed with archiving considerations in mind, such as manipulating files within the archive, preserving the attributes of the files archived, and allowing for a myriad of compression schemes. The NuFX standard (documented in the Apple II File Type Note for File Type $E0, Auxiliary Type $8002) is such an archival format, which Apple recommends be used for those purposes. The program ShrinkIt is an example of a NuFX archival utility.

In an ideal world, all files would be transferred with their attributes sent transparently by the telecommunication program. The user would select the file to send, and the program would automatically send the attributes. When the program receives a file, it would already have the attributes with the file, so no postprocessing by the user would be necessary to use the file.

Even archival files such as NuFX should be transferred with all attributes intact. Although the archival utility may allow the user to select any file for processing (in case the file's attributes were lost), assuming that this will happen implies that it's acceptable. It is not. No file should **ever** be transferred without all its attributes, down to, and including the GS/OS `option_list`, if present.

Apple IIGS Considerations

A few more guidelines for Apple IIGS-specific telecommunication applications follow:

- **Don't ignore slot configurations.** Attempting to use a serial port through hardware while an interface card for that slot is switched in will break dynamic slot arbitration if, and when, it becomes available, unless the application does not use the firmware.

- **Be a good neighbor to interrupt handlers.** Interrupts will be coming through that you did not enable. (This is true for Apple IIe computers with Workstation Cards, and is also true for IIGS computers even when AppleTalk is not involved.) Programs not prepared for this could bring the system down.

  Stealing main interrupt vectors is **not** a good idea. The main interrupt handler is already very tight code, and it runs in ROM 10% faster than any code you can replace it with in RAM. If you patch out the main interrupt vector and add more than about two instructions to the code path before returning to the ROM, AppleTalk will lose data. If you patch out the main interrupt vector, you make it impossible for Apple to add additional functionality by patching the same vector without breaking AppleTalk—and the vector is there for system software's convenience, not yours.

  I can't make it any plainer than this—**do not patch out the main interrupt vector unless it absolutely, positively cannot be avoided.** The only cases we know about where it absolutely can't be avoided are very high-speed communications from slot-based cards; high-speed serial communications from the serial ports can be handled by patching the serial interrupt vector (see Apple IIGS Technical Note #18, Do-It-Yourself SCC Interrupts). If you have to patch the main interrupt vector to run at 38400 bps, unpatch it when you switch to 2400 baud. **Only** patch the vector while it's absolutely necessary, and don't leave it continually patched just because it's easier. You're breaking things when you do that, whether your testing reveals it to you or not.

  If you must patch out the main interrupt vector, make it very clear to your users, both in the documentation and on-screen, that other system services like AppleTalk will not function and may not return until the computer is restarted. Give them a chance to back out.

- **Don't go stepping on things you don't own.** It is better to alert the user that a certain resource (like a slot or a port) is not available than to blindly switch it in and crash the system. Never switch slots without using the Slot Arbiter.

- **Behave yourself.** Don't make wild assumptions or do things differently just because you're a terminal program and you think you have to do it for speed. Most users won't be impressed by a terminal program that's fast and robust if it breaks every time they activate a desk accessory or if they have to reboot the system when they're done with it. Don't compromise system integrity for superficial functionality.

Further Reference

- *Apple IIGS Firmware Reference*
- *Apple IIGS Hardware Reference*
- Apple IIGS Technical Note #18, Do-It-Yourself SCC Interrupts
- Apple II File Type Notes, File Type $E0, Auxiliary Type $8000
- Apple II File Type Notes, File Type $E0, Auxiliary Type $8002