NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note DV14

SCSI Bugs

CONTENTS

There are a number of problems in the SCSI Manager; this note lists the ones we know about, along with an explanation of what we're doing about them. Changes made for the 2/88 release are made to more accurately reflect the state of the SCSI Manager. System 4.1 and 4.2 are very similar; one bug was fixed in System 4.2.

Updated: [July 01 1987]

SCSI Manager Problems

There are several categories of SCSI Manager problems:

1. Those in the ROM boot code (Before the System file has been opened, and hence, before any patches could possibly fix them.)
2. Those that have been fixed in System 3.2
3. Those that have been fixed in System 4.1/4.2
4. Those that are new in System 4.1/4.2
5. Those that have not yet been fixed.

The problems in the ROM boot code can only be fixed by changing the ROMs. Most of the bugs in the SCSI Manager itself have been fixed by the patch code in the System 3.2 file. There are a few problems, though, that are not fixed with System 3.2 - most of these bugs have been corrected in System 4.1/4.2. Any that are not fixed will be detailed here. ROM code for future machines will, of course, include the corrections.

Back to top

ROM boot code problems

In the process of looking for a bootable SCSI device, the boot code issues a SCSI bus reset before each attempt to read block 0 from a device. If the read fails for any reason, the boot code goes on to the next device. SCSI devices which implement the `Unit Attention` condition as defined by the Revision 17B SCSI standard will fail to boot in this case. The read will fail because the drive is attempting to report the `Unit Attention` condition for the first command it receives after the SCSI bus reset. The boot code does not read the sense bytes and does not retry the failed command; it simply resets the SCSI bus and goes on to the next device.

If no other device is bootable, the boot code will eventually cycle back to the same SCSI device ID, reset the bus (causing `Unit Attention` in the drive again), and try to read block 0 (which fails for the same reason).

The `new' Macintosh Plus ROMs that are included in the platinum Macintosh Plus have only one change. The change was to simply do a single SCSI Bus Reset after power up instead of a Reset each time through the SCSI boot loop. This was done to allow `Unit Attention` drives to be bootable. It was an object code patch (affecting approximately 30 bytes) and no other bugs were fixed. For details on the three versions of Macintosh Plus ROMs, see Technical Note #154.

We recommend that you choose an SCSI controller which does not require the `Unit Attention` feature--either an older controller (most of the SCSI controllers currently available were designed before Revision 17B), or one of the newer Revision-17B-compatible controllers which can enable/disable `Unit Attention` as a formatting option (such as those from Seagate, Rodime, et al). Since the vast majority of Macintosh Plus computers have the ROMs which cannot use `Unit Attention` drives, we still recommend that you choose an SCSI controller that does not require the `Unit Attention` feature.

- If an SCSI device goes into the `Status` phase after being selected by the boot code, this leads to the SCSI bus being left in the `Status` phase indefinitely, and no SCSI devices can be accessed. The current Macintosh Plus boot code does not handle this change to `Status` phase, which means that the presence of an SCSI device with this behavior (as in some tape controllers we've seen) will prevent any SCSI devices from being accessed by the SCSI Manager, even if they already had drivers loaded from them. The result is that any SCSI peripheral that is turned on at boot

time must not go into Status phase immediately after selection; otherwise, the Macintosh Plus SCSI bus will be left hanging. Unless substantially revised ROMs are released for the Macintosh Plus (highly unlikely within the next year or so), this problem will never be fixed on the Macintosh Plus, so you should design for old ROMs.

- The Macintosh Plus would try to read 256 bytes of blocks 0 and 1, ignoring the extra data. The Macintosh SE and Macintosh II try to read 512 bytes from blocks 0 and 1, ignoring errors if the sector size is larger (but not smaller) than 512 bytes. Random access devices (disks, tapes, CD ROMS, etc.) can be booted as long as the blocks are at least 512 bytes, blocks 0, 1 and other partition blocks are correctly set up, and there is a driver on it. With the new partition layout (documented in *Inside Macintosh* volume V), more than 256 bytes per sector may be required in some partition map entries. This is why we dropped support for 256-byte sectors. Disks with tag bytes (532-byte sectors) or larger block sizes (1K, 2K, etc.) can be booted on any Macintosh with an SCSI port. Of course, the driver has to take care of data blocking and de-blocking, since HFS likes to work with 512-byte sectors.

Back to top

Problems with ROM SCSI Manager routines

Note that the following problems are fixed after the System file has been opened; for a device to boot properly, it must not depend on these fixes. The sample SCSI driver contains an example of how to find out if the fixes are in place.

- **Prior to System file 3.2,** blind transfers (both reads and writes) would not work properly with many SCSI controllers. Since blind operation depends on the drive's ability to transfer data fast enough, it is the responsibility of the driver writer to make sure blind operation is safe for a particular device.
- **Prior to System file 3.2,** the SCSI Manager dropped a byte when the driver did two or more SCSIReads or SCSIRBlinds in a row. (Each Read or RBlind has to have a Transfer Information Block (TIB) pointer passed in.) The TIB itself can be as big and complex as you want--it is the process of returning from one SCSIRead or SCSIRBlind and entering another one (while still on the same SCSI command) that causes the first byte for the other SCSIReads to be lost.
- Note that this precludes use of file-system tags. Apple no longer recommends that you support tags; see Technical Note #94 for more information.
- **Prior to System file 3.2,** SCSIStat didn't work; the new version works correctly.
- **Running under System file 3.2,** the SCSI Manager does not check to make sure that the last byte of a write operation (to the peripheral) was handshaked while operating in pseudo-DMA mode. The SCSI Manager writes the final byte to the NCR 5380's one-byte buffer and then turns pseudo-DMA mode off shortly thereafter (reported to be 10-15 microseconds). If the peripheral is somewhat slow in actually reading the last byte of data, it asserts REQ after the Macintosh has already turned off pseudo-DMA mode and never gets an ACK. The CPU then expects to go into the Status phase since it thinks everything went OK, but the peripheral is still waiting for ACK. Unless the driver can recover from this somehow, the SCSI bus is `hung' in the Data Out phase. In this case, all successive SCSI Manager calls will fail until the bus is reset.
- **Running under System file 4.1/4.2,** the SCSI Manager waits for the last byte of a write operation to be handshaked while operating in pseudo-DMA mode; it checks for a final DRQ (or a phase change) at the end of a SCSIWrite or SCSIWBlind before turning off the pseudo-DMA mode. Drivers that could recover from this problem by writing the last byte again if the bus was still in a Data Out phase will still work correctly, as long as they were checking the bus state.
- **Running under System file 3.2,** the SCSI Manager does not time out if the peripheral fails to finish transferring the expected number of bytes for polled reads and writes. (Blind operation does poll for the first byte of each requested data transfer in the Transfer Information Block.)
- **Running under System file 4.1/4.2,** SCSIRead and SCSIWrite return an error to the caller if the peripheral changes the bus phase in the middle of a transfer, as might happen if the peripheral fails to transfer the expected number of bytes. The computer is no longer left in a hung state.
- **Running under System file 3.2,** the Selection timeout value is very short (900 microseconds). Patches to the SCSI Manager **in System 4.1/4.2** ensure that this value is the recommended 250 milliseconds.
- **Running under System file 3.2,** the SCSI Manager routine SCSIGet (which arbitrates for the bus) will fail if the BSY line is still asserted. Some devices are a bit slow in releasing BSY after the completion of an SCSI operation, meaning that BSY may not have been released before the driver issues a SCSIGet call to start the next SCSI operation. A work-around for this is to call SCSIGet again if it failed the first time. (Rarely has it been necessary to try it a third time.) This assumes, of course, that the bus has not been left `hanging' by an improperly terminated SCSI operation before calling SCSIGet.
- **Running under System file 4.1/4.2,** the SCSIGet function has been made more tolerant of devices that are slow to release the BSY line after a SCSI operation. The SCSI Manager now waits up to 200 milliseconds before returning an error.

Back to top

Problems with the SCSI Manager that haven't been fixed yet

These problems currently exist in the Macintosh Plus, SE, and II SCSI Manager. We plan to fix these problems in a future release of the System Tools disk, but in the mean time, you should try to work around the problems (but don't "require" the problems!).

- Multiple calls to SCSIRead or SCSIRBlind after issuing a command and before calling SCSIComplete may not work. Suppose you want to read some mode sense data from the drive. After sending the command with SCSICmd, you might want to call SCSIRead with a TIB that reads four bytes (typically a header). After reading the field (in the four-byte header) that tells how many remaining bytes are available, you might call SCSIRead again with a TIB to read the remaining bytes. The problem is that the first byte of the second SCSIRead data will be lost because of the way the SCSI Manager handles reads in pseudo-DMA mode. The work-around is to issue two separate SCSI commands: the first to read only the four-byte header, the second to read the four-byte header plus the remaining bytes. We recommend that you not use a clever TIB that contains two data transfers, the second of which gets the transfer length from the first transfer's received data (the header). These two step TIBs will not work in the future. This bug will probably not be fixed.

- On read operations, some devices may be slow in de-asserting REQ after sending the last byte to the CPU. The current SCSI Manager (all machines) will return to the caller without waiting for REQ to be de-asserted. Usually the next call that the driver would make is SCSIComplete. On the Macintosh SE and II, the SCSIComplete call will check the bus to be sure that it is in Status phase. If not, the SCSI Manager will return a new error code that indicates the bus was in Data In/Data Out phase when SCSIComplete was called. The combination of the speed of the Macintosh II and a slow peripheral can cause SCSIComplete to detect that the bus is still in Data In phase before the peripheral has finally changed the bus to Status phase. This results in a false error being passed back by SCSIComplete.

- The scComp (compare) TIB opcode does not work in System 4.1 on the Macintosh Plus only. It returns an error code of 4 (bad parameters). This has been fixed in System 4.2.

Back to top

Other SCSI Manager Issues

- At least one third-party SCSI peripheral driver used to issue SCSI commands from a VBL task. It didn't check to see if the bus was in the free state before sending the command! This is guaranteed to wipe out any other SCSI command that may have been in progress, since the SCSI Manager on the Macintosh Plus does not mask out (or use) interrupts.

We strongly recommend that you avoid calling the SCSI Manager from interrupt handlers (such as VBL tasks). If you must send SCSI commands from a VBL task (like for a removable media system), do a SCSIStat call first to see if the bus is currently busy. If it's free (BSY is not asserted), then it's probably safe; otherwise the VBL task should not send the command. Note that you can't call SCSIStat before the System file fixes are in place. Since SCSI operations during VBL are not guaranteed, you should check all errors from SCSI Manager calls.

- A new SCSI Manager call will be added in the future. This will be a high-level call; it will have some kind of parameter block in which you give a pointer to a command buffer, a pointer to your TIB, a pointer to a sense data buffer (in case something goes wrong, the SCSI Manager will automatically read the sense bytes into the buffer for you), and a few other fields. The SCSI Manager will take care of arbitration, selection, sending the command, interpreting the TIB for the data transfer, and getting the status and message bytes (and the sense bytes, if there was an error). It should make SCSI device drivers much easier to write, since the driver will no longer have to worry about unexpected phase changes, getting the sense bytes, and so on. In the future, this will be the recommended way to use the SCSI Manager.

- The SCSI Manager (all machines) does not currently support interrupt-driven (asynchronous) operations. The Macintosh Plus can never support it since there is no interrupt capability, although a polled scheme may be implemented by the SCSI Manager. The Macintosh SE has a maskable interrupt for IRQ, and the Macintosh II has maskable interrupts for both IRQ and DRQ. Apple is working on an implementation of the SCSI Manager that will support asynchronous operations on the Macintosh II and probably on the SE as well. Because the interrupt hardware will interact adversely with any asynchronous schemes that are polled, it is strongly recommended that third parties do not attempt asynchronous operations until the new SCSI Manager is released. Apple will not attempt to be compatible with any products that bypass some or all of the SCSI Manager. In order to implement software-based (polled) asynchronous operations it is necessary to bypass the SCSI Manager.

The SCSI Manager section of the alpha draft of *Inside Macintosh* volume V documented the Disconnect and Reselect routines which were intended to be used for asynchronous I/O. Those routines cannot be used. Those routines have been removed from the manual. Any software that uses those routines will have to be revised when the SCSI Manager becomes interrupt-driven. Drivers which send SCSI commands from VBL tasks may also have to be modified.

Back to top

Hardware in the SCSI

There is some confusion on how many terminators can be used on the bus, and the best way to use them. There can be no more than two terminators on the bus. If you have more than one SCSI drive you must have two terminators. If you only have one drive, you should use a single terminator. If you have more than one drive, the two terminators should be on opposite ends of the chain. The idea is to terminate both ends of the wire that goes through all of the devices. One terminator should be on the end of the system cable that comes out of the Macintosh. The other terminator would be on the very end of the last device on the chain. If you have an SE or II with an internal hard disk, there is already one terminator on the front of the chain, inside the computer.

On the Macintosh SE and II, there is additional hardware support for the SCSI bus transfers in pseudo-DMA mode. The hardware makes it possible to handshake the data in Blind mode so that the Blind mode is safe for all transfers. On the Macintosh Plus, the Blind transfers are heavily timing dependent and can overrun or underrun during the transfer with no error generated. Assuring that Blind mode is safe on the Macintosh Plus depends upon the peripheral being used. On the SE and II, the transfer is hardware assisted to prevent overruns or underruns.

Back to top

Changes in SCSI for SE and II

The changes made to the SCSI Manager found in the Macintosh SE and Macintosh II are primarily bug fixes. No new functionality was added. The newer SCSI Manager is more robust and has more error checking. Since the Macintosh Plus SCSI Manager only did limited error checking, it is possible to have code that would function (with bugs) on the Macintosh Plus, but will not work correctly on the SE or II. The Macintosh Plus could mask some bugs in the caller by not checking errors. An example of this is sending or receiving the wrong number of bytes in a blind transfer. On the Macintosh Plus, no error would be generated since there was no way to be sure how many bytes were sent or received. On the SE and II, if the wrong number of bytes are transferred an error will be returned to the caller. The exact timing of transfers has changed on the SE and II as well, since the computers run at different speeds. Devices that are unwittingly dependent upon specific timing in transfers may have problems on the newer computers. To find problems of this sort it is usually only necessary to examine the error codes that are passed back by the SCSI Manager routines. The error codes will generally point out where the updated SCSI Manager found errors.

Back to top

To report other bugs or make suggestions

Please send additional bug reports and suggestions to us at the address in Technical Note #0. Let us know what SCSI controller you're using in your peripheral, and whether you've had any particularly good or bad experiences with it. We'll add to this note as more information becomes available.

Back to top

References

The SCSI Manager

SCSI Developer's Package

Back to top

Downloadables

 Acrobat version of this Note (K) Download

Back to top
