To: X3T9.2 Committee X3T9.2/92-245r0

From: Randell Jesup, Commodore Business Machines (Engineering Dept)
jesup@cbmvax.cbm.commodore.com

Subject: Comments for the CAM group of the SCSI X3T9.2 committee

Date: Dec 21, 1992

This has been sent by mail to Lynn Barra, X3 Secretariat, and John Lohmeyer by US mail today. I'll also send John an electronic copy if this won't do; if so, John, please send me your email address.

Again, thanks to all who helped answer and refine my questions about the spec.

This is a formal set of comments and questions on the CAM draft, which is currently in a public review period. I understand that formal responses are unlikely to be available for several months, so I would be appreciative of any unofficial answers or comments on any or all of these questions, either from members of the committee or any other interested party. I and several others are currently working on CAM modules to the draft spec, and any guidance would be quite helpful.

Some of these comments/questions are actual problems in the spec, some may point out areas where notes to implementers would be advisable to avoid incompatible implementations.

These questions are based on the draft CAM spec, Rev 3.0, dated April 27, 1992.

1. There is no definition of XPT_NOP.

2. How "timely" must the SIM timeout mechanism be? Is +.99.. seconds allowed? Are +1.x seconds, or +10 seconds, or any arbitrary (but finite) time after the specified time allowed? If they aren't allowed, that may be a problem, as most OS timer mechanisms don't guarantee any specific time other than longer than you asked for (because of task priorities, etc). Also, on some systems there may be a requirement to function before the system is totally initialized, and at that time timing facilities/interrupts may not be available (I'm told at least on UN*X variant has this problem).

3. When aborting a timed-out request, how can this be safely done in all instances? Exactly what action should the SIM take? Some cases are easy (IO not started yet), some may be hard or take a while (data transfer in progress that can't be interrupted), and some may be impossible, perhaps (what if the target is disconnected - must/can you hold the CCB until it reconnects so you can send an Abort? What if it never reconnects?

4. In 6.6, paragraph 5, the spec states that a driver can change it's events mask by another call using the same callback value. What about the pdrv_buf and buffer length? I've been told that those are supposed to be updated as well. This needs to be reflected in the spec.

5. In a number of places, the CAM spec says "shall return a non-zero CAM Status". Shouldn't that say that is "shall return a CAM Status of other than Request in Progress", or "shall return a non-zero CAM Status (status other than Request in Progress)"? Using "zero" requires that the reader already have read the CAM status table, which occurs later in the chapter, to understand the meaning. I understand that this may be being revised, so by the time this is answered, the question may be moot.

6. In 6.3.2, it states that Callback on Completion is for Execute SCSI CCB's. I think it also applies to CCB's passed to the SIM via Enable Lun.

7. A number of XPT/SIM commands list CAM statuses to be returned. Are other statuses allowed? I assume so, since none of them mention Invalid Path ID.

8. There is a CAM status of LUN Already Enabled. Does this mean that an Enable LUN for an already-enabled LUN _must_ be returned with that error? Note that 10.1 (Enable Lun) mentions a number of errors, but they do not include that one. Also, nowhere in 10.1, 10.2, or 10.3 does it say that an Enable LUN of an already-enabled LUN is invalid. For example, it could be used to add more CCB's to the list of CCB's for the LUN. If it is to be disallowed, it should be so stated. If it is allowed, does it replace the previous set of CCB's, or is it added to them? (Replace seems more correct to me.)

9. 10.1 (Enable LUN) mentions a CAM status of Invalid Request as being returned if "there is currently a nexus established with an initiator that shall be terminated first". I assume that would only happen on an Enable LUN for an already-enabled LUN. In that case, what should happen to the other CCB's already queued for the LUN?

10. It's very unclear in the spec as to whether processor mode target operation supports only the 4 commands mentioned in 10.3, or whether those are an example, or whether the SIM is required to support those but not all others.

11. Typo: 10.3.2, 7th paragraph, starts like this: "If an Inquiry CDB is and there..."

12. If the only way to change or add CCB's to a enabled LUN is to Enable LUN with number of CCB's = 0, and then Enable LUN again to add the new set, then doesn't this open a hole where the LUN could be selected and it will return that the lun is invalid? If you are able to add/change CCB's to an Enabled LUN, this is not a problem (see question 8).

13. In a few places it is stated that 0xff is the path ID of the XPT itself. I'm told that any CCB's for path 0xff other than Path Inquiry should be rejected with Invalid Path ID. Is this correct? If so, should a note to this effect be included in the text?

14. Is there a reason that Set Device Type doesn't allow setting more than the first byte of the inquiry data? If a device exists, and Set Device Type is called for that path/id/lun, what data should be returned as inquiry data? What if it doesn't exist? In 7.x (OSD Operation), for DOS only the first byte must be retained. What about other OS's? Doesn't this limit of 1 byte of data conflict with 8.2.1 (Get Device Type), which states that 36 bytes of inquiry data will be copied?

15. In Execute SCSI IO, how is residual length calculated? Is it requested - actual, or actual - requested? The text never makes it clear. (I understand that this is a known problem.)

16. CAM Flags in all commands are listed as OSD. However, they are fully defined in table 9.2, implying that they are not OS-dependant.

17. In Execute SCSI IO (9.1), VU Field and VU Flags are shown, and listed as being defined in the vendor specification. Which vendor is this? OS, XPT, SIM, driver? Perhaps it means OSD, not vendor? Or is it reserved to the SIM?

18. Can Message Buffer Pointer/Length in a SCSI IO request (table 9-1) be ignored in processor target mode?

19. I'm told that an XPT is required to pass all CCB's to the SIM. Is this correct (I think so), and if so where is it said? If it must pass them, are there any actions it must perform (such as setting status to CAM_REQ_CMP) before it is passed? Any after sim_action() returns? What

is the status if a request is handled by XPT successfully, but the SIM has an error in handling it? Must it be passed to the SIM if the XPT has some sort of failure in handling it (can't allocate memory, for example)? Must it pass all CCB's it gets, even if it's an unknown type and not a vendor-unique?

20. Exactly what is meant by "validating" processor-mode target CCB's at Enable LUN time? 10.3 states that any invalid request shall be returned and the LUN not enabled. Does this mean that valid requests _aren't_ returned, but the LUN is still not enabled? Must this validation occur before or after replying the Enable LUN?

21. Request Sense is listed as one of the commands to be supported in 10.3. Doesn't this cause a problem, in that sense info is supposed to be maintained on an I_T_x basis (at least)? Or perhaps this is meant to be the sense data returned for the lun for all initiators when not I_T_x sense data exists? If so, is the SIM required to maintain a separate I_T_x-based list of sense data, if any, for sense data generated inside the SIM, and use the CCB-list request-sense data if it has no sense data for that initiator?

22. There's an assumption in the processor-mode target description that CCB's that have had their callback called are to be automatically reused by setting CCB_Available to 1, without calling xpt_action (or anything else). Is my reading of this assumption correct?

23. If an Enable LUN with number of CCB's of 0 is sent, can the SIM release any stored I_T_x sense data?

24. In 10.3, shouldn't Test Unit Ready be added to the list of "normal" processor mode commands, since it's mandatory for all devices?

25. Should 10.3 mention that unless commands {x,y,z...} are supported, the device will not be a SCSI-2 compliant device? (Test Unit Ready, Request Sense, Inquiry, etc).

26. What the SIM can or should or must do on data overrun/underrun is unclear. Perhaps it must be unclear, or OSD, but this issue should be mentioned. Certainly some of the cases should NOT generate errors (underrun on DATA_IN). Some of them are arguable. For one I've been told that the only solution is a hard Reset(!) (overrun on DATA_OUT). I have serious problems with generating hard resets when other activity, including in disconnected devices may be occuring. My preferred solution would be to transfer garbage data and assert ATN, until I get a message out, and then send an Abort or Abort Tag message. Neither is good, I think this one is less bad. This should only come up (I hope) due to a driver bug, a target bug, or maybe a flakey bus (maybe).

27. If one is using target processor-mode to implement a processor device using Send and Receive (so you can communicate with an initiator that doesn't do target mode), how can you accept the Receive and disconnect it until you have data for the initiator that sent it?

28. There's no way to tag a Recieve command in processor mode as being for a specific initiator - it's first-come, first-served. Doesn't this pose a problem in using processor mode to talk to more than 1 initiator? Basically, the driver is giving data to a random initiator, making it very hard to implement a more-than-2 node VLAN (which is a 'goal' of SCSI-2 processor mode devices).

29. In processor mode, is there any way to find out which initiator executed the command that caused the CCB to be replied? I don't see anything about this in the spec (10.3)

30. When a target processor-mode CCB has been replied by the SIM, and before the driver sets CCB Available again, may the driver change the CDB for the CCB? I assume it can't be changed while it's marked as available, or can it?

31. 10.3 states that CCB's shall be examined at Enable LUN time for (among other things) "valid data pointers". A) What does this mean? B) what if the driver changes data pointers/lengths while the CCB is not available?

32. In processor mode, when a Receive command is executed, how should mismatches in the data size be handled? What about for Send?

I just want to add thanks to John Lohmeyer, John Gallant, Tim Smith, and Johnathan Vail (a lot of John's, sounds almost like Buckaroo Banzai...). Their answers and discussion on comp.periphs.scsi helped focus these questions far better than my original informal set, and helped bring out some important issues I wouldn't have noticed otherwise.

Randell Jesup