To: X3T9.2 Committee Membership

From: Edward A. Gardner, Digital Equipment Corporation

Subject: Minutes of SBP Working Group, March 1-2, 1993

Attendance:

Jeff Stai	Western Digital	
Ed Gardner	Digital Equipment	
Scott Smyers	Apple Computer	
Greg Floryance	IBM Rochester	
George Penokie	IBM Rochester	
Jim McGrath	Quantum	(March 1 only)
Nathan Hays	Quantum	
Jerry Marazas	IBM Boca Raton	

All present thanked Jerry Marazas of IBM for hosting the meeting, and complimented the luxurious facilities. At Jerry's request we noted his IBM corporate attire before he removed his jacket.

Agenda:

- Make log in optional (Jerry)
- Make Initiator ID optional (Jerry)
- Have sign in be optional (Jerry)
- Make first command block act as tap for the chain (Jerry, Scott, Nate)
- Command block alignment (Ed)
- Dealing with SBP target resets (Ed)
- Squirrel (Scott, Nate)
- Fetch policy tutorial (Greg)
- Address boundary requirements (Jeff)

This was the agenda proposed at the start of the meeting, the actual agenda as followed is reflected in the following section headings.

1. Fetch Policy Tutorial (March 1, Greg)

Greg handed out a diagram illustrating an implementation of the fetch policy discussed (and nominally agreed to) at the Austin meeting. Discussion led to the following points.

How urgent should an Urgent Tap be? There was agreement that “Following an Urgent Tap, the target shall fetch as most one SBP command block before acting on the Urgent Tap.” This would allow a target to complete a fetch it had already begun (even if its decision to do so were not yet externally visible) while still providing prompt response to Urgent Taps. [Note: I believe people were not considering ACA or Isochronous taps in this discussion, instead focusing on the relationship of Urgent and Normal taps.]

Linked commands. Wide ranging discussion (and confusion) of the relationship between linked command execution and linked command chain fetching. The one concise statement that emerged was that linked command chains are not pre-fetched. When a linked command completes, the linked command's chain is placed at the head of the tap slot fetch queue, so that the next command in the linked I/O process shall be the next command block fetched. This is necessary to avoid a potential deadlock. Some present found this so obvious they had trouble expressing it clearly, while others thought this necessary behavior was prohibited by the spec. The specification shall be clarified.

Consensus was reached on the following [again, ignoring ACA conditions and isochronous operations]:

> The target maintains a tap slot fetch queue, an ordered list of taps waiting to be fetched. The target fetches commands from the tap at the head of the tap slot fetch queue. That is, the target reads the next command block from the chain at the head of the tap slot fetch queue and enters that command into the designated task set. That chain remains at the head of the tap slot fetch queue unless the SCE flag is set, the L flag is set, or an Urgent Tap is received.
>
> If the SCE flag is set, the chain is moved to the tail of the tap slot fetch queue.
>
> If an Urgent Tap is received, the Urgent Tap becomes the new head of the tap slot fetch queue, and the previous head is pushed down to become the second element in the queue. If a fetch is in progress when the Urgent Tap is received, the fetch may be completed before the Urgent Tap is processed.
>
> If the L flag is set, the tap slot (chain) is placed in a holding area until the command completes execution. After the command completes execution, the held tap slot is placed at the head of the tap slot fetch queue and the fetch of its next command block begun. The next command block of the linked sub chain shall be the first command block fetched after the linked command's resources (e.g., internal command storage area) are released.

2. Command Block Alignment (March 1, Ed)

Ed Gardner distributed a document describing how aligning the CDB results in internal fields of the CDB being misaligned. Discussion ranged over living with the existing alignment, changing the CDB field's alignment within command block, and maintaining the CDB field's alignment byte but shifting the CDB within it by two bytes.

Several straw votes were taken:

1. The CDB field shall be both 16 bytes long and aligned on a 16-byte boundary within the command block. 6-1-0
2. The Operation Code byte shall be at a fixed location within the CDB field. 8-0-0
3. The Operation Code byte shall be at byte 16 of the command block (4); at byte 18 (2); abstain (2).
4. We should seek to minimize impact on existing device driver software, existing target firmware, and existing hardware. 4-0-4

Ed Gardner will update his document to reflect these results for presentation to the general working group.

3. Make first command block act as tap for the chain (March 1, Jim, Nate)

Jim and Nate presented the argument that separate tap packets are an unnecessary complication. Sending the first command block instead of a tap message reduces the number of distinct data structures that must be dealt with, therefore simplifying the protocol and its implementation, improves performance, is actually easier for targets to implement, and overall is a generally good thing. After considerable discussion there was concensual agreement with the overall concept, but details had not yet been resolved. [Many other people have been informally suggesting the same thing, but Jim McGrath managed to speak up first. Later topic “Reconsideration of Tap Messages” resolves the details of this concept.]

4. Scsi Queues Using Indirect Reference in Real-time Editable Lists (March 1, Nate)

Nathan Hays has devised a two dimensional queuing scheme, variously called SQRL or SQUIRREL, that allows targets to store the tap slot fetch list in initiator memory. As with CCU, this would allow arbitrarily large tap lists. Unlike CCU, Nathan’s SQRL scheme does not use interlocked memory accesses. However it would require increasing the size of the command block, which it might or might not be possible to compensate by removing other fields.

Nathan’s presentation led to discussion of the consequences of overflowing target memory resources. Some present viewed as desirable anything that eliminates BUSY, Queue Full status, or their SBP equivalents. Others disagreed, claiming this proposal treated merely the symptoms but not the underlying problem. No concensus was reached on the inherent merits of the proposal. However, while no formal vote was taken, most present observed that taps and chains have been accepted and reasonably stable for many months, and expressed concern that considering SQRL or SQUIRREL would re-open a messy can of worms, regardless of its technical merits. There appeared to be concensus (at least from everyone except Nate) that SQRL should not be persued.

5. Reconsideration of Tap Messages (March 2, Jerry)

Jerry made three proposals:

1. Same as now: 12-byte tap, 64-byte command block, sent as separate packets.
2. Concatenate tap packet to first command block. First command block in a chain would be longer than 64 bytes, subsequent command blocks in a chain would remain at 64 bytes. Tap message would be first (longer) command block (perhaps 72 bytes), target would fetch subsequent 64-byte command blocks.
3. Keep all command blocks at 64 bytes, but use a different, shorter value for the correlation ID (queue tag value). The tap is the first 64 byte command block, target would fetch subsequent 64-byte command blocks.

Ed suggested:

4. Separate command block into 64-byte portion used for command execution and an extension used to construct command chains. For example, a 72 byte command block where the Next Command Address field is in bytes 65 through 71. The first 64 bytes are sent as the tap message. The remaining 8 bytes are fetch by the target, but only if the chain contains additional commands.

After discussion, Jeff and the room at large proposed:

5. Command blocks are 72 bytes long, tap messages send the first 64 bytes of the first command block. The Next Command Address field is in bytes 0-7 of the command block, the This Command Address (a self pointer) is in bytes 8-15 of the command block. The Sense Buffer Address is at the end of the command block, that is, bytes 64-71. Typically a target will initially fetch just the first 64 bytes, and only fetch the last eight bytes (Sense Buffer Address) if sense data needs to be returned.

Everyone present (Jerry, Jeff, Ed, Scott, Greg, George, Nate) liked and supported this proposal, albeit with some quibbling over wording.

6. Work that Needs to be Done List (March 2, Jerry)

A laundry list of various details, to explicitly record there is concensus on:

1. The log in procedure guarantees that an initiator gets an 8-bit ID for which there are no outstanding commands. (Note there may be some relation to Ralph Weber's concerns).
2. Log in is done with the FIFO address corresponding to initiator ID zero; this is the only FIFO address that may be used for log in. Note that zero is not a valid initiator ID value. Log in is mandatory.
3. There is consensus we need to do something about resource or tap slot allocation, reporting, queue full avoidance, or something in this general area. There is little understanding or consensus of what should be done. Jerry proposed a minimalist approach of the target reporting (perhaps by a configuration ROM entry) the number of tap slots it supported. There were no detectable signs of consensus on this.
4. Sign in is optional for initiators (an initiator can choose whether or not to request asynchronous events). Sign in is mandatory for targets.

7. Tap Slot Allocation Model (March 2, Ed)

After lunch Ed presented the beginnings of a model for allocating and managing tap slots in SBP, and perhaps command slots in other SCSI-3 protocols. While discussion was somewhat curtailed by people leaving for the airport, there was agreement that this was worth persuing further, particularly in the general working group.