AHCI 1.0

Erratum 006

AHCI 1_0 Erratum_006.doc

Please send comments to Amber Huffman
amber.huffman@intel.com

Table of Contents

1 Aggressive Power Management Clarifications

1.1 Description of Technical Issue

The specification is not clear on software requirements regarding aggressive power management when the CAP.SALP bit is cleared to '0'. The errata clarifies that software shall only utilize the aggressive power management register bits if CAP.SALP is set to '1'.

1.2 Description of Correction to Specification

Modify the definition of the "Supports Aggressive Link Power Management" bit in the CAP register in section 3.1.1 as follows:

26	RO	Impl. Spec	**Supports Aggressive Link Power Management (SALP):** When set to '1', indicates that the HBA can support auto-generating link requests to the Partial or Slumber states when there are no commands to process. When cleared to '0', software shall treat the PxCMD.ALPE and PxCMD.ASP bits as reserved. Refer to section 8.3.1.3.

Modify the definition of the "Aggressive Link Power Management Enable" bit in the PxCMD register in section 3.3.7 as follows:

26	RW/ RO	0	**Aggressive Link Power Management Enable (ALPE):** When set to '1', the HBA shall aggressively enter a lower link power state (Partial or Slumber) based upon the setting of the ASP bit. Software shall only set this bit to '1' if CAP.SALP is set to '1'; if CAP.SALP is cleared to '0' software shall treat this bit as reserved. See section 8.3.1.3 for details.

Modify the definition of the "Aggressive Slumber / Partial" bit in the PxCMD register in section 3.3.7 as follows:

27	RW/ RO	0	**Aggressive Slumber / Partial (ASP):** When set, and ALPE is set, the HBA shall aggressively enter the Slumber state when it clears the PxCI register and the PxSACT register is cleared or when it clears the PxSACT register and PxCI is cleared. When cleared, and ALPE is set, the HBA shall aggressively enter the Partial state when it clears the PxCI register and the PxSACT register is cleared or when it clears the PxSACT register and PxCI is cleared. If CAP.SALP is cleared to '0' software shall treat this bit as reserved. See section 8.3.1.3 for details.

2 PxSERR.ERR.E bit definition is incorrect

2.1 Description of Technical Issue

The PxSERR.ERR.E bit definition in AHCI is much narrower in scope than the Serial ATA 1.0a definition of this bit. This erratum broadens the definition to cover all internal errors as in the Serial ATA 1.0a definition.

2.2 Description of Correction to Specification

Modify the definition of the PxSERR.ERR.E bit in section 3.3.12 as follows:

11	**Internal Error (E):** ~~The SATA controller failed due to a master or target abort when attempting to access system memory.~~ The host bus adapter experienced an internal error that caused the operation to fail and may have put the host bus adapter into an error state. The internal error may include a master or target abort when attempting to access system memory, an elasticity buffer overflow, a primitive mis-alignment, a synchronization FIFO overflow, and other internal error conditions. Typically when an internal error occurs, a non-fatal or fatal status bit in the PxIS register will also be set to give software guidance on the recovery mechanism required.

3 MSI corrections

3.1 Description of Technical Issue

The message signaled interrupt (MSI) sections require several clarifications. The PCI specification (where MSI is defined) does not require the device to handle software incorrectly setting the Multiple Message Enable field. This errata updates AHCI to allow incorrect software settings of this field to result in indeterminate behavior. This errata also add clarifications for when single MSI versus multiple MSI is in use.

3.2 Description of Correction to Specification

Modify the "Multiple Message Enable" field in section 2.3.2 as follows:

06:04	RW	000	**Multiple Message Enable (MME):** Indicates the number of messages the HBA should assert. See section 0. If the value programmed into this field exceeds the MMC field in this register, ~~only a single message shall be generated~~ the results are indeterminate.

Modify the first paragraph in section 10.6.2.1 as follows:

10.6.2.1 Pin Based and Single MSI Message Based Behavior

This is the mode of interrupt operation if any of the following conditions are met:

- Pin based interrupts are being used – MSI is disabled (MSICAP.MC.MSIE='0')
- Single MSI is being used – MSI is enabled (MSICAP.MC.MSIE='1') and MSICAP.MC.MME=0h
- ~~MSI is disabled via MSICAP.MC.MSIE~~
- ~~HBA only supports a single MSI interrupt via the configuration register MSICAP.MC.MMC~~
- ~~HBA only enabled for a single interrupt via MSICAP.MC.MME~~
- ~~MSICAP.MC.MME is programmed to a larger value than MSICAP.MC.MMC~~

Modify the first paragraph in section 10.6.2.2 as follows:

10.6.2.2 Multiple MSI Based Messages

An HBA may optionally support multiple MSI messages for better performance. In this mode, multiple interrupt messages are allocated for the controller; each port may have ~~has~~ its own interrupt message. To support this mode, the MSICAP.MC.MMC field represents a power-of-2 wrapper on the number of implemented ports in the global memory space PI register. For example, if 3 ports are implemented, then the MSICAP.MC.MMC field must be '010' (4 interrupts).

Modify the paragraph that follows Table 3 in section 10.6.2.2 as follows:

When generating an MSI message, a port looks at its PxIS register, and uses the following rules to generate a message:

- If a new bit is set in PxIS, and the corresponding bit in PxIE is set, send a message
- If bits are cleared in PxIS, and other bits remain set, if their corresponding bits in PxIE are set, send a message.
- If a PxIE bit transitions from '0' to '1' and the corresponding bit in PxIS is set, send a message.

4 Command Completion Processing Clarifications

4.1 Description of Technical Issue

In order to determine what commands have completed, software must compare the PxCI or PxSACT register values to a stored list of outstanding commands that it has previously issued. The current language in section 5.4.3 incorrectly says that software should compare a previous value of PxCI/PxSACT against the current value to determine those commands that have been completed.

4.2 Description of Correction to Specification

Modify section 5.4.3 as shown:

5.4.3 Processing Completed Commands

Software processes the interrupt generated by the device for command completion. In the interrupt service routine, software checks IS.IPS to determine which ports have an interrupt pending.

For each port that has an interrupt pending:

1. Software determines the cause of the interrupt by reading the PxIS register. It is possible for multiple bits to be set
2. Software clears appropriate bits in the PxIS register corresponding to the cause of the interrupt.
3. Software clears the interrupt bit in IS.IPS corresponding to the port.
4. If executing non-queued commands, software reads the PxCI register, and compares the current value to the list of commands previously issued by software that are still outstanding ~~a previously read value~~. If executing native queued commands, software reads the PxSACT register and compares the current value to the list of commands previously issued by software ~~a previously read value~~. Software~~It~~ completes with success any outstanding command~~s~~ whose corresponding bit has been cleared in the respective register ~~since the last value was read~~. PxCI and PxSACT are volatile registers; software should only use their values to determine commands that have completed, not to determine which commands have previously been issued.
5. If there were errors, noted either in the PxIS register or PxTFD.STS.ERR, software performs error recovery actions (see section 6.2.2).

5 Updates to Command Header structure

5.1 Description of Technical Issue

There are four bytes of reserved space that follow each command header. This reserved space is often overlooked leading to confusion. This errata includes the reserved space as part of the command header definition itself to avoid this issue.

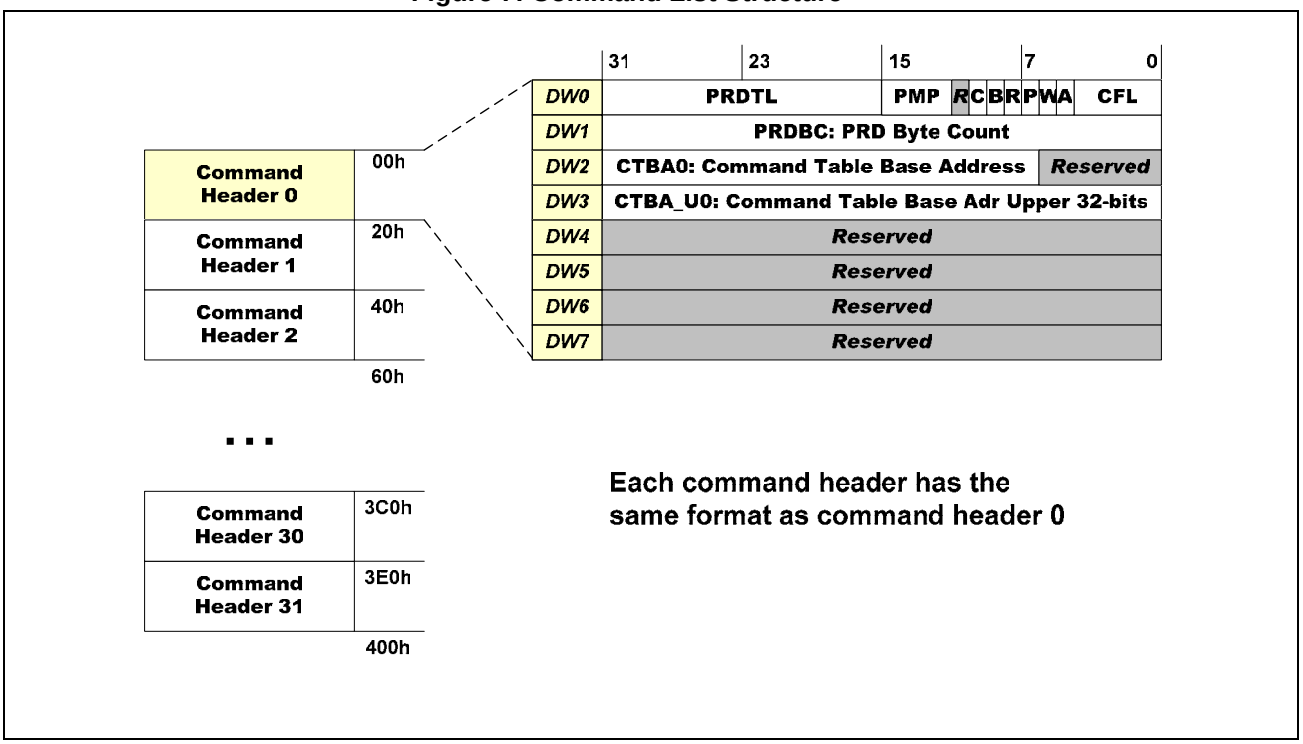
5.2 Description of Correction to Specification

Replace Figure 7 with the figure shown and modify the text in section 4.2.2 as shown:

4.2.2 Command List Structure

Figure 7 shows the command list structure. Each entry contains a command header, which is a 32~~16~~-byte structure that details the direction, type, and scatter/gather pointer of the command. Further details of each field are listed below.

Figure 7: Command List Structure



Include a new figure following Figure 11 to describe Dwords 4-7 in the Command Header:

Figure 11a: DW 4-7 – Reserved

Dword	Description
4	*Reserved*
5	*Reserved*
6	*Reserved*
7	*Reserved*

6 Hot Plug Capable Port definition

6.1 Description of Technical Issue

The definition of hot plug capable port is not as precise as it could be and leads to misinterpretations. With external SATA starting to become available, the confusion around this bit becomes greater. This errata clarifies that this bit is only used for traditional hot plug that includes the power connector as part of the hot plug. A future revision of AHCI is planned to capture showing external SATA capability (where the signal cable only is included in the hot plug operation).

6.2 Description of Correction to Specification

Modify the "Hot Plug Capable Port" field in section 3.3.7 as follows:

18	RO	HwInit	**Hot Plug Capable Port (HPCP):** ~~If set to '1', the platform has exposed this port to the user for hot plug insertion/removal. If cleared to '0', the platform has not exposed this port to the user for hot plug insertion/removal. If P0CPD or P0ISP is set to '1', then this bit should be set to '1'. If the device connected to this port is screwed into the system chassis such that it is not hot pluggable, this bit should be cleared to '0'.~~ When set to '1', indicates that this port's signal and power connectors are externally accessible via a joint signal and power connector for blindmate device hot plug. When cleared to '0', indicates that this port's signal and power connectors are not externally accessible via a joint signal and power connector.