Seagate

Seagate Technology
10323 West Reno (West Dock)
Oklahoma City, OK 73127-9705
P.O. Box 12313
Oklahoma City, OK 73157-2313

Tel: 405-324-3070
Fax: 405-324-3794
gene_milligan@notes.seagate.com

Date: 8/10/97

To: X3T10 Membership

From: Gene Milligab, T10 Principal member

Subject: **Proposal for Contingent Allegiance / Auto Contingent Allegiance Handling**

Overview:

Although the SCSI requirement for one queue per LUN shared among all initiators with a requirement that the queue be frozen until a contingent allegiance is cleared offers a robust environment for data integrity, there are application environments that are demanding a different approach. These application environments are strongly requesting that SCSI devices handle errors in such a way that host systems can be unaware of any errors that are encountered by other host systems. Achievement of this goal places requirements on the device's behavior such as maintaining separate sense data for each host, not freezing the queue for all hosts when one host is in contingent allegiance, and not sending BUSY status to a host because another host has encountered some type of error condition. This need was discussed at the July T10 working group meeting and action was assigned to generate this proposal.

A key aspect of this proposal is to provide a mechanism that will allow those applications that prefer an environment as currently required by SCSI to continue to be supported by the default settings while adding the tools to allow the other environment in a standard manner. (The proposal is crafted such that those applications need do nothing beyond the present SCSI-3 definition.) Lack of such a standard mechanism will result in the proliferation of non-standard approaches. This proposal assumes applications that want no impact in command execution from other initiators will not invoke ordering. This proposal does not include a method to divide ordering on a per initiator basis.

Over view of the two alternative error handling methods:

Strict single queue model:

This method is the one which is currently required by SCSI. The new mode bit Sustained Execution bit is zero and may or may not be changeable at the vendors choice. Whether a Contingent Allegiance (CA) or Auto Contingent Allegiance (ACA) is used is dependent upon the vendor's choice as indicated by the NormACA bit in the standard INQUIRY data and the NACA bit in the Control Byte of the CDB. For simplicity in the overview CA will be used for the description.

- When an error condition is encountered, the command is terminated with a CHECK CONDITION status and the target enters a CA condition with the initiator that sent the command.
- While the CA condition exists, the command queue is frozen. None of the commands that are in the queue when the CA condition is entered will be enabled or re-enabled until the CA condition is cleared.
- To clear the CA condition, the target must receive an untagged command (NACA 0) or a CLEAR ACA task management function (NACA 1) from the initiator for which the CA condition exists.
- After the clearing action is complete, the queue is unfrozen and normal command execution is resumed (unless the queue was aborted with a QErr field of 01b or 11b [for the faulting initiator]).
- If the command cannot be used to clear the CA condition, the device will return BUSY or ACA ACTIVE status (depending upon NACA) to the initiator.
- BUSY or ACA ACTIVE status (depending upon NACA) will be returned for any command from any initiator that is not in contingent allegiance and for any tagged command from the initiator that is in contingent allegiance.

Proposed alternate queue model:

- When an error is encountered, the command is terminated with a CHECK CONDITION status and the device enters a CA condition with the initiator that sent the command.
- Once the CA condition has been entered, the device will stop enabling commands from the faulting initiator. The device will continue to enable and execute commands from the other initiators to the ready queue provided that the rules for ordering commands are met.
- If a command, from an initiator that is not in contingent allegiance, that was received after an ordered command from the faulting initiator is next to be enabled, the queue will be frozen, and the device will not enable any more commands until the CA condition is cleared.
- If a command is received from an initiator that is not in contingent allegiance, the command will be executed or queued depending upon the ordering rules and the state of the queue.
- If a tagged command is received from a host that is in contingent allegiance, the device will return BUSY or ACA ACTIVE status (dependent upon NACA).
- If an untagged command (NACA 0) or a CLEAR ACA task management function (NACA 1) is received from an initiator that is in contingent allegiance its placement in the queue is vendor specific but it is allowed to be treated as a head of queue.
- The CA condition will be cleared for the initiator that sent the untagged command when that command (NACA 0) or a CLEAR ACA task management function (NACA 1) completes.
- Because the device will continue to execute commands for initiators that are not in contingent allegiance, these commands have a low but finite probability of also encountering errors which will result in more than one initiator being in contingent allegiance at a time.

CA_ACA Proposal:

The following proposal uses snippets of the clauses to keep sufficient context. Changes are denoted by: addition; <<deletion>>; and **((proposer s comment which is not part of the proposal nor part of the draft standard)).**

Changes to SPC-2:

3.1.5 auto contingent allegiance: <<The>> One of two alternative conditions of a task set following the return of a CHECK CONDITION or COMMAND TERMINATED status. A detailed definition of ACA may be found in SAM.

3.1.TBD contingent allegiance:<<The>> One of two alternative conditions of a task set following the return of a CHECK CONDITION or COMMAND TERMINATED status. A detailed definition of CA may be found in SCSI-2 and SAM.

3.1.TBD task set: A group of tasks within a logical unit, whose interaction is dependent on the task management (queuing), CA, and ACA rules defined in SAM and as augmented by the Control Mode Page (See 8.3.4).

CA	Contingent Allegiance (see 3.1.TBD and SCSI-2)

5.4 Multiple port and multiple initiator behavior

Once a device server grants a reservation, all initiators (regardless of port) except the initiator to which the reservation was granted shall be treated as different initiators. If SEXE = 1 only the following operations allow an initiator to interact with the tasks of another initiator, regardless of the service delivery port:

a) the PERSISTENT RESERVE OUT with Preempt service action removes persistent reservations for another initiator (see 7.13.1.5);
b) the PERSISTENT RESERVE OUT with Preempt and Clear service action removes persistent reservations and all tasks for another initiator (see 7.13.1.6);
c) the PERSISTENT RESERVE OUT with Clear service action removes persistent reservations and reservation keys for all initiators (see 7.13.1.4);
d) the TARGET RESET task management function removes reservations established by the Reserve/Release method and removes all tasks for all logical units in the target and for all initiators (see SAM). Persistent reservations remain unmodified;
e) the LOGICAL UNIT RESET task management function removes reservations established by the Reserve/Release method and removes all tasks for all initiators for the addressed logical unit and any logical units depending from it in a hierarchical addressing structure (see SAM). Persistent reservations remain unmodified; and
f) the CLEAR TASK SET task management function removes all tasks for the selected logical unit for all initiators. Most other machine states remain unmodified, including MODE SELECT parameters, reservations, and ACA (see SAM).

If SEXE = 0 in addition to the above list operations from an initiator which result in a CA or ACA will block the tasks of another initiator, regardless of the service delivery port.

((I don t think regardless of the service delivery port should qualify an initiator but that fact is independent of this proposal.))

7.13.1.6 Preempt and Clear

The Preempt and Clear service action shall clear any CA or ACA condition associated with the initiator being preempted and shall clear any tasks with an ACA attribute **((the plural in this requirement was in violation of SAM prior to this proposal))** from that initiator. If SEXE = 0 CA or ACA conditions for other initiators shall prevent the execution of the PERSISTENT RESERVE OUT task, which shall end with status of BUSY (NACA = 0) or ACA ACTIVE (NACA = 1).

NOTE 28 The Preempt and Clear service action will clear the ACA condition associated with the initiator being preempted even_though the task is terminated with an ACA ACTIVE status. Thus, the next command arriving at the device server will not encounter the ACA condition previously active for the initiator being preempted.

7.20 REQUEST SENSE command

If the device server is in the Standby power condition or Idle power condition when a REQUEST SENSE command is received and there is no CA or ACA condition, the device server shall return a sense key of NO SENSE and an additional sense code of LOW POWER CONDITION ON. On completion of the command the logical unit shall return to the same power condition that was active before the REQUEST SENSE command was received. A REQUEST SENSE command shall not reset any active power condition timers.

7.20.3 Deferred errors

“If the task terminates with CHECK CONDITION status and the subsequent sense data returns a deferred error that task shall not have been executed. After the device server detects a deferred error condition, it shall return a deferred error according to the rules described below:

a) If no external system intervention is necessary to recover a deferred error, a deferred error indication shall not be posted unless required by the error handling parameters of a MODE SELECT command. The occurrence of the error may be logged if statistical or error logging is supported.
b) If it is possible to associate a deferred error with a causing initiator and with a particular function or a particular subset of data, and the error is either unrecovered or required to be reported by the mode parameters, a deferred error indication shall be returned to an application client on the causing initiator. If an application client on an initiator other than the causing initiator attempts access to the particular function or subset of data associated with the deferred error, <<a BUSY status shall be returned to that application client in response to>> the command attempting the access shall be responded to according to the requirements for a contingent allegiance condition.
c) If a deferred error cannot be associated with a causing initiator or with a particular subset of data, the device server shall return a deferred error indication to an application client on each initiator. If multiple deferred errors have accumulated for some initiators, only the last error shall be returned.
d) If a deferred error cannot be associated with a particular logical unit, the device server shall return a deferred error indication to an application client associated with any logical unit on the appropriate initiator.

If a task has never entered the enabled task state, and a deferred error occurs, the task shall be terminated with CHECK CONDITION status and deferred error information posted in the sense data. If a deferred error occurs after a task has entered the enabled task state and the task is affected by the error, the task shall be terminated by CHECK CONDITION status and the current error information shall be returned in the sense data. In this case, if the current error information does not adequately define the deferred error condition, a deferred error may be returned after the current error information has been recovered. If a deferred error occurs after a task has entered the enabled task state and the task completes successfully, the device server may choose to return the deferred error information after the completion of the current command in conjunction with a subsequent command that has not started execution.”

8.3.4 Control mode page

The control mode page (see table 98) provides controls over several SCSI-3 features that are applicable to all device types such as tagged queuing, asynchronous event reporting, and error logging.

Table 98 _ Control mode page

Byte \ Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	Page code (0Ah)					
1	Page length (0Ah)							
2	Reserved						GLTSD	RLEC
3	Queue algorithm modifier				SEXE	QErr		DQue
4	Reserved	RAC	Reserved		SWP	RAERP	UAAERP	EAERP
5	Reserved							
6	(MSB)	Ready AER holdoff period						
7								(LSB)
8	(MSB)	Busy timeout period						
9								(LSB)
10	Reserved							
11	Reserved							

A sustained execution (SEXE) bit of zero specifies that tasks in the task set shall be blocked when a CA or an ACA condition is established. A SEXE bit of one and a QErr field of 00b or 11b specifies that tasks in the task set for initiators other than the faulting initiator shall not be blocked by an ACA condition. If the SEXE bit equals one and the QErr field is equal to 01b, the Mode Select command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. The state of the SEXE bit shall not change any of the task ordering requirements.

((Proposer s Note: Rather than adding the SEXE bit another alternative would be to use the reserved 10b value for the QErr field. It appears that the impact on designs which had not yet implemented the two bit QErr field would be the same. But for those, if any, that have already implemented the two bit value the added SEXE may be better.))

The queue error management (QErr) field specifies how the device server shall handle <<blocked>> tasks when another task receives a COMMAND TERMINATED or CHECK CONDITION status (see table n1).

Table n1 - Queue error management (QErr)

Value	Definition
00b	Blocked tasks in the task set shall resume after CA or an ACA condition is cleared (see SAM).
01b	All the blocked tasks in the task set shall be aborted when the COMMAND TERMINATED or CHECK CONDITION status is sent. A unit attention condition (see SAM) shall be generated for each initiator that had blocked tasks aborted except for the initiator to which the COMMAND TERMINATED or CHECK CONDITION status was sent.

	The device server shall set the additional sense code to COMMANDS CLEARED BY ANOTHER INITIATOR.
10b	Reserved
11b	Blocked tasks in the task set belonging to the initiator to which a COMMAND TERMINATED or CHECK CONDITION status is sent shall be aborted when the status is sent.

A disable queuing (DQue) bit of zero specifies that tagged queuing shall be enabled if the device server supports tagged queuing. A DQue bit of one specifies that tagged queuing shall be disabled. Any queued commands received by the device server shall be aborted. The method used to abort queued commands is protocol-specific.

Changes to SAM-2:

5.6.1.1 Logical Unit Response to Auto Contingent Allegiance

The contingent allegiance (NACA = 0) or auto contingent allegiance (NACA = 1) condition shall not cross task set boundaries and shall be preserved until it is cleared as described in 5.6.1.2. If requested by the application client and supported by the protocol and logical unit, sense data shall be returned as described in 5.6.4.2.

Notes:
1. The SCSI-2 contingent allegiance condition and extended auto contingent allegiance condition have been <<replaced>> augmented in SCSI-3 by auto contingent allegiance in conjunction with the NACA bit.
2. If the SCSI-3 protocol does not enforce state synchronization as described in 4.6.1, there may be a time delay between the occurrence of the contingent or auto contingent allegiance condition and the point at which the initiator becomes aware of the condition.

After sending status and a service response of TASK COMPLETE, the logical unit shall modify the state of all tasks in the faulted task set as described in clause 7. A task created by the faulted initiator while the auto contingent allegiance condition is in effect may be entered into the faulted task set under the conditions described below. **((Order rearranged))**

If the SEXE bit (See SPC-2) equals zero: As described in 5.6.1.2, the setting of the NACA bit in the control byte of the faulting command determines the rules that apply to an ACA condition caused by that command. If the NACA bit was set to zero the SCSI-2 contingent allegiance rules shall apply. In that case, the completion of a subsequent command from the faulted initiator with a status of CHECK CONDITION or COMMAND TERMINATED shall clear the existing contingent allegiance and cause a new auto contingent allegiance condition to exist. The rules for responding to the new auto contingent allegiance condition shall be determined by the state of the NACA bit in the new faulted command. If the NACA bit was set to one in the CDB control byte of the faulting command, then a new task created while the ACA condition is in effect shall not be entered into the faulted task set provided unless all of the following conditions are true:
a) The command was originated by the faulted initiator,
b) The task has the ACA attribute,
c) No other task having the ACA attribute is in the task set.

If any of the conditions listed above are not met, the newly created task shall not be entered into the task set and shall be completed with a status of ACA ACTIVE. If a task having the ACA attribute is received and no auto contingent allegiance condition is in effect for the task set or if the NACA bit was set to zero in the CDB for the faulting command, then the ACA task shall be completed with a status of CHECK

CONDITION. The sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID MESSAGE ERROR. As noted in 5.6.1.2, any existing contingent or auto contingent allegiance condition for that initiator shall be cleared and a new contingent or auto contingent allegiance condition shall be established.

Except for a PERSISTENT RESERVE command with a Preempt and Clear action as described in subclause 5.6.1.2, tasks created by other initiators while the ACA condition is in effect shall not be entered into the task set and shall be completed with a status of BUSY if NACA = 0 or ACA ACTIVE if NACA = 1. <<Tasks created by other initiators while the ACA condition is in effect shall not be entered into the faulted task set and shall be completed with a status of ACA ACTIVE.>>

If the SEXE bit (See SPC-2) equals one: As described in 5.6.1.2, the setting of the NACA bit in the control byte of the faulting command determines the rules that apply to an ACA condition caused by that command. If the NACA bit was set to zero the SCSI-2 contingent allegiance rules shall apply. In that case, the completion of a subsequent command from the faulted initiator with a status of CHECK CONDITION or COMMAND TERMINATED shall clear the existing contingent allegiance and cause a new auto contingent allegiance condition to exist. The rules for responding to the new auto contingent allegiance condition shall be determined by the state of the NACA bit in the new faulted command. If the NACA bit was set to one in the CDB control byte of the faulting command, then a new task created by the faulting initiator while the ACA condition is in effect shall not be entered into the faulted task set provided unless all of the following conditions are true:
<<a) The command was originated by the faulted initiator,>>
b) The task has the ACA attribute,
c) No other task from the faulting initiator having the ACA attribute is in the task set.

If the task is from the faulting initiator and any of the conditions listed above are not met, the newly created task shall not be entered into the task set and shall be completed with a status of ACA ACTIVE. If a task from the faulting initiator having the ACA attribute is received and no auto contingent allegiance condition is in effect for the <<task set>> faulting initiator or if the NACA bit was set to zero in the CDB for the faulting command, then the ACA task shall be completed with a status of CHECK CONDITION. The sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID MESSAGE ERROR. As noted in 5.6.1.2, any existing contingent or auto contingent allegiance condition for that initiator shall be cleared and a new contingent or auto contingent allegiance condition shall be established.

<<Except for a PERSISTENT RESERVE command with a Preempt and Clear action as described in subclause 5.6.1.2,>> Tasks in the task set for other initiators shall be enabled and completed in accordance with the requirements of Clause 7. Tasks created by other initiators while the ACA condition is in effect shall <<not>> be entered into the task set provided that the task set is not full and the command is a supported command without an illegal parameter.<<and shall be completed with a status of ACA ACTIVE>>.

5.6.4.2 Autosense

Autosense is the automatic return of sense data to the application client coincident with the completion of an SCSI-3 command under the conditions described below. The return of sense data in this way is equivalent to an explicit command from the application client requesting sense data immediately after being notified that an ACA condition has occurred. Inclusion of autosense support in an SCSI-3 protocol standard is optional.

As specified in clause 5, the application client may request autosense service for any SCSI command. If supported by the protocol and logical unit and requested by the application client, the device server shall only return sense data in this manner coincident with the completion of a command with a status of CHECK CONDITION or COMMAND TERMINATED. The sense data and the CA (NACA = 0) or ACA (NACA = 1) shall then be cleared.

5.6.5 Unit Attention Condition

A unit attention condition shall persist on the logical unit for each initiator until that initiator clears the condition as described in the following paragraphs.

If an INQUIRY command is received from an initiator to a logical unit with a pending unit attention condition (before the logical unit generates the contingent or auto contingent allegiance condition), the logical unit shall perform the INQUIRY command and shall not clear the unit attention condition.
If a request for sense data is received from an initiator with a pending unit attention condition (before the logical unit establishes the contingent or auto<<matic>> contingent allegiance condition), then the logical unit shall either:
a) report any pending sense data and preserve the unit attention condition on the logical unit, or,
b) report the unit attention condition.

If the second option is chosen (reporting the unit attention condition), the logical unit may discard any pending sense data and may clear the unit attention condition for that initiator.

If the logical unit has already generated the contingent or auto contingent allegiance condition for the unit attention condition, the logical unit shall perform the second action listed above. If NACA for the REQUEST SENSE command is zero and the command is untagged the contingent allegiance condition shall be cleared.

If an initiator issues a command other than INQUIRY or REQUEST SENSE while a unit attention condition exists for that initiator (prior to generating the contingent or auto contingent allegiance condition for the unit attention condition), the logical unit shall not perform the command and shall report <<CHECK CONDITION>> BUSY (NACA = 0) or ACA ACTIVE (NACA = 1) status unless a higher priority status **((??))** as defined by the logical unit is also pending (see 50)**((SIC))**.
((Because of the ?? this change is still under construction since for NACA = 0 the specific command is not the trigger but whether it is tagged or untagged. Therefore this clause should remain an issue regardless of the outcome of the basic proposal. It appears to me this subclause has become confused as to whether it is describing a Unit Attention condition or an AEN without an error.))

If a logical unit successfully sends an asynchronous event report informing the initiator of the unit attention condition, then the logical unit shall clear the unit attention condition for that initiator on the logical unit (see 5.6.4.1).

6 Task Management Functions

CLEAR ACA (Logical Unit Identifier) - Clear auto contingent allegiance condition. This function shall be supported if the logical unit accepts a<<n>> NACA bit value of one in the CDB control byte. <<and may be supported if the logical unit does not accept an NACA bit value of one in the CDB control byte>> (see 5.1.2).

All SCSI-3 protocol <<specifications>> standards shall provide the functionality needed for a task manager to implement all of the task management functions defined above.

6.3 CLEAR ACA
Function Call
Service response = CLEAR ACA (Logical Unit Identifier)

Description:
This function shall only be implemented by a logical unit that accepts an NACA bit value of one in the CDB control byte (see 5.1.2). **((Note the conflict prior to the above deletion.))**

The initiator invokes CLEAR ACA to clear an auto contingent allegiance condition from the task set serviced by the logical unit according to the rules specified in 5.6.1.2. If successful this <<The>> function shall <<always>> be terminated with a service response of FUNCTION COMPLETE.

If the task manager clears the auto contingent allegiance condition, any task within that task set may be completed subject to the rules for task set management specified in clause 7.

6.4 CLEAR TASK SET
Function Call:
Service response = CLEAR TASK SET (Logical Unit Identifier)

Description:
This function shall be supported by all logical units that support tagged tasks (see object definition 7) and may be supported by logical units that do not support tagged tasks.

The target shall perform an action equivalent to receiving a series of ABORT TASK requests from each initiator.

All tasks, from all initiators, in the specified task set shall be aborted. The medium may have been altered by partially executed commands. All pending status and data for that logical unit for all initiators shall be cleared.

No status shall be sent for any task. A unit attention condition shall be generated for all other initiators with tasks in that task set. When reporting the unit attention condition the additional sense code shall be set to COMMANDS CLEARED BY ANOTHER INITIATOR.

If the SEXE bit equals 0 (See SPC-2) previously established conditions, including MODE SELECT parameters <<(see the SPC standard)>>, reservations, and contingent or auto contingent allegiance shall not be changed by the CLEAR TASK SET function. If the SEXE bit equals 1 previously established conditions, including MODE SELECT parameters <<(see the SPC standard)>> and reservations shall not be changed by the CLEAR TASK SET function but the contingent or auto contingent allegiance condition shall be cleared if the CLEAR TASK SET function is from the faulting initiator and shall not be cleared if from any other initiator.

((Was it really intended that the CLEAR TASK SET be able to come through a contingent allegiance or is this in conflict with other sections?))

7.2 Task Management Events
The following is a description of the events that drive changes in task state.

All older tasks ended: All tasks have ended that were accepted into the task set earlier in time than the referenced task.

All older Head of Queue and older Ordered tasks ended: All Head of Queue and Ordered tasks have ended that were accepted into the task set earlier in time than the referenced task.

ACA : An auto contingent allegiance condition has occurred for a task with NACA = 1.

CA: A contingent allegiance condition has occurred for a task with NACA = 0.

task abort: One of the events described in subclause 7.3 has occurred.

task completion: The device server has returned sent a service response of TASK COMPLETE for the task (see clause 5 and subclause 5.4).

task ended: A task has completed or aborted.

ACA cleared: An ACA condition has been cleared.

CA cleared: A CA condition has been cleared.

Subclause 7.4 describes the events, changes in task state and device server actions for a Simple, Ordered, ACA or Head of Queue task.

7.3 Task Abort Events

A Task Abort event is one of the following:
a) Completion of an ABORT TASK task management function directed to the specified task;
b) Completion of an ABORT TASK SET task management function under the conditions specified in subclause 6.2;
c) Completion of a CLEAR TASK SET task management function referencing the task set containing the specified task;
d) Completion of a PERSISTENT RESERVE with a Preempt and Clear action directed to the specified task;
e) A CA or an ACA condition was cleared and the QErr bit was set to one in the control mode page (see the SPC standard);
f) An ACA condition was cleared and the task had the ACA attribute;
g) A hard reset (see 5.6.6);
h) The return of an Execute Command service response of SERVICE DELIVERY OR TARGET FAILURE as described in clause 5.
i) A power on condition.

7.4 Task States

7.4.2 Blocked

A task in the Blocked state is prevented from completing due to contingent or an auto contingent allegiance condition. A task in this state shall not become a current task. While a task is in the Blocked state, any information the logical unit has or accepts for the task shall be suspended. If the SEXE bit (See SPC-2) is zero the blocked state is independent of the initiator. If the SEXE bit equals one, except for the impact of ordering, the blocked state applies only to the faulting initiator.

7.5.4 ACA Task **((Alternative 1))**

A task having the ACA attribute shall be accepted into the task set in the Enabled state. As specified in 5.6.1.1, there <<may>> shall be no more than one ACA task per task set. **((It is not clear that this is specified in 5.6.1.1.))**

7.5.4 ACA Task **((Alternative 2))**
A task having the ACA attribute shall be accepted into the task set in the Enabled state. <<As specified in 5.6.1.1,>> If the SEXE bit equals zero (See SPC-2) there <<may>> shall be no more than one ACA task per task set. If the SEXE bit equals one there may be more than one ACA task per task set but there shall be no more than on ACA task per initiator.

7.6.1 Transition S0:S1 (Ordered Task): Provided a CA or an ACA condition does not exist or if SEXE = 1 (see SPC-2) provided the task is not for the faulting initiator, a dormant task having the ORDERED attribute shall enter the Enabled state when all older tasks have ended. If SEXE = 0 this transition shall not occur while a CA or an ACA condition is in effect for the task set.

7.6.2 Transition SO:S1 (Simple task): Provided a CA or an ACA condition does not exist or if SEXE = 1 provided the task is not for the faulting initiator, a dormant task having the SIMPLE attribute shall enter the Enabled state when all older Head of Queue and older Ordered tasks have ended. If SEXE = 0 this transition shall not occur while a CA or an ACA condition is in effect for the task set.

7.6.3 Transitions S0:S3, S2:S3: A task abort event shall cause the task to unconditionally enter the Ended state. **((Are there no CA or ACA exceptions?))**

7.6.4 Transition S1:S2: If SEXE = 0 a CA or an ACA condition shall cause an enabled task to enter the Blocked state. If SEXE = 1 a CA or an ACA condition shall cause an enabled task for the faulting initiator to enter the Blocked state.

7.6.5 Transition S1:S3: A task that has completed or aborted shall enter the Ended state. If SEXE = 0 this is the only state transition that applies to a CA or an ACA task.

7.6.6 Transition S2:S1: When a CA or an ACA condition is cleared and the QErr <<bit>> field is set to 00b in the control mode page (see the SPC-2 standard), a task in the Blocked state shall re-enter the Enabled state. When a CA or an ACA condition is cleared and the QErr field is set to 11b in the control mode page, a task in the Blocked state for other than the faulting initiator shall re-enter the Enabled state.

((The state diagrams in the pdf file are not easy to work with, so I have gone strictly by the descriptions. Consequently this needs to be checked against the state diagrams.))

((I elected to not propose what changes are needed for the 7.7 Examples and left that as a task (effort) to be accomplished after the requirements are agreed to.))

Proposal Post Script:

5.6.1.2 Clearing an Auto Contingent Allegiance Condition
An auto contingent allegiance condition shall always be cleared after a power on condition or a hard reset (see 5.6.6). If the NACA bit is set to zero in the CDB control byte of the faulting command, then the SCSI-2 rules for clearing contingent allegiance shall apply. In this case, the logical unit shall also clear the associated auto contingent allegiance condition upon the return of sending sense data by means of the autosense mechanism described in 5.6.4.2.

While the SCSI-2 rules for clearing the ACA condition are in effect, a logical unit that supports the CLEAR ACA task management function shall ignore all CLEAR ACA requests and shall return a service response of FUNCTION COMPLETE (see 6.3). **((Note that this requirement is in conflict with 5.6.1.1. Although not part of this proposal the choice of setting an error or ignoring**

the CLEAR ACA should be chosen by the committee rather than leaving the conflicting requirements.))

5.6.4.1 Asynchronous Event Reporting

An error condition or unit attention condition shall be reported to a specific initiator once per occurrence of the event causing it. The logical unit may choose to use an asynchronous event report or to return CHECK CONDITION status on a subsequent command, but not both.

Notification of command-related error conditions shall be sent only to the device that initiated the affected task. **((Is this in conflict with the ACA requirements with SEXE = 0?))**