From: Gerry Houlder, Seagate Technology <gerry.houlder@seagate.com>
Subj: White paper on existing SCSI low power modes
Date: Feb. 11, 2008

Overview

There are low power states documented in SBC-3 clause 4.16 – START STOP UNIT and power conditions. This provides methods for invoking one of several low power states and methods for reviving the device for active use when needed.

The existing low power states do not provide for lower power modes for the interface, however. This is because the interface is necessary to revive the device. If a method is developed to put the interface into a low power mode, along with a method to quickly revive the interface, then additional power savings can be achieved. This white paper shows how the existing methods can work together to achieve the greatest possible power savings while maintaining reasonable performance for the end device.

The existing low power states

The existing low power states allow for IDLE, STANDBY, and STOPPED states. The IDLE state provides for lower power consumption but with a slight performance penalty. The STANDBY state provides for even lower power consumption but with a higher performance penalty. The STOPPED state provides for the lowest consumption and the highest performance penalty.

Two approaches for controlling low power states

SBC-3 provides for two approaches for controlling low power states – host-controlled and device timer controlled.

The host-controlled approach makes use of the START STOP UNIT command to force the device into IDLE, STANDBY, or STOPPED state. Setting the POWER CONDITION field can cause transition to the IDLE or STANDBY states, or clearing the START bit can cause transition to the STOPPED state. When the host wants to revive the device for access, it can send another START STOP UNIT command with appropriate settings in the POWER CONDITION field and START bit to make the device active again. This provides the greatest predictability to the host, but it also requires the host predict its own activity to take advantage of saving power on a device it may not be accessing in the near future.

If the device is in STOPPED state, a START STOP UNIT command with the START bit set to one is required to activate the device. If the device is in IDLE or STANDBY state the host can revive the device by sending a START STOP UNIT command with the POWER CONDITION field set to zero or by sending an access command (e.g., READ or WRITE) that requires the device to be fully active. This saves having to send a START STOP UNIT command to revive the device, but it also makes the access command have a longer than usual response time. The host must manage the expected response time so that it can tell whether the longer time is expected (because of the lower power state) or whether the longer time indicates a more serious error with the device.

The timer controlled approach uses timers in the end device to control when the device should change to a lower power state. These timers are controlled by the Power Condition page (mode page 1Ah). This page contains separate timer values for IDLE and STANDBY states and separate bits to enable the IDLE and STANDBY states. When the device has not been accessed for the period of time set into the timer the device transitions into a lower power state. The separate timers allow specifying a shorter time for transitioning the device into IDLE state and a longer time to transition it to STANDBY state. When the device receives an access command, it transitions to active state to satisfy the request. As noted in the previous paragraph, the host must tolerate the longer access time of the low power state and not treat it as an error. This is often handled by the host by having a large enough access timeout value that takes into account the longest possible low power state recovery time. In this way the host doesn't need to be aware of whether the device was in a low power state, it only needs to be designed to accommodate the longest possible power state recovery time.

The timer approach is still controlled by the host, in the sense that the timers in the device can be completely disabled so that the device will not transition to a lower power state on its own. Also, if the timers are enabled the values can be set by the host to meet the needs of that particular system.

Neither of these approaches allows for halting the interface transceivers, because the interface is required to bring the device back to the active state. Note that a START STOP UNIT command or some other access command is required to revive the device and this is not possible if the interface is disabled.

Saving transceiver power too

Most protocols treat disabling the interface as an I_T Nexus Loss situation, which then requires extra recovery and device initialization. To get around this, the protocol must design a low power mode for the transceivers that won't trigger the I_T Nexus Loss detection.

Some might argue that the transceiver power (e.g., 100 mw) is small compared to the device power that can be saved using the IDLE, STANDBY, and STOPPED states (e.g., several watts). This is certainly true for most devices like disk drives. However, large systems can have many disk drives and only a few might have to be active at any one time. Turning off transceivers might only save 100 mw each, but if a hundred devices can be turned off this is still a significant power savings that should be addressed.

Another factor is that the recovery time for turning transceivers off and on is much faster (e.g., 10 usec to 10 Msec) than recovery time for IDLE, STANDBY, or STOPPED states (e.g., 1 to 30 seconds). Any time a device is in one of these states, the transceiver can be disabled to save a little more power without a noticeable increase the recovery time for the IDLE, STANDBY, or STOPPED states. Also, a device that is in full active mode (because the host system wants maximum performance from that device) can still save some power by using the transceiver low power modes during lulls in the action without hurting performance. Some devices already use this idea to reduce power consumption of internal circuits during idle periods. Many circuits can be revived within 10 to 100 usec and therefore do not add significantly to the access time of the device.

SAS devices have two ports. Many systems use one port for most accesses and the second port is a backup path that may not be used unless a problem makes the primary path inaccessible. This is a case where power can be saved on the backup port even though the device is fully active to satisfy requests from the primary port, which prevents the use of IDLE or STANDBY states to save power.

Suggested method for transceiver power control

The SATA interface already defines a method to invoke lower power transceiver modes. One end of a link sends a primitive to the other end that requests changing to a low power mode. If the receiving end ignores or rejects the request, both ends continue transmitting normally. If the receiving end acknowledges the request, then both ends stop transmitting. When either ends want to resume transmitting, it initiates OOB (Out Of Band) signaling and sends COMWAKE to the other end. This brings both ends back to normal transmission within a relatively short period of time.

SAS is very similar to SATA in its use of serial interface based on primitives and separate link per device connection architecture. Also, it is possible to have SAS and SATA drives installed in the same backpanel. We should leverage the SATA technique for use with SAS drives and expanders as well. A proposal to accomplish this will soon be presented at the Protocol Working Group.

Suggested enhancements to existing low power states

The power savings for each low power state is vendor specific. There isn’t much that can be done about this because there are many implementations in the marketplace and each has their own options for saving power. Additionally, there is no guideline about how much time it takes to bring the device back to full operation. This has inhibited adoption of these low power states because the host system cannot predict how long the response time to service a command will be unless there is a restricted choice of devices in the system, the system integrator does the recovery time characterization, and the recovery time expectations have been programmed into the I/O handlers accordingly.

A more universal solution to this issue would be to add another field to control the IDLE, STANDBY, and STOPPED states. The field would indicate the maximum recovery time to revive the device from that state. This could satisfy the initiator’s need to predict the response time in these cases. A new proposal would be required for this.

Summary

SCSI already defines lower power states for target devices (IDLE, STANDBY, and STOPPED) and two methods to invoke those states (START STOP UNIT command issued by the initiator or timers located in the target device). These options allow for saving a lot of power in an idle device. This is particularly useful in large systems where most of the devices are idle for long periods of time.

SAS doesn’t have a method to save transceiver power by turning them off when the interface is idle. This option would save a little more power when the device is idle, but can also save some power during short idle periods or on secondary ports of a device that may be very busy on its primary port. A new proposal is needed to make use of the power saving opportunities. Saving power has become an important device feature and we should make use of every opportunity to allow for power savings.