A Serial ATA/ATAPI Proposal

e00121r0
Revision 0 - 21 May 2000
T13 Doc Number (not assigned yet)

by Hale Landis
hlandis@attglobal.net

BACKGROUND, GOALS, ETC

Background

About four years ago I started talking to people about a "serial" version of the ATA/ATAPI interface. Today there are as many as three "serial ATA" efforts in progress around the industry and here is another to add to the confusion! I should add that I have not been, and I am not now, involved in any of these other "serial" ATA proposals or discussions.

Goals of a Serial ATA/ATAPI Interface

My goals are listed here in descending order of priority:

1. Data Integrity and Reliability. A new interface, especially a new ATA/ATAPI interface, is of no value unless it is has better data integrity and better reliability. This means more ability to detect, report and recover from errors in the interface. Current ATA/ATAPI has only one error that can be detected and reported: the UltraDMA ICRC error. A new ATA/ATAPI interface should detect and report errors in command, status and data transfers.
2. Faster Data Transfer Rates. Of course any new interface must have the ability to transfer data faster. And it must have a clear growth path for at least 5 to 10 years into the future.
3. One device per cable. To keep it simple a new interface should be point-to-point with a single host and a single device per cable.
4. Few or very small Host software changes. A new physical transport layer for ATA/ATAPI should not require changes in existing host software. To make this work with PIO commands while the device is using High Speed Data Transfer mode a host adapter will be required to buffer some data and emulate the traditional REP INSW and REP OUTSW host activities.
5. Easy to implement. A new ATA/ATAPI interface, especially a serial interface using a minimum number of interface signals, should be easy to implement. If such a new interface is to be successful it should be equally easy for all types of host devices to implement, everything from handheld hosts to big box hosts. And it should work for a wide range of host speeds. Not all hosts need high speed data transfer rates.
6. Slight increase in cable length. It would be nice if a new interface would support a slightly longer cable. There is no need to make a new interface work for external devices, we have USB and 1394 for that, but something more than 18 inches but no more than 36 inches would be nice.
7. Use existing cable and connector technology. There is no need to define a new cable or connectors for a new ATA/ATAPI interface. If some way can not be found to use the existing device connectors then a cable and connector like the RJ telephone cable and connector should be used. We don't need yet another expensive cable and connector system.
8. Dual interface devices. If possible, devices should be able to implement the existing parallel ATA/ATAPI interface and also implement a new serial ATA/ATAPI interface. And no jumpers should be required in order for a device to detect which interface is being used.
9. Reduce Host Overhead. This isn't really something that a new interface can directly address but the specification for the interface should, unlike the existing ATA/ATAPI standards, clearly state the minimum requirements for a high performance host adapter. A new interface can be implemented in phases and the first phase can implement high compatibility and ignore

the host overhead problem. The second and later phases can move away from full backward compatibility especially full host compatibility. The second phase of a new ATA/ATAPI interface should reduce host interrupt overhead. This can only be done by having a host adapter that has a queue of commands ready to issue to a device so that the device can start the next command while the host software is processing the ending interrupt from the previous command.

Do we need a Serial ATA/ATAPI interface?

A new interface, even the simple one proposed here, has a number of problems. First it is yet another interface and we all know how difficult it is to get these things implemented and widely used... just look at UCB and 1394 as examples. But anything "serial" is the "IN THING", the current industry fad, so such a thing will probably happen even if it isn't the best solution.

The interesting thing about a serial ATA/ATAPI interface is that it must run at speeds of 2G bit today and 4G bit or 8G bit in a few years. Can cheap, easy to implement serial interfaces be designed and built for those data transfer speeds? Maybe a parallel interface with better electrical characteristics is the way to go for high performance systems?

But anything "serial" is the current rage (even when it makes no sense)...

THE PROPOSAL

NOTE: This is revision 0 of this proposal so there are a few things not yet fully specified.

Physical Interface

This serial interface uses four wires, two pairs of wires implementing a differential signal interface between the host and the device and between the device and the host. Each pair of wires is unidirectional.

Note: This interface is designed such that a PIO only prototype can be built using existing ATA/ATAPI host adapters and devices. Such a prototype consists of a parallel to serial board at the host side and a serial to parallel board at the device side. Such a prototype would not implement High Speed Data Transfers (HSDT).

ATA/ATAPI Command, Status and Data Transfers

A serial interface transaction is the exchange of two packets between each side of the interface. One side sends a packet and in many cases the other side returns the same packet or an Interface Error packet. In a few cases the returned packet contains different information.

Full backward compatibility with host software is achieved by mapping each host I/O Read or I/O Write cycle into a serial interface transaction. Serial interface transactions are initiated by the device or by the host. A host I/O Read/Write cycle is converted to a small packet, usually 38 bits, and sent to the device. The device responds by sending the packet back to the host. For I/O Read cycles, the packet returned by the device contains the data read. All ATA and ATAPI reset, Non-Data commands and PIO Data In/Out commands are implemented in this manner.

While a host can read the Data register one word at a time this is not efficient. This interface implements a single method higher speed data transfer. A host can transfer the data for any ATA or ATAPI data transfer command the High Speed Data Transfer (HSDT) method.

A device can also initiate the transfer of an Interrupt packet. This is done anytime the device changes its interrupt pending state and the device's nIEN bit is 0.

Each packet contains a CRC for error checking. Both the host and the device check packets for CRC errors and for Interface Error packets during each serial interface transaction.

The basic packet format is:

- PT (5 bits) = Packet type (and also the five register address bits).
- WR (1 bit) = normally used to indicate "write" or "read", in one packet it is used to indicate "enable" or "disable".
- Count (16 bits) or Data (16 bits) = register data or word count for data transfer packets.
- One or more data words, only in data transfer packets.
- A 16-bit(?) CRC.

The PT field determines the packet type. The possible values of PT are:

- 10aaaB - Command/Status packet - addresses the Alternate Status and Device Control registers, the only valid value of aaa is 110B, all other values are reserved.
- 01aaaB - Command/Status packet - addresses a Command Block register, including the Data register, aaa is the register address.
- 00100B - Interrupt packet - this packet is sent from the device to the host to signal a change in the device's interrupt pending state.
- 00010B - Enable/Disable HSDT packet - enable High Speed Data Transfers, this packet is sent from the host to the device.
- 00001B - Data Transfer packet - transfers data in HSDT mode.
- 11000B - Interface Error packet - used to signal an interface or packet receive error. Packets with this address value contain no other information and the WR bit and transfer count are don't care values. These packets do not contain data.
- 11111B - Hardware Reset packet - used by the host to initiate Hardware Reset.
- All other Address values are reserved.

Each packet type is described in detail below.

Command/Status transfer packet:

- PT = either 10110B (for Device Control and Alternate Status) or 01aaaB (aaa is the Command Block register address).
- WR = 0 for read, 1 for write.
- Data = Sixteen bits of data, for most register accesses only the least significant 8 bits are used.
- CRC.

Interrupt packet:

- PT = 00100B.
- WR = 0 if device has no interrupt pending and 1 if the device has an interrupt pending.
- Count = 0.
- CRC.

Enable/Disable HSDT packet:

- PT = 00010B.
- WR = 0 for disable, 1 for enable.
- Count = The maximum word count allowed by the host in Data Transfer packets.
- CRC.

Data Transfer packet:

- PT = 00001B.
- WR = 0 for read (transfer to host), 1 for write (transfer to device).
- Count = word count, the number of data words that follow.
- Zero or more data words.
- CRC.

Interface Error packet:

- PT = 11000B.
- WR = 0 (reserved).
- Count = 0 (reserved).
- CRC.

Hardware Reset packet:

- PT = 11111B.
- WR = 0 (reserved).
- Count = bit significant list of the interface speeds supported by the host or by the deivce.
- CRC.

HOW PACKETS ARE USED

I/O Write Transaction

The host sends a Command/Status packet to the device. The device receives the packet and checks the CRC. If the packet is invalid the device returns an Interface Error packet. If the packet is valid the data is placed into the addressed Command or Control Block register. The device sends (reflects) the received packet back to the host.

The host checks the packet returned by the device. If the host receives a packet with an invalid CRC or an Interface Error packet there was an interface transmission error. If the Host received an Interface Error packet the host hardware or software should send the Command/Status packet to the device again.

I/O Read Transaction

The host sends a Command/Status packet to the device. The device receives the packet and checks the CRC. If the packet is invalid the device returns an Interface Error packet. If the packet received is valid the device inserts the request register data into the packet, computes the CRC and returns the packet to the host.

The host checks the packet returned by the device. If the host receives a packet with an invalid CRC or an Interface Error packet there was an interface transmission error. If the Host received an Interface Error packet the host hardware or software should send the Command/Status packet to the device again.

Interrupt Packet

Whenever the interrupt pending state of a device changes and if the nIEN is 0, the device sends an Interrupt packet to the host. The host acts upon the new value of the device's interrupt state and the host sends (reflects) the Interrupt packet back to the device unchanged.

Enable HSDT Data Transfers

High speed data transfer is achieved using a single data transfer protocol. The host can use this method to transfer the data for any ATA or ATAPI data transfer command. If a host plans to use High Speed Data Transfers (HSDT) then the host must send a Enable HSDT packet to the device before executing any data transfer commands using HSDT.

When the device receives an Enable/Disable HSDT packet it checks the CRC. If the packet is invalid the device returns an Interface Error packet to the host. If the packet is valid the device enables or disables HSDT and for enable it saves the maximum host word count for future use.

When a device is ready to transfer data it sends a "data transfer" packet to the host. There are two formats of this packet (two transaction types), one for a "write" transfer (data transfer to the device and a "read transfer" (data transfer to the host).

HSDT Write Transfers

The device sends a "write data request" packet to the host:

- PT is set to 00001B.
- WR is set to 1.
- Transfer count set to the number of words the device is requesting from the host, not to exceed the value specified by the host in the Enable/Disable HSDT packet. A value of 0 indicates the device has completed the "write" data transfer.
- CRC.

The host should check the packet validity. If any error the host returns an Interface Error packet to the device and updates any host interface error indicators. If the packet is valid the host returns the following "write data response" packet to the device:

- PT is set to 00001B.
- WR is set to 1
- Transfer count set to the number of words the host is sending to the device, not to exceed the maximum number of words from the most recent Enable/Disable HSDT packet exchange.
- The data words being returned to the host.
- CRC.

HSDT Read Transfers

The device sends a "read data request" packet to the host:

- PT is set to 00001B.
- WR is set to 0.
- Transfer count set to the number of words the device is sending to the host, not to exceed the maximum number of words from the most recent Enable/Disable HSDT packet exchange. A value of 0 indicates the device has completed the "read" data transfer.
- The data words being transferred to the host.
- CRC.

The host should check the packet validity, store the data and update any host interface error indicators. The host returns the following "read data response" packet to the device:

- PT is set to 00001B.
- WR is set to 0.
- Count is set to 0.
- CRC.

Hardware Reset

Hardware Reset is initiated by the host sending one or more Hardware Reset packets to the device. Following "power on", the host waits 50ms and then sends Hardware Reset packets to the device until a Hardware Reset packet is received from the device. At any other time the host can send a Hardware Reset packets to initiate this reset. The Count field of the packet sent to the device contains a bit significant list of the serial interface speeds supported by the host. When the device receives this packet it initiates Hardware Reset processing (sets BSY=1, etc) and it returns to the host a Hardware Reset packet with a bit significant list of the serial interface speeds that are supported by both the host and the device. As soon as the device sends its Hardware Reset packet back to the host the device switches to the highest serial interface speed that is supported by both the host and the device. When the host receives the Hardware Reset packet from the device the host switches to the highest serial interface speed supported by both the host and the device. All hosts and devices are required to support the same minimum speed corresponding to bit 0 of the Count field.

EXAMPLES

Note that the host and device sides of this interface can be implemented just as described by ATA/ATAPI-5. However such an implementation does not take advantage of HSDT for PIO data transfer commands. DMA data transfer commands require the host to enable HSDT.

Hard Reset Example

The host and device exchange Hardware Reset packets and both set the serial interface speed. The host reads (polls) the Status or Alternate Status registers waiting for the device to set the BSY to 0.

Software Reset Example

The host writes the Device Control register with SRST set to 1 and then again with SRST set to 0. Then the host reads (polls) the Status or Alternate Status registers until the device sets BSY to 0.

ATA/ATAPI Non-Data Example

The host writes the command parameters and the command code to the Command Block registers and then waits for an interrupt or reads (polls) the Status or Alternate Status registers until the device sets BSY to 0.

When the command register is written the device sets BSY to 1, executes the command and then sets BSY to 0 and if nIEN is 0 the device sends a Interrupt packet to the host.

ATA/ATAPI PIO Data In Example

The host writes the command parameters and the command code to the Command Block registers and then waits for an interrupt or reads (polls) the Status or Alternate Status registers until the device sets BSY to 0. If the device has BSY=0 with DRQ=1 status then the host transfers the DRQ data block to host memory using the standard REP INSW (or similar)

instructions just as is done for today's parallel ATA/ATAPI interface. Without HSDT enabled the host is reading the device Data register one word at a time.

ATA PIO Data Out HSDT Example

Assume the host and device have exchanged an Enable/Disable HSDT packet that enabled HSDT since the last hardware reset. When HSDT is enabled the data transfer on the interface looks just like the data transfer for a DMA command. It is expected that the host software will set up a DMA transfer just as it does for DMA commands and wait for command completion just as it does for DMA commands.

PACKET PIO HSDT Example

Assume the host and device have exchanged an Enable/Disable HSDT packet that enabled HSDT since the last hardware reset. When HSDT is enabled the data transfer on the interface looks just like the data transfer for a DMA command. It is expected that the host software will set up a DMA transfer just as it does for DMA commands and wait for command completion just as it does for DMA commands.

ATA DMA Example

The host writes the command parameters and the command code to the Command Block registers and then waits for an interrupt or reads (polls) the Status or Alternate Status registers until the device sets BSY to 0.

When the Command register is written the device sets BSY to 1 and prepares to transfer data. When the device is ready to transfer data it sends Data Transfer packets to the host. The host responses by sending Data Transfer packets back to the device.

When the data transfer is completed the device sends and if nIEN is 0 the device sends an Interrupt packet to the host.

PACKET DMA Example

The host writes the command parameters and the command code (A0H) to the Command register and then reads (polls) the Status or Alternate Status registers until the device sets BSY=0 with DRQ=1 requesting the command packet.

The host writes the command packet to the device.

The device sets BSY to 1 and prepares to transfer data. When the device is ready to transfer data it sends Data Transfer packets to the host. The host responses by sending Data Transfer packets back to the device.

When the data transfer is completed the device sends and if nIEN is 0 the device sends an Interrupt packet to the host.

/end/