ATA Command Queuing

October 13, 1994

James McGrath

Systems Engineer
Strategic and Technical Marketing

Quantum
500 McCarthy Blvd
Milpitas, CA 95035

phone: 408-894-4504
fax: 408-894-6375
internet: JMCGRATH@QNTM.COM

Document History

- General Note: revisions are coming out fast, prompted by good feedback - do not consider things set in stone, give me feedback!
- Revision 5: feedback from revision 4:
  - -- added note in slide 9 and appendix A to deal with HEAD OF QUEUE
  - -- made slide 16 consistent with 15 in use of lock and arbitration bits
- Revision 4: first document for general distribution, targetted at the October 19 ATA-3 meeting. (October 13, 1994). This is an architectural overview, with some key decisions (e.g. how to pass the tag) and timing considerations still loosely defined. The key is to look at the forest first, then the trees.
- Revision 2 and 3: incorporate private feedback from several companies. (October 11, 1994)
- Revision 1: initial proposal (October 4 1994)

Agenda

- Why Command Queuing on AT Drives?
- Why the Drive and not the Host?
- IO Subsystem Constraints
- Host Constraints
- ATA is Simplier than SCSI Command Queuing
- ATA Protocol Changes Needed for Command Queuing
- Configuring the Host for Queuing
- Queue Tags
- Synchronizing the Host and the Devices
- Implementing Locks (Host to Device)
- Host Sending a Command to a Device
- Arbitration Between Two Queuing Devices
- Implementing Locks (Device to Host)
- Status and PIO Data Transfers
- DMA Data Transfers
- Transfer Count
- Error Recovery

Why Command Queuing on AT Drives?

- Goal is higher drive performance at little or no additional cost

  -- SCSI workstation class drives have demonstrated16 commands) at no incremental cost

- Will a Queue of commands exist in PC systems?

  -- A key enableations must take advantage of the OS

  -- Multi-tasking and workplace environment will generate more IOs

  -- Performance gain will be workload dependent

  -- Benchmarks will be adapted to show off queuing at its best

- Although protocol needs to be defined now to allow for ASIC and device driver development, we do not see a market need until 1996 or beyond.

ATA Command Queuing

Why the Drive and not the Host?

Source of Performance Gain	Relative Importance	Drive Capable	Host Capable
Access Time Reordering	1	Yes	No
Seek Time Reordering	2	Yes	Yes
Command Overhead Overlap	3	Yes	No

Why Overlap Commands Between Two Devices?

- In a well balanced IO subsystem, twice the throughput can be achieved by overlapping command execution (e.g. a simple mirrored system).
- When one device is very slow (e.g. CD-ROM, Tape), this allows some overlap of fast disk IO with slow device IO.

IO Subsystem Constraints

- The IO subsystem consists of the host and one or two devices sharing an IDE cable (secondary address/cables are just a duplicate IO subsystem, although the primary addresses are used as examples throughout).
- A component (host, device) can be queuing capable or non-queuing capable (a legacy componentction with a legacy host, so queuing devices must power up in a queuing disabled mode and have queuing enabled by the host.
- The queuing protocol proposal is designed to address the case of a queuing host and one or two queuing devices.
- With a queuing host and a legacy device, the host simply does not invokeap commands between devices. It must also inform the queuing device so it can work in a non-arbirtaration mode.
- ATA and ATAPI should be able to mix on the same cable (at least as much as they can do today).
- Although focused on ATA commands, this proposal also works for ATAPI.

Host Constraints

- Queuing is not designed to work directly on the ISA expansion bus - local bus only will be supported.
  - -- Current proposal should work on ISA expansion buses, but we may wish to simplify some of the protocol and violate this compatibility.
- Host connection is via a bridge (which may be discrete or integrated into system hardware) connecting a dedicated ISA bus to PCI/VL-Bus/EISA/MCA/other.
- Since bridges must be supported, bridge constraints must be taken into consideration:
  - -- Intel PCI bridge guidelines allow access only to 3F6 and the task file
  - -- PCI bridges are moving towards first party DMA, implying no CPU intervention for data transfer. Therefore all drive data transfer protocol must be simply enough to be handled by a bridge chip.
- Changes requiring new host hardware is allowed if the cost is small.
- Queued and non-queued commands cannot be mixed on the same device.

ATA Command Queuing

ATA is Simpler

- ATA has only one initiator
  - -- Number of available queue slots always known
  - -- Error recovery can be simple

- ATA has no command reordering restrictions
  - -- All commands are treated like SCSI SIMPLE queue tag commands - drive can freely reorder any commands it receives
  - -- To insure strict order of execution (ORDERED queue tag), the host does not queue the command until the drive is idle - and does not queue the following command until the first has completed.
  - -- Commands cannot jump the queue (HEAD OF QUEUE queue tag does not exist) - although resettingles has suggested a less brutal technique - see appendix A)

- From the upper level of system software, ATA and SCSI command

ATA Protocol Changes Needed for Command Queuing

- A legacy (non-queuing capable) device can co-exist with a queueing device. While commands can be queued at the queuing device, commands cannot be overlappedevice is executing another command.
- Commands queued or executing at the device must be uniquely identified so that data and status can be returned to the host in a different order than the commands were received by the device.
- An arbitratingthe ATA bus must be developed so that command execution can be overlappedally well with PIO and DMA commands.
- An error recovery procedure must be specified
- Ability for the host to detect queuing capability, the maximum queue size, and the ability to activate and deactivate the queueing protocol

Configuring the Host for Queuing

- By default devices power up in a non-queuing mode.
- The host can determine if queuing is supported, and the maximum size of the queue, by issuing the IDENTIFY DRIVE command to the device.
- Queuing is enabled/disabled by using the SET FEATURES command.
- Arbitrationenabled vs devices without queuing capability or those with queuing disabled.
- Queuing and arbitrationa hardware reset.

Queue Tags

- The host assigns each command an ID number, a queue tag.
- Queue tag values range from 0 to (maximum queue size - 1).
- Queue tag values are re used as commands are completed.
- Queue tags are passed between the host and the device via either:
  - -- the data fifo, or
  - -- the upper byte of 3F6 accessed as a word.
- Which of the above to do is still to be decided - the final document will specify only one (there will be no options)

Synchronizing the Host and the Devices

- Today ATA does not provide a way to insure the host can access the task file and device control registers without device interference, and vica versa.
- A new lock request/lock granted mechanism is provided to provide such exclusive access at critical points in the command queuing protocol.
- While a device or the host is locked, it cannot modify the contents of the task file or device control registers with the following exceptions
  - -- The host can always write to the soft reset and interrupt control bits.
  - -- The locked device can always make new data available via the data fifo register.
  - -- The locked device can always modify a few status bits that are not latched (e.g. index).
- A lock can be though of as a SCSI connection. Although you may think of the locking mechanism analogousECTION mechanism of SCSI, note that a host always locks BOTH devices.

Implementing Locks (Host to Device)

- Host locking all devices
  - -- A host requests a lock by setting a bit in writing a word at 3F6 .
  - -- Device 1 grants a lock request by asserting PDIAG within 25 ns and holding it for a short period (e.g. 50 ns) to signal device 0. Device 0 grants the lock request allowing the host to read a 1 in the device's lock granted bit when it reads a word at 3F6. The grant must be available immediately (e.g. within 50 ns of request), and implies that BOTH device 0 and 1 have granted the lock.
  - -- In systems with a single queuing device, then that device can grant the lock without coordinating with the other device. Devices can tell if they are in a single queuing environment if queuing and arbitration are enabled.
  - -- A host considers the lock to be granted if the lock granted bit is set.
  - -- A host can clear a lock explicitlyode register in the task file.

ATA Command Queuing

Implementing Locks (Host to Device)

3F6 word access

	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
Host Write/Drive Read	AE	LRLC	R	Queue Tag (or Reserved)				
Host Read/Drive Write	AE	LC	D#	Queue Tag (or Reserved)				

LRLC = 1 for requesting a lock, 0 for clearing a lock

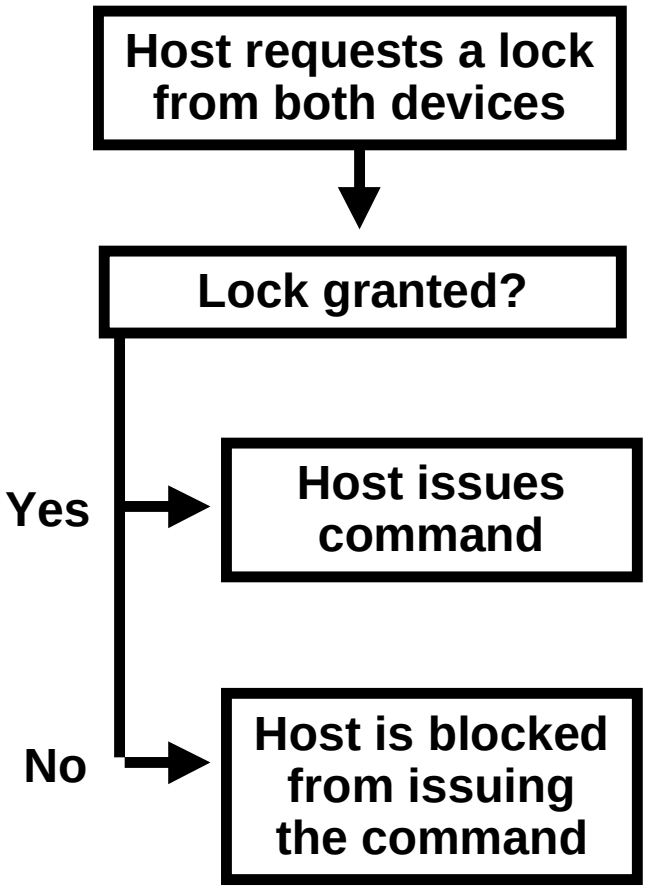
AE = arbitration enabled

LG = lock granted for both devices

D# = 0 or 1, used to indicate device requesting a lock

Host Sending a Command to a Device

- Assume two devices with queuing enabled



Host writes a 1 to bit 14 of device control register 6 (3F6)

Host reads device control register 6 (3F6) and checks if bit 14 is set

Host writes the tag into the bits 13-8 of 3F6 or the data fifo, the rest of the task file as normal, and finally writes to the op code register. Locks are implicitly cleared when op code is written.

Host writes a 0 to bit 14 of device control register 6 (3F6) to clear any locks

Arbitration

- If a device wishes to generate an interrupt to the host for data or status transfer, or to have the DMA controller transfer data, then it must make sure the other device will not interfere.
- A pre condition for this protocol is the host enabling arbitration device may always assume arbitrationers in preparation for arbitratingthe lock.
- Device 0 can generate a interrupt or assert DREQ as long as the device is not locked and PDIAG is not asserted by device 1. The device is said to have won arbitrationot locked and PDIAG has been asserted by device 1 for 500 ns without Device 0 generating an interrupt or asserting DREQ. The device is said to have won arbitrationear that the host will not react to the arbitration

Implementing Locks (Device to Host)

- Device locking a host

  -- A lock is requested by the device by asserting IRQ (status and PIO data) or asserting device can conflict since this device has won arbitrationetting a host to device lock (for status and PIO data) - the host requests a lock, and is immediately granted it by the devices. Another mechanism may be evolved. For DMA data, the lock is granted by the host asserting DACK.

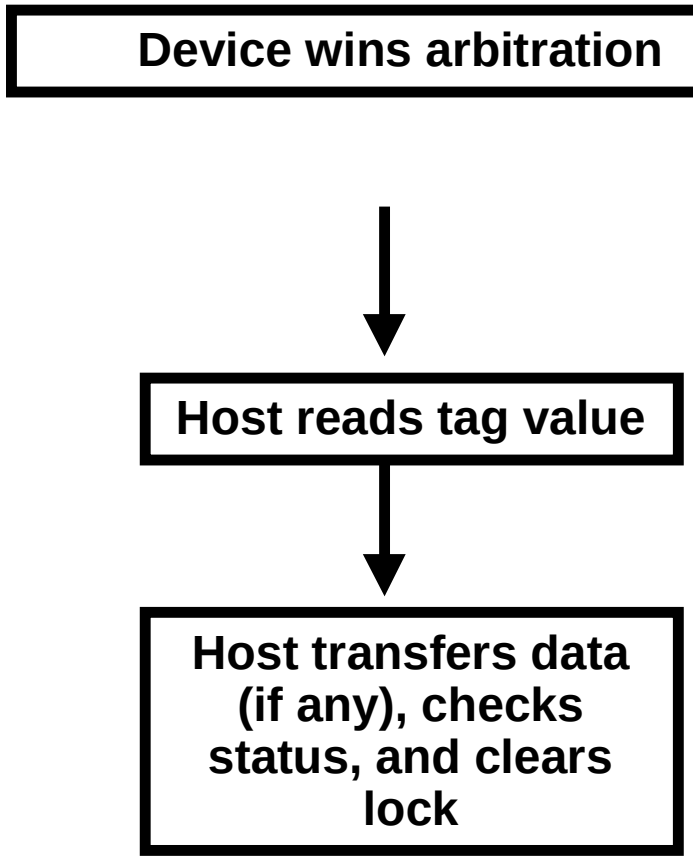
  -- The host clears the lock in the same way as clearing a host to device lock (for status and PIO data) - writing a 0 to bit 14 of 3F6. For DMA, the lock is cleared when both DREQ and DACK are deasserted.

  -- The host or DMA controller can identify the command for which a lock is being requesting by examining bit 13 of device control register 6 (the device number, 0 or 1) and bits 12 to 8 of the same register (the queue tag). Alteratively

ATA Command Queuing

Status and PIO Data Transfers

- Assume two devices with queuing enabled

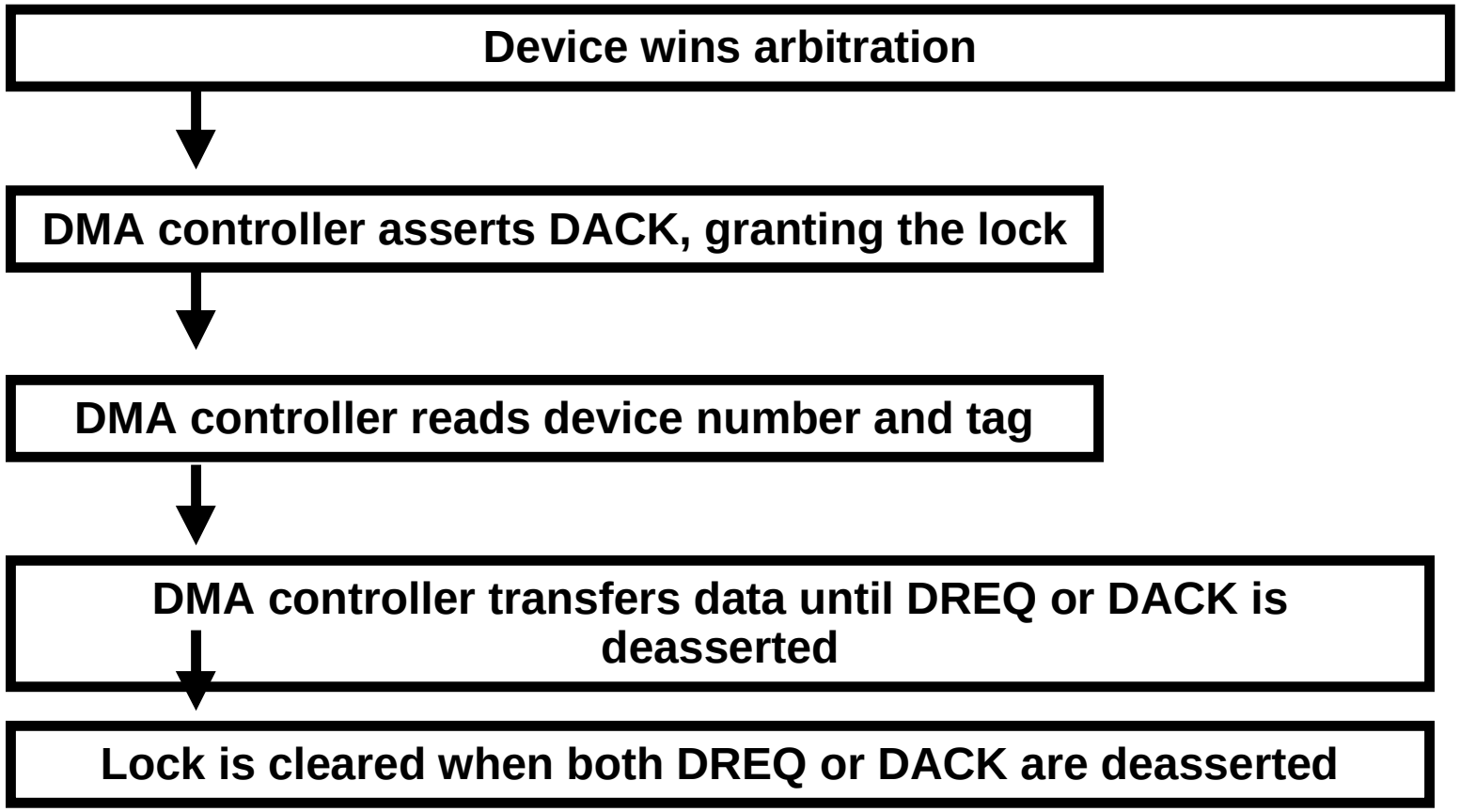


Host reads from bits 13-8 of 3F6 or the data fifo register

Host writes a 0 to bit 14 of device control register 6 (3F6) to clear the lock

DMA Data Transfers

- Assume two devices with queuing enabled



Transfer Count

- After reading the queue tag, the host may read a transfer count word from the device's data fifo register.
- This value indicates the number of words of data available from the device for immediate transfer (i.e. words of data in the device buffer) for reads. It indicates the number of words of space available in the device for immediate transfer (i.e. words of space in the device buffer) for writes.
- For PIO commands, this value is 256 for backward compatibility, but allows a larger, variable number of words to be transferred at a time.
- For DMA commands, this value may not indicate the length of the current DMA burst, but does limit the length of the burst.
- The transfer count mechanism may not be needed, but may assist the host and DMA controller in making transfers more efficient.

Error Recovery

- Errors are reported in the STATUS register as usual.
- The devices continue execution as normal when errors occur - there is no halting of the queue.
- The host can always abort everything via a soft reset - all the commands are wiped clean.
- Individual commands can be aborted by locking the devices and issuing a NOP to the same device/queue tag as the command to be aborted. The drive will return with ABORTED command status.
- Since ATAPI allows the gathering of more extensive sense information via REQUEST SENSE, the device shall maintain sense data for each command, discarding it when the queue tag is re used by the host. The REQUEST SENSE command packet must specify the tag for which sense is requested (there are lots of reserved bytes).

Appendix A: another way to do HEAD OF QUEUE

Commant from Charles Monia (SAM editor)

Just one comment on your ATA queuing slides. On pp 9, you suggest that simulating HOQ commands would require the host-resident SCSI emulation software to abort and reissue previously pending commands.

As I describe below, it is possible for the host to simulate the behavior for HOQ and all other command types without going to those lengths.

One way to do that is for the host to maintain a host-resident queue of pending commands, as your slide suggests. A command is unconditonally entered on this queue when it is received and remains there until completion. The host marks commands on the HR queue that have been passed to the drive.

If an ordered command arrives and the HR queue is empty, the ordered command is immediately forwarded to the drive. Otherwise, the pending command is placed on the HR queue and forwarded when all previous commands have ended.

If a Simple command arrives and there are no pending HOQ or Ordered commands, the Simple command is forwarded to the drive. Otherwise, it remains on the queue and is forwarded when all previous HOQ and ordered commands have ended.

If an HOQ command arrives, it is unconditionally forwarded to the drive.

Even though there's no way to for such a command to 'jump the line' of commands that have been forwarded, that is not an issue since commands previous to an HOQ command are allowed to complete before the HOQ command completes.

I assume in all these cases that an error condition unconditionally aborts all commands that have been passed to the drive. Since all such commands would still be in the HR queue, the host-resident emulation software could just restart them when the ACA is cleared.