ATA Tagged Command Queuing

Conner Peripherals Proposal

v1.0
November 4, 1994

Technical Editors:

Mark Gurkowski	(303) 682-8317	Mark.Gurkowski@conner.com
Lane Lee	(303) 682-8396	Lane.Lee@conner.com
Don Clay	(303) 682-8321	Don.Clay@conner.com

Goals

The goals for this definition are as follows:

- Use edge driven interrupts.
- Minimize host polling requirements.
- Maintain compatibility with current silicon.
- Allow error reporting after transfer on reads.
- Allow for a minimum queue depth of 16.
- Allow two drives.
- Gain OS support for queuing.
- Allow the drive to disconnect before command completion.
- Support PIO multiple mode and DMA mode.
- Support only simple queued commands, but design for simple, ordered and head of queue.
- Keep as simple as possible.

Overview

ATA tagged command queuing allows the host to issue multiple commands to either one or two drives and provides a mechanism for these commands to be executed in any order by the drive. Each command has an associated "tag" value which is used to identify the command during its execution. After completion of the command, the tag is free to be used on subsequent commands.

From initial power-on, the drive behaves as an ATA-1 or ATA-2 device. Tagged queuing is activated when the host issues the appropriate set features command. The only commands which are allowed when command queuing is active are read/write multiple, read/write DMA, read verify, set multiple, and a set features command to disable command queuing. Since high throughput rates are desirable during data transfer, sector mode in PIO is not supported, only multiple mode.

The protocol is broken down into three phases: command phase, data phase, and completion phase. The command phase is initiated by the host and allows the host to issue commands to the drives. The data phase and completion phase are initiated by the drives in order to transfer data or signal the end of a queued command. Before either of these phases, the drives arbitrate for control of the bus, thereby eliminating complications caused by the overlapping of simultaneous interrupt requests.

One hardware change is required to implement command queuing with an ATA-1 or ATA-2 device. Currently the HIRQ line cannot be driven unless the drive is selected in the drive/head register. External hardware is required to overcome this deficiency.

Two methods are proposed to handle the host/drive interlock required during the data and completion phases. The first method (Method 1), uses a host-controlled bit as an acknowledgment signal. This method requires no extra hardware and removes certain requirements on the host. The second method (Method 2) uses reading of the status register as the acknowledgment signal. This method requires hardware capable of detecting read of the status register for the selected drive. Additionally, the host is required to use the alternate status at all times other than when it wishes to make an acknowledgment. Both methods are detailed below.

Register Definitions

When tagged queuing is enabled, the error register, precomp register and status register change their meanings. The Error register is changed to include the command tag, with the error conditions being consolidated into two bits:

7	6	5	4	3	2	1	0
ERR1	ERR0	DTG5	DTG4	DTG3	DTG2	DTG1	DTG0

- ERR1 & ERR0 are a two bit code which the drive sets up before returning an error. These bits are only valid if the ERR bit is set in the status register during the completion phase. The error codes are as follows:

ERR1/ERR0	**Error condition:**
00	Write Fault / Write Error (?)
01	Uncorrectable Data / Read Error (?)
10	Command Aborted
11	ID Not Found/All Other Errors (?)

  These errors are the only error codes supported. All errors besides write fault, uncorrectable data, and command aborted are lumped into the '11' error code. There are no separate error conditions reported in the status register.

- DTG5-DTG0 contain the tag for the current command in the data phase or completion phase.

The Precomp register is changed so that the host may issue a tag with a command:

7	6	5	4	3	2	1	0
rsvd	HACK	HTG5	HTG4	HTG3	HTG2	HTG1	HTG0

- Method 1: HACK is the host interrupt acknowledge and end of data/completion phase acknowledgment signal. The host asserts this signal to acknowledge the beginning of either the data phase or completion phase. The host negates this signal to acknowledge the end of either the data phase or completion phase.
  Method 2: This signal is not needed.
- HTG5-HTG0 is the host tag value. In the command phase, the host places a command tag value in this register before issuing the command.

The Status register is changed to allow additional queuing control signals:

7	6	5	4	3	2	1	0
BUSY	DRDY	DISC	CCPT	DRQ	DTPH	IDX	ERR

- BUSY is set during the command phase to indicate the drive has control of the command block registers. During the data phase and completion phase, busy will only be set once the drive "owns" the task file.
- DRDY - no change.

- DISC (disconnect) indicates to the host that the drive wishes to end either the data phase or the completion phase.
- CCPT (command complete) indicates to the host that a completion phase has been initiated.
- DRQ (data request) is set during the data phase when the drive is ready to transmit or receive data.
- DTPH (data phase) indicates to the host that a data phase has been initiated.
- IDX (index) - no change.
- ERR (error) indicates to the host that an error occurred during a command. This bit may only be set during the completion phase.

Queued Command Definitions

Queued commands use the following command codes:

Cmd Code	Command	Queue Type
40	Read Verify	Simple
41		Ordered
42		Head of Queue
C6	Set multiple size	Queue Empty
D0	Read PIO Queued	Simple
D1		Ordered
D2		Head of Queue
D4	Read DMA Queued	Simple
D5		Ordered
D6		Head of Queue
D8	Write PIO Queued	Simple
D9		Ordered
DA		Head of Queue
DC	Write DMA Queued	Simple
DD		Ordered
DE		Head of Queue
EF	Set Features (enable/disable queuing)	Queue Empty

Notes:

- Conner would prefer to support only simple queued commands, in which case the above table may be simplified significantly. Shown here is a solution to support all three queue types.
- All PIO reads and writes will use the multiple mode transfers. The set multiple command may be used to change the size of the data block transferred.
- The set multiple command and set features command may only be sent when there are no other commands pending in the queue.
- There is no support for disabling retries.

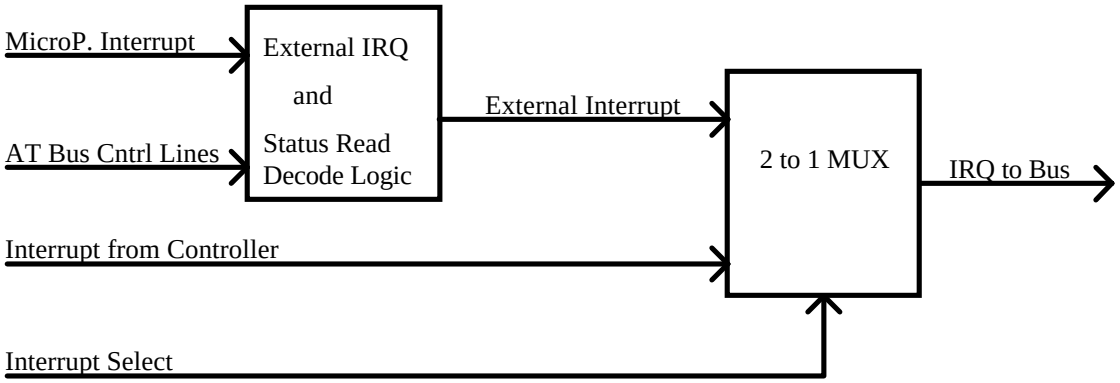
Hardware Changes and Bus Definitions

In general, current controllers cannot drive the HIRQ line unless the drive is currently selected. Additionally, many controllers will clear the HIRQ signal on a read of the status register to the other drive.

For these reasons, an external hardware method for driving the HIRQ line is needed. In order to meet the requirement that the drive power-on in an ATA-1 or ATA-2 compatible manner, a method to use either the controller's HIRQ or the new external HIRQ is required.

For Method 2 of host/drive interlock, status read decode logic is required which can detect a read of the status register if and only if the drive is selected. Some controllers will "see" a status read when the other drive is selected. In this case, external status read decode logic will be required.

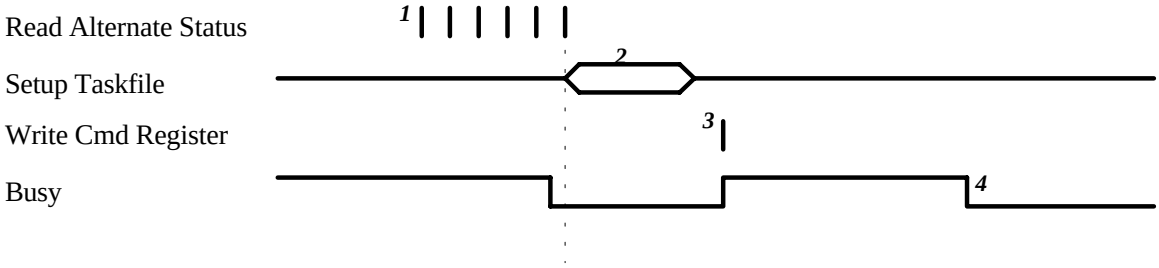
A block diagram showing the necessary changes follows:



Two lines of the bus are redefined, once the drive has been placed into queuing mode, in order to accommodate arbitration for the IRQ line. The changes are as follows:

Queued Definition	ATA-2 Acronym	Queued Acronym	Pin
Drive 0 Owns HIRQ	DASP*	OWN0*	39
Drive 1 Owns HIRQ	PDIAG*	OWN1*	34

Command Phase Protocol



1) Method 1: Host reads Status or Alternate Status until the drive is ready and not busy.
   Method 2: Host reads Alternate Status until the drive is ready and not busy.

   The host will ignore all other bits in the status register except the drive ready and busy bits.

2) Host selects drive and writes task file with the parameters for the command.

3) Host sends command.

4) Drive sets busy until the command has been taken from the task file. When the drive has all the information out of the task file, it clears busy. No errors are reported at this time. Clearing busy does not imply command acceptance - an error condition may be posted at the completion phase. The host is not required to wait for busy to clear.

Notes:

- There is no interrupt sent to the host for this phase. This eliminates the problem of the other drive issuing an interrupt for a data or completion phase just after the host writes to the command register, and the host misinterpreting the interrupt as the end of the command phase.
- The host may not write the drive select register until busy is cleared.
- In some current controllers, the busy bit will be set when the host writes the command register even if the drive is not currently selected. Once the firmware inspects the drive select bit and sees that the command is not for that drive, busy is then cleared. Unfortunately, if the drive for which the command was intended clears busy more quickly, and the host changes the drive select bit, neither drive is selected. A subsequent read of the status register will return unpredictable results, unless there are pull-ups or pull-downs on the data bus. This problem exists within both the ATA-1 and ATA-2 protocols.

  Possible solutions include a pull-up resistor on bit 7, thereby making BUSY look set even if the bus is floating. The host would have to continually reselect the drive while waiting for busy to clear, just in case it is looking at a floating bus. Another solution would be to have a handshake between the two drives that informs the selected drive when it could clear busy. A third solution is to require the host to compare the status register for an exact value, rather than just inspecting the BUSY bit. Finally, if the controller does not assert busy when it is not selected, then there is no problem.

Method 1:

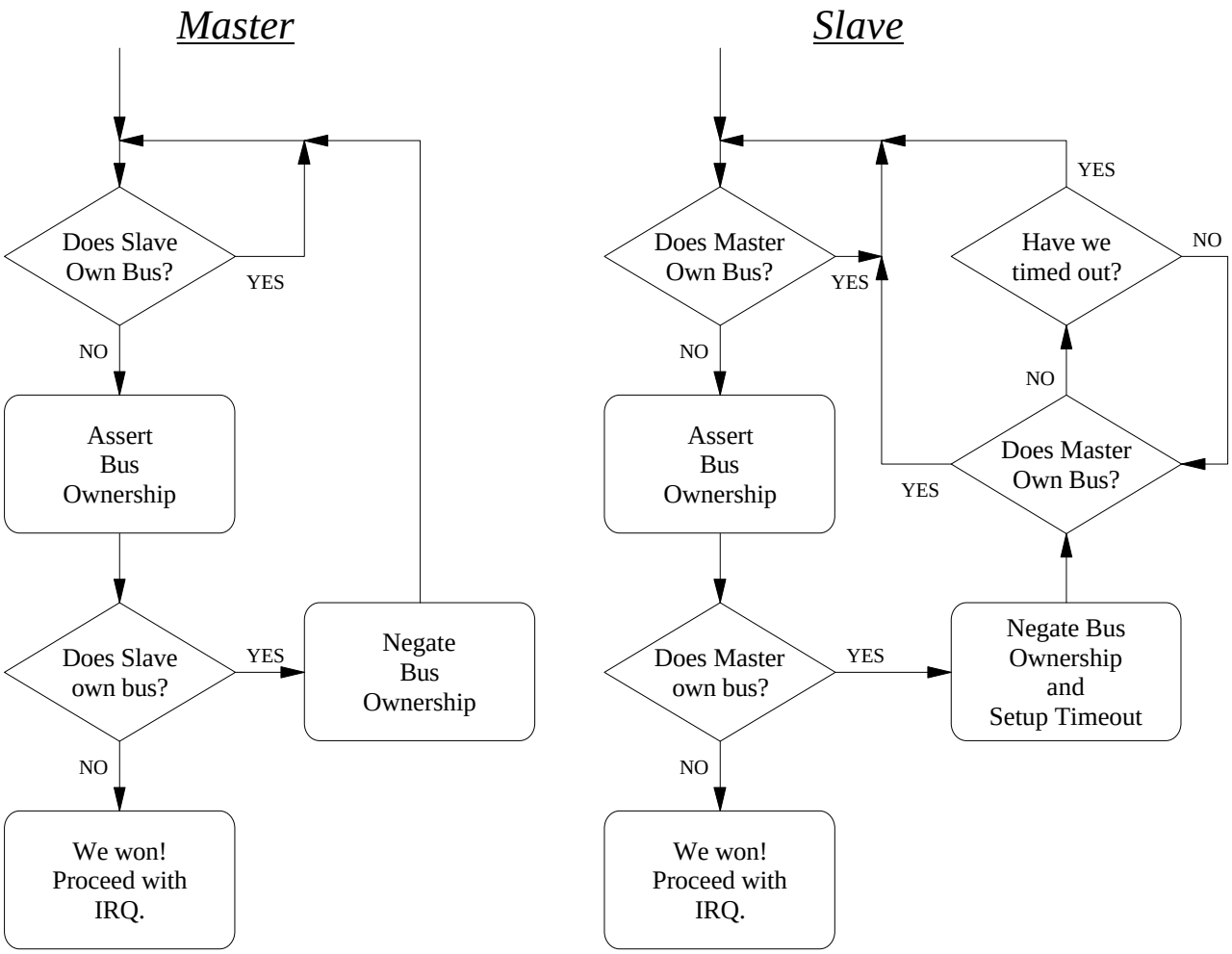
- The host may not send a command while HACK is asserted in the host tag register.

Method 2:

- The host must read the alternate status register, rather than the status register, preventing an accidental clearing of an IRQ. For Method 1, there are no restrictions, because the status read does not affect the state of the IRQ.
- The host may only send a command if the drive is not busy, and there is no currently active data phase.

IRQ Arbitration Protocol

Before either drive may enter into the data phase or the completion phase, it must arbitrate with the other drive for "ownership" of the HIRQ line. Two lines on the bus, formerly DASP* and PDIAG*, are used to perform the arbitration. The following flowchart depicts the arbitration protocol:

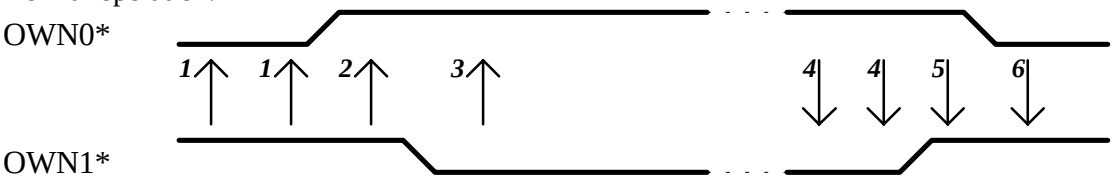


Notes:

- The amount of time that the slave uses as a time-out waiting for the master to assert bus ownership may be any length of time. In fact, the preferred method would have the time-out be a randomly chosen length of time for each arbitration loop.
- After winning arbitration, the slave may assert OWN0* (this is the DASP* line which is used for the LED by the host). The slave must negate OWN0* before relinquishing control of the bus.

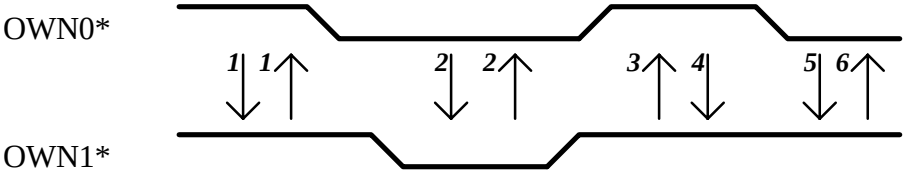
The following diagram provides some examples of arbitration. The down arrow indicates a point at which the master inspects the slave's ownership line. The up arrow indicates a point at which the slave inspects the master's ownership line.

Normal operation:



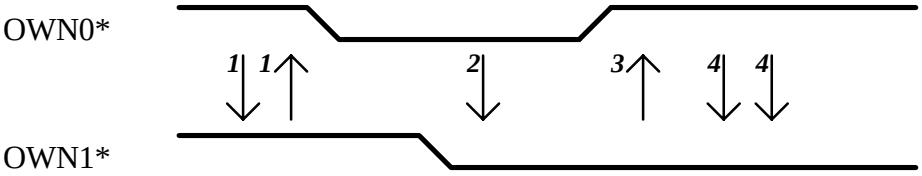
1) Slave waits for bus to be free.
2) Slave sees bus free. Asserts ownership.
3) Slave rechecks after assertion and assumes ownership.
4) Master waits for bus to be free.
5) Master sees bus free. Asserts ownership.
6) Master rechecks after assertion and assumes ownership.

Simultaneous assertion - master wins:



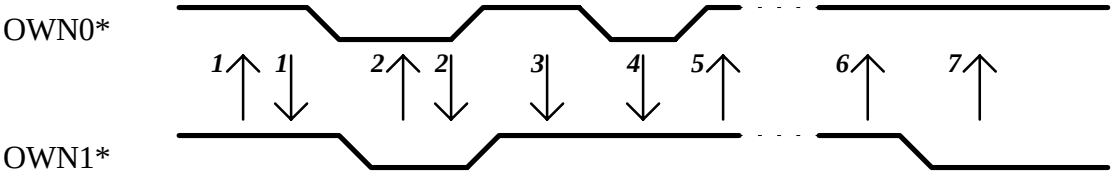
1) Master and slave see IRQ free and assert ownership.
2) Both recheck after assertion and see the other drive's assertion. Both negate.
3) Slave gives up bus until master takes it.
4) Master sees bus free and attempts to take ownership.
5) Master sees bus free and assumes ownership.
6) Slave sees master assert ownership and waits for master to negate.

Simultaneous assertion - slave wins:



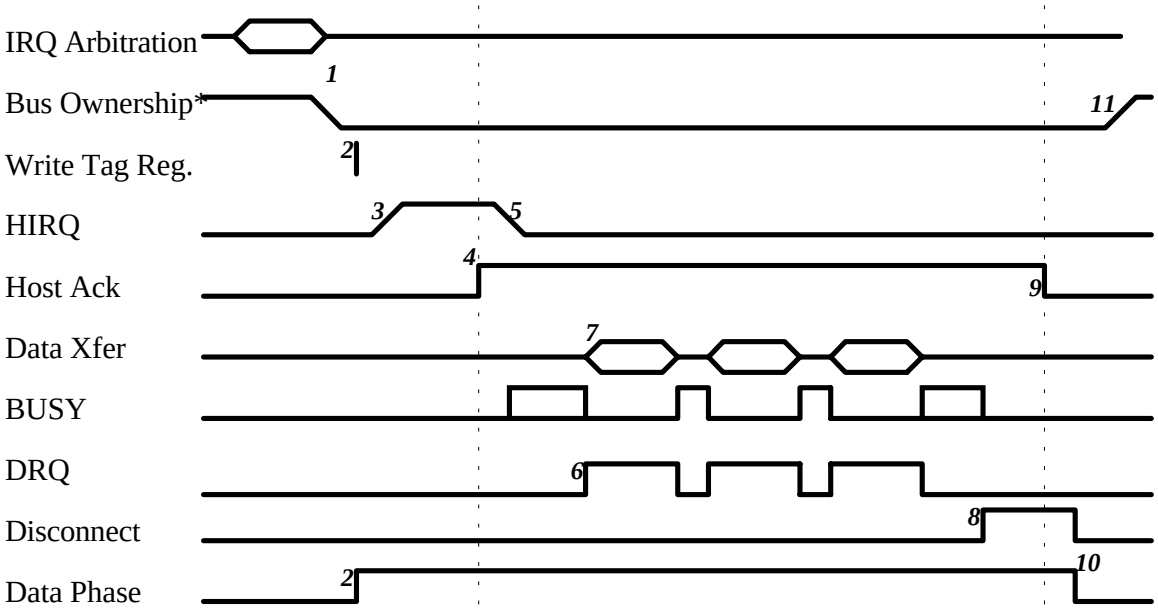
1) Master and slave see IRQ free and assert their lines.
2) Master rechecks after assertion and see the slave's assertion. Master negates.
3) Slave sees bus free and assumes ownership.
4) Master waits for slave to negate.

Simultaneous assertion - slave time-out:



1) Master and slave see IRQ free and assert their lines.
2) Both recheck after assertion and see the other drive's assertion. Both negate.
3) Master sees bus free and attempts to take ownership.
4) Master sees bus free and assumes ownership. Master completes phase, but the slave doesn't see it.
5) Slave waits until Master assumes ownership.
6) Slave times out waiting for master.
7) Slave assumes ownership.

Data Phase Protocol



1) Drive arbitrates for bus. Upon winning the bus, the drive asserts ownership and holds it throughout the data phase.

2) The drive asserts the data phase bit in the status register. The drive writes the tag for the command to be serviced.

3) Drive asserts IRQ. The status at this time will be: Data Phase set, and BUSY, DRQ, Disconnect, and Command Complete cleared.

4) Method 1: Host responds to IRQ when ready by setting acknowledgment bit (HACK).
Method 2: Host responds to IRQ when ready by reading the status register.

   The drive may now setup the task file as it wishes in order to perform the transfer. Between step 4 and step 9, the drive "owns" the task file; the host may not modify the task file even if busy is cleared.

5) The drive clears the IRQ and optionally sets BUSY. The drive sets up the task file for the transfer.

6) The drive sets DRQ and clears BUSY (if it was set in step 5) when it is ready to transfer data.

7) The host transfers data blocks, polling for BUSY cleared and DRQ set to determine when data may be transmitted. No interrupts will be issued during the data transfer. The data is transmitted in blocks as determined by the current multiple size setting.

8) The drive clears busy and sets Disconnect. At this point the drive is relinquishing control of the task file.

9) Method 1: The host responds to the disconnect bit by negating HACK.
   Method 2: The host responds to the disconnect by reading the status register.

   Upon acknowledging the disconnect, the host regains control of the task file.

10) The drive sees the host acknowledge and clears Disconnect and Data Phase.

11) The drive negates bus ownership.

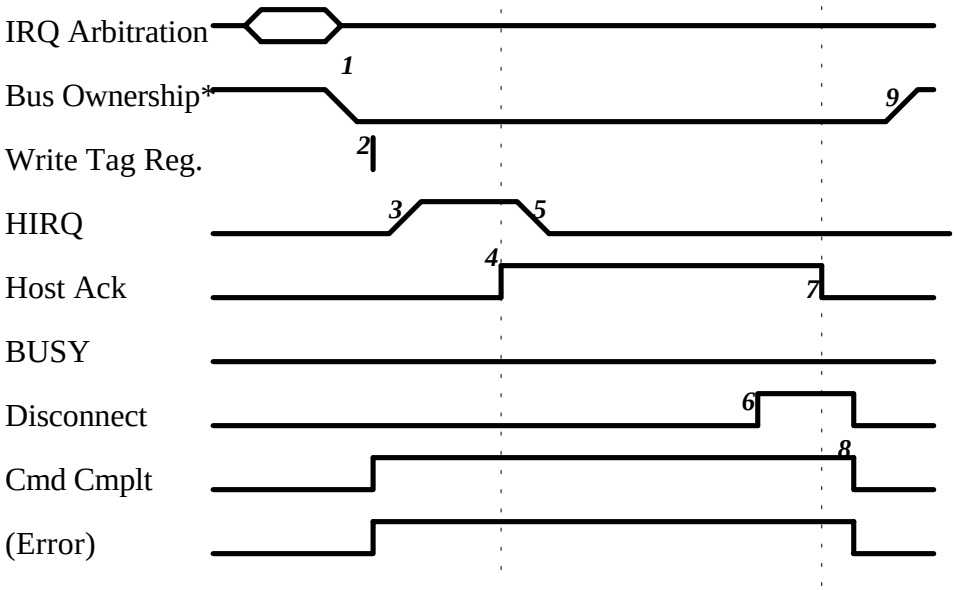
Notes:

- The HIRQ line is controlled directly from the drive's microprocessor rather than by the interface controller. This allows the HIRQ to be asserted regardless of the drive selection bit.
- The drive will assert disconnect when it has run out of data to send to the host, or out of buffer space in which to receive data. This frees the bus until the drive is ready to continue the transfer later.
- The drive asserts disconnect even after the last sector requested. The completion phase is used to signal the end of the command.
- The BUSY bit does not imply task file ownership during the data phase as it does in ATA-1 and ATA-2. Between steps 4 and 9, the task file is "owned" by the selected drive.

Method 2:

- Method 2 of host acknowledgment requires that the host read the status register after the interrupt and after it sees disconnect. At all other times, it must read alternate status.

Completion Protocol



1) Drive arbitrates for bus. Upon winning the bus, drive asserts ownership and holds throughout completion phase.

2) The drive asserts the command complete bit in the status register. If there was an error on the command, the drive sets the Error bit in the status register, and the proper error code in the error bits of the tag register. The drive writes the tag for the command to be serviced.

3) Drive asserts IRQ. The status at this time will be: Command Complete set, and BUSY, DRQ, disconnect, and data phase cleared.

4) Method 1: Host responds to IRQ when ready by setting acknowledgment bit (HACK).
Method 2: Host responds to IRQ when ready by reading the status register.

   Between step 4 and step 7, the drive "owns" the task file; the host may not modify the task file even if busy is cleared.

5) The drive clears the IRQ and optionally sets BUSY.

6) The drive clears busy (if set in step 5) and sets Disconnect. At this point the drive is relinquishing control of the task file.

7) Method 1: The host responds to the disconnect bit by negating HACK.
Method 2: The host responds to the disconnect by reading the status register.

   Upon acknowledging the disconnect, the host regains control of the task file.

8) The drive sees the host's acknowledgement of disconnect, and clears Disconnect and Command Complete.

9) The drive negates bus ownership.

Notes:

- If an error occurs, the location of the error is not reported. A possible mechanism to report the error location would be to update the task file after the host acknowledges the interrupt, and before setting disconnect. The host could wait for disconnect, read the task file for the error location, and then acknowledge the disconnect.