WESTERN DIGITAL

Revision 0.3
July 21, 1998
Status: Proposal

Preliminary Draft for Review Only

ATA / ATAPI - Multi Threading

Point of Contact: **Tom Hanan**
Western Digital Corporation
8105 Irvine Center Drive
Irvine Calif. 92718
Ph: (714) 932-7472
Fx: (714) 932-7314
E-Mail hanan_t@a1.wdc.com

Reflector: **ata@dt.wdc.com**

1.0 Overview

This proposal is intended to provide a basis for extensions to the existing ATA and ATAPI protocols needed to support Multi-threading.

The need to provide Multi-threading extensions to ATA and ATAPI devices is driven by system manufacturers' desire to allow inexpensive, high performance hard disk drives, ATAPI CD-ROM and ATAPI tape devices to share a single low cost ATA connection.

This proposal has been put forward by Western Digital in an effort to address industry wide compatibility issues related to the specifics of implementing high performance Multi threading on ATA host systems and peripherals.

It is the intent of the editor of this document to publish this document as an SFF extension to the existing ATA-2 and SFF ATAPI 1.2 documents.

It is also the intent of the editor to work closely with the editors of the ATAPI protocol and ATA-3 documents to incorporate the appropriate sections of this proposal within their respective documents.

Comments or suggestions regarding the technical content of this proposal should be posted on the ATA / SFF ATAPI reflectors.

Comments or suggestions regarding editorial or administrative issues related to this document should be sent to:

Tom Hanan
Western Digital
8105 Irvine Center Drive
Irvine CA 92718

PH: 714 932-7472
FAX: 714 932-7314
E-Mail hanan_t@a1.wdc.com

P.S. I hope I get more e-mail on the technical content of the document than either the format or the politics of which sections will be included in which X3T10 document. Let's get the compatibility issues worked out before we decide where the information should go......

Regards,
Tom Hanan

I) Traditional

```
           Cmd       Data      Data          Cmd       Data
          +--+--+   +--+--+   +--+--+--+    +--+--+   +--+--+--+
          |SL|IC|o o|RS|TD|o o|RS|TD|RL|    |SL|IC|o o|RS|TD|RL|
          +--+--+   +--+--+   +--+--+--+    +--+--+   +--+--+--+
          |B0|R1|B1 |B0|In|B1 |B0|In|B0     |B0|R1|B1 |B0|In|B0
          |s0|  |   |D1|  |   |D1|  |D0     |s0|  |   |D1|  |D0
          |  |  |   |Ia|  |   |Ia|  |R1     |  |  |   |Ia|  |R1
```

II) Overlapped

```
 Cmd                                    Poll                     Poll
+--+--+--+                             +--+--+                  +--+--+
|SL|IC|RL| o o o o o o o o o o o o o o|SL|RL|o o o o o o o o o |SL|RL|
+--+--+--+                             +--+--+                  +--+--+
|B0|R1|B0                              |B0|B0                   |B0|B0|
|s1|  |D0                              |s1|D0                   |s1|D0|
|  |  |C0                              |  |C0                   |R1|C1|
```

==

New

III) Shared IRQ (ATA-3)

```
 Cmd                           Data                Data
+--+--+--+                    +--+--+--+          +--+--+--+
|SL|IC|RL| o o o o o o o o o|SL|TD|RL| o o o o|SL|TD|RL|
+--+--+--+                    +--+--+--+          +--+--+--+
|B0|R1|B0                     |B0|In|B0           |B0|In|B0
|s0|  |D0                     |s0|  |D0           |s0|  |D0
|  |  |R0                     |D1|  |R0           |D1|  |R1
      |C0                     |A1|  |C0           |A1|  |C1
```

IV) Tagged Queing (ATAPI Only)

```
          Cmd(1)    Cmd(2)           Data(2)            Data(1)
         +--+--+   +--+--+--+       +--+--+--+         +--+--+--+
         |SL|IC|o o|SL|IC|RL| o o o o|SL|TD|RL| o o o o |SL|TD|RL|
         +--+--+   +--+--+--+       +--+--+--+         +--+--+--+
         |B0|R1|B0 |B0|R1|B0        |B0|In|B0          |B0|In|B0
         |s1|  |D0 |s1|  |D0        |s1|A0|D0          |s1|A0|D0
         |t1|  |R1 |t2|  |R0        |D1|  |R0          |D1|  |R0
         |tt|  |C0 |tt|  |C0        |T2|  |C1          |T1|  |C1
                                    |A1|  |            |A1|  |
```

==|

B=BSY, D=DRQ, s=DRV, R=DRDY, C=DSC or POST, A=ATTN(IDX,Status:bit1), Ia=HIRQ Asserted, In=HIRQ Negated, t(n) or T(N)=Tag, tt or TT= Tag Type, ST=Set Taskfile, SL=Select, RL=Release, IC= Issue Command, RS= Read Status, TD=Transfer Data.

BSY	DRDY	DRQ	New Cmd	Change DRV
0	0	0	Ignore New Cmd	OK
0	1	0	Accept New Cmd	OK
0	1	1	Abort Cur Cmd Accept New Cmd	?????????????

```
+-------------------++---------------+----------------+
```

2.0 Shared IRQ:

The shared IRQ protocol is designed to allow more than one device on a given IDE cable to process commands and data concurrently. To achieve this goal the function of several bits within the task file have been enhanced. Specifically the function and behavior of DRDY, DSC and IDX have been changed as follows:

2.1 Shared IRQ Signal Timing:

```
          +     IC        + RL RA ID+   SL  +    TD     +IC or RL
================================================================
         b________________      h   +                     __
DRV    __/+               +XXXXX_________________________/  XXXX
       ___________________         ____________________________
nIEN   _/a               e\_______/j      +            +\______
          +__ c           +         +__k
cmd    ___/A0\______________________/A2\________________________
          +   __       ___+         +   ___m        p___q
BSY    ______/  \_____/  d\____________/   \________/   \_______
          +       ____     _________________n_______o   +
(DRQ)  __________/    \____XXXXX            XXXXXXXX________XXXX
          +               +         +      +n           +
Data   ___________/\/\______________________/\/\/\/\____________
          +               +        i+      +            +
Status __#__________________#_____#_______________________#_____
           ________________         +      n________p   + ______
T File ___/                \_______________/        \____/______
          +               +    ____j       +            +
INTRQ  _______________________/g \_\____________________________
          +               +   __________l  +            +
(ATTN) ______________________/f     +   \________________XXXXXXX
         _______________________________________________________
(DRDY) XX    \_____________XXXXXXXXXXXXXXXXX_________XXXXXXXXXXX
```

Signals enclosed in () brackets are valid any time BSY=0.

a-e) Issue Cmd to Device 1
f-j) Release Cable
k-l) Select Device 0
n-q) Transfer Device 0 Data

2.1.1 a-d) IC - Issue Command

a) The IC-Issue Command phase of a transfer is started by selecting the device to which the command will be sent. If the previous phase was an IC to the same device the host may not need to Release and Select the device again in order to issue it another tagged queue command. Device 1 is used in the example above.

b) For backward compatibility the host must disable the assertion of interrupts using nIEN after selecting the device to which it intends to issue a command. nIEN may already be disabling the interrupts if the previous phase was also an IC to the same device. Writing to nIEN with any device selected shall prevent all devices on the cable from asserting interrupts to the host. Disabling interrupts is necessary to prevent interrupts from non selected device from confusing host drivers while they are issuing commands. Note that disabling interrupts while issuing ATA and ATAPI commands requires ATAPI devices supporting shared interrupts to have hardware accelerated command packet DRQ capabilities.

If the host intends to issue an ATAPI Tagged command to the device then it must set up the tt-Tag Type and Tn-Tag number in the task file, before issuing the A0 ATAPI Packet command to the device.

c) After the host has disabled interrupts it can issue a shared IRQ command to the selected peripheral. Once the command has been issued it will continue normally until BSY and DRQ are de-asserted.

d) When BSY and DRQ are both de-asserted the selected device is signaling the host system that it is releasing control of the cable.

2.1.2 e) RL - Release Cable
If the host system has no other commands for that device or the devices DRDY bit indicates that the device is not ready for another command, the host should poll the other devices on the cable before enabling interrupts using nIEN. This is necessary to detect other devices capable of accepting commands or devices which have pending interrupts to the host which are disabled because of nIEN. If no other devices on the cable require servicing then the host system can enable host interrupts using nIEN and wait for an interrupt. The host system may also wish to launch a timer thread to allow it to poll the devices on the cable for another opportunity to issue more commands to the devices before the next interrupt from a device on the cable.

Note that the cable's task file becomes invalid from the time that the host uses nIEN to disable host interrupts until nIEN is used to re-enable host interrupts or the device is selected using the ATA-3 A2 Select Device command. (See j-n)

2.1.3 f-g) RA - Request Attention
While the cable is Released, I.E. the host is not issuing a command , interrupts are enabled, and no device has been selected to process an interrupt, any device on the cable may assert ATTN:IDX and assert the interrupt signal to the host to request attention from the host. Note that the assertion of ATTN in the status register is independent of nIEN and must precede the assertion of the interrupt signal.

2.1.4 i-j) ID - Identify Device requesting Attention
Upon receipt of an interrupt the host system must use the DRV bit to poll the devices in the cable to determine which device is requesting attention.

Depending on the IRQ mode selected using the Set Features command the interrupt signal asserted by the device may be either a 1-2us pulse or a signal negated by a read to the device's status register.

2.1.5 j-n) SL - Selection
Devices configured for Shared IRQ must be selected to enable their task file for interrupt processing. This is true for command termination and data transfer interrupts. In both cases the device will assert ATTN until they are selected by the host using the ATA-3 A2 Device Select command.

Once the host has determined which device requires attention the host must disable host interrupts from the devices on the cable using nIEN before issuing the ATA-3 Select Device command.

2.1.6 n-q) TD - Transfer Data

Data transfer occurs as defined by the protocol for the non A2 command issued to the device. The Transfer data phase is complete when the device again releases the cable by de-asserting BSY and DRQ.

2.2 New ATA-3 Command Definitions:

Several new commands have been defined to support Shared IRQ.

2.2.1 Select Device (A2h)

This command allows the host to assign control of the cable to a specific device. The device is considered to have been selected only if, after the command is issued to the device, the device de-asserts BSY without aborting the A2 command. Devices aborting the A2 command do not support the shared IRQ protocol and may need to be reset before they will again function according to the traditional IDE protocol.

One issue yet to be resolved is how long the host system should be forced to wait for BSY to be de-asserted when the host issues an A2-Select Device command. Because nIEN has been set to disable interrupts an interrupt can not be generated by the device to indicate that it is done processing the Select Device command. Since it is assumed that initial shared IRQ devices will not hardware accelerate the Select Device command, the host system must poll for the de-assertion of BSY.

To prevent degradation of host performance it is recommended that the host system poll for 50us after which it should generate a timer thread to return control to the driver not more than 500us later. The host should continue to use a timer driven polling sequence for up to 3ms at which point the host should reset the devices on the cable assuming a catastrophic command failure has occurred.

2.2.2 Set Features:

Enable Shared IRQ Protocol:
Disable Shared IRQ Protocol:
These two commands are used to enable and disable the shared IRQ protocol within the devices on the cable. These commands are designed to allow the OS drivers to identify devices compatible with the shared IRQ protocol and configure them for maximum performance. The device will abort or accept these commands to identify its compatibility with the feature.

Peripherals shipped with Shared IRQ protocol enabled may lock up BIOS and OS drivers which are not designed to utilize this feature when they are unable to abort an active command because DRDY remains = 0. BIOS and drivers which issue an SRST as a result of a DRDY time-out are 100% compatible with drives shipped with Shared IRQ enabled.

Enable OCI-Open Collector IRQ Protocol:
Disable OCI-Open Collector IRQ Protocol:
These two commands are used to enable and disable the open collector IRQ protocol within the devices on the cable. These commands are designed to allow the BIOS or adapter drivers to identify devices compatible with the open collector protocol and configure the host and peripherals for maximum performance. The device will abort or accept these commands to identify its compatibility with the feature.

2.3 New Bit & Signal Definitions:

2.3.1 DRDY:

DRDY has been changed so that it can be used to indicate when the peripheral is ready to accept another command into its command Queue. To achieve this functionality the peripheral no longer aborts the current command when it receives a new command with DRDY=1. When the peripherals queue is full, it must indicate this condition to the host system by de-asserting DRDY. Commands received while DRDY=0 shall be ignored by the peripheral. Note that de-asserting DRDY may prevent the host from aborting a command by issuing another command before the first command has completed.

2.3.2 DSC (POST):

DSC has been changed so that it can be used to indicate when the peripheral has completed the command for which it is reporting status. In the case of Tagged ATAPI commands the DSC bit is used in conjunction with the tag to identify which command in the queue has completed.

2.3.3 IDX (ATTN):

IDX has been changed so that it can be used to indicate which peripherals are asserting HIRQ. This bit mimics the peripheral's internal IRQ bit and is unaffected by the state of nIEN. Note that this bit may also be used to poll the peripherals without the use of interrupts.

3.0 Tagged Queuing (ATAPI Only):

The tagged queuing protocol for ATAPI is designed to allow the peripheral to process multiple commands concurrently. To achieve this goal the function of several TAG fields within the ATAPI task file have been defined. Specifically the function and behavior of Tag and Tag Type have been defined as follows:

If only SIMPLE QUEUE TAG commands are used, the peripheral may execute the commands in any order that is deemed desirable.

If ORDERED QUEUE TAG commands are used, the peripheral shall execute the commands in the order received with respect to other commands received with ORDERED QUEUE TAGs. All commands received with a SIMPLE QUEUE TAG Type prior to a command received with an ORDERED QUEUE TAG type, shall be executed before that command with the ORDERED QUEUE TAG type.

All commands received with a SIMPLE QUEUE TAG Type after a command received with an ORDERED QUEUE TAG Type, regardless of initiator, shall be executed after that command with the ORDERED QUEUE TAG type.

A command received with a HEAD OF QUEUE TAG type is placed first in the queue, to be executed next. A command received with a HEAD OF QUEUE TAG message shall be executed prior to any queued I/O process. Consecutive commands received with HEAD OF QUEUE TAG type are executed in a last-in-first-out order.

A command received without a queue tag type, while there are any tagged I/O commands in the command queue, shall be aborted with a Tag of 00h.

An ATAPI peripheral that supports shared IRQ, but not tagged queuing, shall use DRDY to accept one tagged command at a time. All peripherals supporting Shared IRQ shall return the tag issued to the peripheral with the command.

3.1 Typical sequences for tagged queuing:

An ATAPI command using tagged queuing uses the following sequences for normal execution. The host system selects the device and monitors DRDY until the device is ready to accept another command (DRDY=1). The host then sets up the task file for the command including the Tag and Tag Type fields. Note the Tag and Tag Type fields are specified as 0 for non tagged ATAPI commands. Once the Task file is configured then the host reads the status register again to verify that the peripheral is ready to accept a command. If DRDY=1 then the host writes to the command register to issue the tagged command. The host is then free to issue more queued commands to this peripheral or select another device on the cable.

3.2 Aborting Tagged Commands:

The host may abort a specific tagged command in the peripheral's queue by issuing a new command to the peripheral with the tag used for the command to be aborted. The new command is placed into the queue using the order defined by the new tag type. Note that the tag type is not required to match the original tag.

Peripherals can abort tagged commands using the existing aborted command sequence with the additional requirement that the peripheral identify the aborted command with the command's tag.

```
          Cmd(1)    Cmd(2)            Data(2)              Data(2)
         +--+--+   +--+--+--+        +--+--+--+         +--+--+--+
         |ST|IC|o o|ST|IC|RL| o o o o|SL|TD|RL| o o o o |SL|TD|RL|
         +--+--+   +--+--+--+        +--+--+--+         +--+--+--+
         |B0|R1|B0 |B0|R1|B0         |B0|In|B0          |B0|In|B0
         |s1|  |D0 |s1|  |D0         |s1|A0|D0          |s1|A0|D0
         |t1|  |R1 |t2|  |R0         |D1|  |R0          |D1|  |R0
         |tt|  |C0 |tt|  |C0         |A1|  |C0          |A1|  |C1
                                     |T2|  |            |T2|

                           Abort(1)
                           +--+--+
                           |SL|RL|
                           +--+--+
                           |B0|B0
                           |s1|D0
                           |D0|R0
                           |T1|C1
                           |E1
```

3.3 Example of Tag Queue Ordering:

An example of the execution of five queued Commands is given below to demonstrate how tagged queuing operates. The five commands are defined in the table below. At the time the Commands are first being executed, it is assumed that the actuator is in position to access logical block 10 000.

Commands in order received by peripheral

Command	Queue tag message	Queue tag value	Logical block address	Transfer length	Status
READ	SIMPLE	01h	10 000	1 000	Queued
READ	SIMPLE	02h	100	1	Queued
READ	ORDERED	03h	1 000	1 000	Queued
READ	SIMPLE	04h	10 000	1	Queued
READ	SIMPLE	05h	2 000	1 000	Queued

The optimum order would require that those blocks close to the actuator position be the first blocks accessed, followed by those increasingly far from the actuator position. However, the command with queue tag 03h is an ordered command, so that all simple Commands transferred previously must be executed before, while all simple Commands transferred after the ordered command must be executed after the ordered command.

If a peripheral supports an optimizing algorithm the actual order in which the Commands are executed could be as shown in the table below.

Commands in order of execution

Command	Queue tag message	Queue tag value	Logical block address	Transfer length	Status
READ	SIMPLE	01h	10 000	1 000	Queued
READ	SIMPLE	02h	100	1	Queued
READ	ORDERED	03h	1 000	1 000	Queued
READ	SIMPLE	05h	2 000	1 000	Queued
READ	SIMPLE	04h	10 000	1	Queued

Commands with queue tag values 01h and 02h are executed in the order received since the actuator is already in position to execute command 01h. Command 02h must be executed before Command 04h or 05h because the ordered Command 03h was transmitted after Commands 01h and 02h but before Commands 04h and 05h. Command 03h is then executed after Command 02h. The Commands 04h and 05h are executed after the ordered Command 03h. Command 05h is executed before Command 04h because the actuator is in position to access block 2 000 after executing Command 03h. Command 04h is executed last.

As an example of the operation of the HEAD OF QUEUE TAG Command, consider that a new Command, identified by a HEAD OF QUEUE TAG message with a queue tag of 08h, is transmitted to the peripheral while the ordered Command 03h is being executed. The Command 03h continues execution, but the new HEAD OF QUEUE TAG Command is placed in the queue for execution before all subsequent Commands. In this case, the queue for execution after the ordered Command 03h was executed would appear as shown in the table below.

Modified by HEAD OF QUEUE TAG message

Command	Queue tag message	Queue tag value	Logical block address	Transfer length	Status
READ	ORDERED	03h	1 000	1 000	Executing
READ	HEAD OF QUE	08h	0	8	Queued
READ	SIMPLE	05h	2 000	1 000	Queued
READ	SIMPLE	04h	10 000	1	Queued

To obtain maximum performance gains using tagged queuing requires careful implementation of the queuing algorithms in the peripheral. In addition, host systems should allow a maximum number of simple Commands to be executed with a minimum number of ordered Commands.