Programming Interface

for

Bus Master IDE Controller

Revision 1.0

5/16/94

Until this specification is ratified, it is solely
owned and maintained by:
Brad Hosler, Intel Corporation
bwh@salem.intel.com (please comment using email)
503-696-8431

1.0. IDE Bus Master

This document defines a register level programming interface for a bus master ATA compatible (IDE) disk controller that directly moves data between IDE devices and main memory. By performing the IDE data transfer as a bus master, the Bus Master Device offloads the CPU (no programmed IO for data transfer) and improves system performance in multitasking environments.

Controllers that implement this programming interface will benefit from bundled software shipped with major OS's limiting the amount of software development required to provide a complete product.

The master mode programming interface is an extension of the standard IDE programming model. This means that devices can always be dealt with using the standard IDE programming model, with the master mode functionality used when the appropriate driver and devices are present. Master operation is designed to work with any IDE device that support DMA transfers on the IDE bus. Devices that only work in PIO mode can be used through the standard IDE programming model.

The programming interface defines a simple scatter/gather mechanism allowing large transfer blocks to be scattered to or gathered from memory. This cuts down on the number of interrupts to and interactions with the CPU.

The interface defined here supports two IDE channels (primary and secondary). Individual controllers that support more than two channels will need to appear to software as multiple controllers if the standard drivers are to be used.

Master IDE controllers should default to Mode 0 Multiword DMA timings to ensure operation with DMA capable IDE devices without the need for controller-specific code to initialize controller-specific timing parameters.

1.1. Physical Region Descriptor Table

Before the controller starts a master transfer it is given a pointer to a Physical Region Descriptor Table. This table contains some number of Physical Region Descriptors (PRD) which describe areas of memory that are involved in the data transfer. The descriptor table must be aligned on a 4 byte boundary and the table cannot cross a 64K boundary in memory.

1.2. Physical Region Descriptor

The physical memory region to be transferred is described by a Physical Region Descriptor (PRD). The data transfer will proceed until all regions described by the PRDs in the table have been transferred.

Each Physical Region Descriptor entry is 8 bytes in length. The first 4 bytes specify the byte address of a physical memory region. The next two bytes specify the count of the region in bytes (64K byte limit per region). A value of zero in these two bytes indicates 64K. Bit 7 of the last byte indicates the end of the table; bus master operation terminates when the last descriptor has been retired.

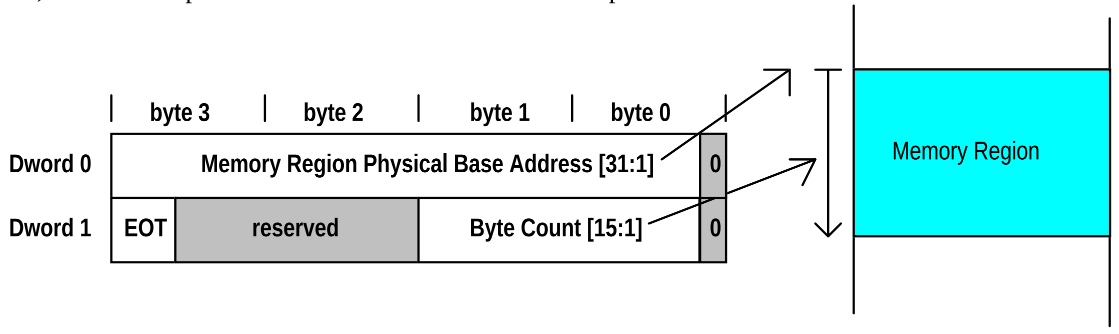

Figure 1: Physical Region Descriptor Table Entry

Note

The memory region specified by the descriptor is *further* restricted such that the region cannot straddle a 64K boundary. This means that the byte count can be limited to 64K, and the incrementer for the current address register need only extend from bit [1] to bit [15]. Also, the total sum of the descriptor byte counts must be equal to, or greater than the size of the disk transfer request. If greater than, then the driver must terminate the Bus Master transaction (by resetting bit zero of the command register to zero) when the drive issues an interrupt to signal transfer completion.

2.0. Bus Master IDE Register Description

The bus master IDE function uses 16 bytes of IO space. All bus master IDE IO space registers can be accessed as byte, word, or Dword quantities. The description of the 16 bytes of IO registers follows:

Offset from Base Address	Register	Register Access
00h	Bus Master IDE Command register Primary	R/W
01h	Device Specific	
02h	Bus Master IDE Status register Primary	RWC
03h	Device Specific	
04h-07h	Bus Master IDE PRD Table Address Primary	R/W
08h	Bus Master IDE Command register Secondary	R/W
09h	Device Specific	
0Ah	Bus Master IDE Status register Secondary	RWC
0Bh	Device Specific	
0Ch-0Fh	Bus Master IDE PRD Table Address Secondary	R/W

2.1. Bus Master IDE Command Register

Register Name:	Bus Master IDE Command Register	
Address Offset:	Primary Channel:	Base + 00h
	Secondary Channel:	Base + 08h
Default Value:	00h	
Attribute:	Read / Write	
Size:	8 bits	

Bit	Description
7:4	**Reserved.** Must return 0 on reads.
3	**Read or Write Control:** This bit sets the direction of the bus master transfer: when set to zero, PCI bus master reads are performed. When set to one, PCI bus master writes are performed. This bit must NOT be changed when the bus master function is active.
2:1	**Reserved.** Must return 0 on reads.
0	**Start/Stop Bus Master:** Writing a '1' to this bit enables bus master operation of the controller. Bus master operation begins when this bit is detected changing from a zero to a one. The controller will transfer data between the IDE device and memory only when this bit is set. Master operation can be halted by writing a '0' to this bit. All state information is lost when a '0' is written; Master mode operation cannot be stopped and then resumed. If this bit is reset while bus master operation is still active (i.e., the Bus Master IDE Active bit of the Bus Master IDE Status register for that IDE channel is set) and the drive has not yet finished its data transfer (The Interupt bit in the Bus Master IDE Status register for that IDE channel is not set), the bus master command is said to be aborted and data transfered from the drive may be discarded before being written to system memory. This bit is intended to be reset after the data transfer is completed, as indicated by either the Bus Master IDE Active bit or the Interrupt bit of the Bus Master IDE Status register for that IDE channel being set, or both.

2.2. Bus Master IDE Status Register

Register Name:	Bus Master IDE Status Register	
Address Offset:	Primary Channel:	Base + 02h
	Secondary Channel:	Base + 0Ah
Default Value:	00h	
Attribute:	Read/Write Clear	
Size:	8 bits	

Bus Master IDE Status Register

Bit	Description
7	**Simplex only:** This read-only bit indicates whether or not both bus master channels (primary and secondary) can be operated at the same time. If the bit is a '0', then the channels operate independently and can be used at the same time. If the bit is a '1', then only one channel may be used at a time.
6	**Drive 1 DMA Capable:** This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 1 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance.
5	**Drive 0 DMA Capable:** This read/write bit is set by device dependent code (BIOS or device driver) to indicate that drive 0 for this channel is capable of DMA transfers, and that the controller has been initialized for optimum performance.
4:3	**Reserved.** Must return 0 on reads.
2	**Interrupt:** This bit is set by the rising edge of the IDE interrupt line. This bit is cleared when a '1' is written to it by software. Software can use this bit to determine if an IDE device has asserted its interrupt line. When this bit is read as a one, all data transfered from the drive is visible in system memory.
1	**Error:** This bit is set when the controller encounters an error in transferring data to/from memory. The exact error condition is bus specific and can be determined in a bus specific manner. This bit is cleared when a '1' is written to it by software.
0	**Bus Master IDE Active:** This bit is set when the Start bit is written to the Command register. This bit is cleared when the last transfer for a region is performed, where EOT for that region is set in the region descriptor. It is also cleared when the Start bit is cleared in the Command register. When this bit is read as a zero, all data transfered from the drive during the previous bus master command is visible in system memory, unless the bus master command was aborted.

2.3. Descriptor Table Pointer Register

Register Name:	Descriptor Table Pointer Register	
Address Offset:	Primary Channel:	Base + 04h
	Secondary Channel:	Base + 0Ch
Default Value:	00000000h	
Attribute:	Read / Write	
Size:	32 bits	

Descriptor Table Pointer Register

Bit	Description
31:2	Base address of Descriptor table. Corresponds to A[31:2]
1:0	reserved

The Descriptor Table must be Dword aligned. The Descriptor Table must not cross a 64K boundary in memory.

3.0. Operation

3.1. Standard Programming Sequence

To initiate a bus master transfer between memory and an IDE DMA slave device, the following steps are required:

1) Software prepares a PRD Table in system memory. Each PRD is 8 bytes long and consists of an address pointer to the starting address and the transfer count of the memory buffer to be transferred. In any given PRD Table, two consecutive PRDs are offset by 8-bytes and are aligned on a 4-byte boundary.
2) Software provides the starting address of the PRD Table by loading the PRD Table Pointer Register . The direction of the data transfer is specified by setting the Read/Write Control bit. Clear the Interrupt bit and Error bit in the Status register.
3) Software issues the appropriate DMA transfer command to the disk device.
4) Engage the bus master function by writing a '1' to the Start bit in the Bus Master IDE Command Register for the appropriate channel.
5) The controller transfers data to/from memory responding to DMA requests from the IDE device.
6) At the end of the transfer the IDE device signals an interrupt.
7) In response to the interrupt, software resets the Start/Stop bit in the command register. It then reads the controller status and then the drive status to determine if the transfer completed successfully.

3.1. Data Synchronization

When reading data from an IDE device, that data may be buffered by the IDE controller before using a master operation to move the data to memory. The IDE device driver in conjunction with the IDE controller is responsible for guaranteeing that any buffered data is moved into memory before the data is used.

The IDE device driver is required to do a read of the controller Status register after receiving the IDE interrupt. If the Status register returns with the Interrupt bit set then the driver knows that the IDE device generated the interrupt (important for shared interrupts) and that any buffered data has been flushed to memory. If the Interrupt bit is not set then the IDE device did not generate the interrupt and the state of the data buffers is unknown.

When the IDE controller detects a rising edge on the IDE device interrupt line (INTRQ) it is required to:

- Flush all buffered data
- Set the Interrupt bit in the controller Status register
- Guarantee that a read to the controller Status register does not complete until all buffered data has been written to memory.

Another way to view this requirement is that the first read to the controller Status register in response to the IDE device interrupt must return with the Interrupt bit set and with the guarantee that all buffered data has been written to memory.

3.1. Status Bit Interpretation

The table below gives a description of how to interpret the Interrupt and Active bits in the Controller status register after a DMA transfer has been started.

Interrupt	Active	Description:
0	1	DMA transfer is in progress. No interrupt has been generated by the IDE device.
1	0	The IDE device generated an interrupt. The controller exhausted the Physical Region Descriptors. This is the normal completion case where the size of the physical memory regions was equal to the IDE device transfer size.
1	1	The IDE device generated an interrupt. The controller has not reached the end of the physical memory regions. This is a valid completion case where the size of the physical memory regions was larger than the IDE device transfer size.
0	0	This bit combination signals an error condition. If the Error bit in the status register is set, then the controller has some problem transferring data to/from memory. Specifics of the error have to be determined using bus-specific information. If the Error bit is not set, then the PRD's specified a smaller size than the IDE transfer size.

4.0. Error Conditions

IDE devices are sector based mass storage devices. The drivers handle errors on a sector by sector basis; either a sector is transferred successfully or it is not.

If the IDE DMA slave device never completes the transfer due to a hardware or software error, the Bus Master IDE command will eventually be stopped (by setting Command Start bit to zero) when the driver times out the disk transaction. Information in the IDE device registers will help isolate the cause of the problem.

If the controller encounters an error while doing the bus master transfers it will stop the transfer (ie. reset the Active bit in the Command register) and set the ERROR bit in the Status register. The controller does not generate an interrupt when this happens. The device driver can use device specific information (e.g.; PCI Configuration Space Status register) to determine what caused the error.

Whenever a requested transfer does not complete properly, information in the IDE device registers (Sector Count) can be used to determine how much of the transfer was completed and to construct a new PRD table to complete the requested operation. In most cases the existing PRD table can be used to complete the operation.

5.0. PCI Specifics

Bus master IDE controllers built to attach to a PCI bus must have the following characteristics:

1) The Class Code in PCI configuration space indicates IDE device (top two bytes have the value 0x0101) and bit 7 of the Programming Interface register (offset 0x09) in PCI configuration space must be set to 1 to indicate that the device supports the Master IDE capability.
2) The control registers for the controller are allocated via the devices Base Address register at offset 0x20 in PCI configuration space.
3) In the controller Status register the Error bit will be set and the Active bit reset if any of the following conditions occur on the PCI bus while the controller is doing a master operation on the bus. The exact cause can be determined by examining the Configuration Space Status register.

Error Condition	Configuration Space Status bits
Target Abort	Anytime bit 12 of the Config Space Status register is set.
Master Abort	Anytime bit 13 of the Config Space Status register is set.
Data Parity Error Detected	Anytime bit 8 of the Config Space Status register is set.