DOCUMENT SOURCE REFERENCE	
Hardware	
Software	

REV.	ECO	DESCRIPTION	APPR.	DATE
		UPDATE STATUS CODES		3/1/88
		MISCELLANEOUS CORRECTIONS (PAGE 32) - UNRELEASED		2/8/88
		ADDED READ DEFECTS AND LOAD PARAMETER COMMANDS - UNRELEASED		1/27/88
		CHANGED EXPRESS COMMANDS AND DIAGNOSE COMMAND - UNRELEASED		1/13/88
		ADDED READ PARAMETERS -- STILL UNRELEASED		12/23/87
		CHANGED CDB FORMAT - STILL UNRELEASED		12/7/87
		UNRELEASED PRELIMINARY VERSION		11/11/87

DATED SIGNATURES AS APPROPRIATE FOR DOCUMENT

ORIGINATOR		CHECKER	
DSGN. ENGR.		COMP. ENGR.	
MATL. ENGR.		REL. ENGR.	
PROD. ENGR.			

WESTERN DIGITAL CORPORATION

TITLE
WD7000V-ASE ESDI SECTION

DOCUMENT NUMBER
UNRELEASED

REVISION

SHT OF

Table of Contents

List of Tables

WESTERN DIGITAL CORP.

Engineering Spec. Highlights

7000-ASE ESDI Controller Section

Project Venus

April 14, 1988

1. PURPOSE

The purpose of this document is to provide the customer with enough information to begin writing a device driver, for the ESDI portion of the 7000V-ASE controller. The driver is expected to be able to initialize the controller, send commands, receive responses, format the drive, read data, and write data. This is not an exhaustive, nor even a comprehensive engineering specification. It is highly focused in its approach. It concentrates primarily on the communications layer between the host and the controller, and on a small subset of commands. The subset of commands chosen were selected on the basis of their importance to the data path and the communications layer. Furthermore, only those aspects of the commands chosen that were considered to be most essential are described.

2. REFERENCE DOCUMENTS

	Description/Title	Document Num./Source
1.	WD7000-ASC	96-000494
2.	WD42c22(Vanilla)	96-00422
3.	ESDI	ANSI X3R9.3/86

This information is confidential and proprietary to WDC and shall not be reproduced or further disclosed to anyone other than wdc employees, without written authorization from Western Digital Corporation.

This document avoids duplicating information that exists elsewhere. Rather it compares and contrasts itself with other documents. With respect to the communications layer you can generally accept what is written in the WD7000-ASC engineering specification. When this spec and the WD7000-ASC spec disagree, this document takes precedence. The actual contents of the command and response packets are to be taken entirely from this document. None of the commands or responses that have associated CDB's are the same for the ESDI portion of Venus and the WD7000-ASC. With respect to the data path, what is written in the Vanilla spec WD42C22 (96 004222), the ESDI spec ANSI X3T9.3/86, and the IBM-PC/AT technical reference manual are to be used, except when they disagree with this document. With respect to specific hardware addresses, this information is still forthcoming.

3. General Description

3.1. WD7000V-ASE features

High Performance ESDI Disk Controller

IBM PC/AT Host Interface Supporting First Party DMA

Even or Odd DMA Starting Address

Programmable DMA Bus On/Off Time

High Performance Buffer Manager (Data Bandwidth=6MB/second)

32KB Data Buffer

2 ESDI Drives Supported

Hard Sector ESDI Drives

Data Rates 10 MBit and 15 MBit

Programmable Interleave 1:1 and greater

56 Bit ECC

Media Defects to be handled by the host

Self Diagnostic

User Selectable Device Address, DMA, and Interrupt Channel

5. HOST INTERFACE PROTOCOL

For the most part everything in the corresponding WD7000-ASC spec for this section is also true for the ESDI portion of the venus controller. Of course, all references to the SCSI interface have no relevance.

5.1. General Signal Flow

All communications between the host and the controller take place via the HIL(see section 4.5 of WD7000-ASC spec for the definition of the HIL), using a command control byte and a mail box system. The host issues commands to the controller by first selecting the controller through its address bus and writing a control byte to the command port. The command strobe clears the COMMAND PORT READY status bit alerting the host and the onboard LCPU that command execution is now in progress. The host must provide additional parameter bytes, if required, by writing to the command port when the COMMAND PORT READY status bit is set within about 50 us. This 8-bit programmed I/O data transfer handshake is repeated until the required number of bytes to complete the command have been transferred. As of this writing only the initialization command requires more than a single byte.

The type of commands that can be put into the command port fall into two very broad categories from a signal flow point of view. These are commands without associated mail boxes and commands with mail boxes. These commands differ in that the ones without mail boxes transfer all command information via the programmed i/o command port, and receive all the status information via the programmed i/o status port. Whereas, the ones with mail boxes require additional DMA activity, both to send the command and to receive the return status. Also, with the exception of the initialization command, all non-mail box commands complete within 50 micro-seconds, whereas mail box commands generally take much longer.

In spite of their differences both types of commands begin in precisely the same way. Namely, by placing a one byte command in the command register. However, in the case of mail box commands this command will always be a start i/o command, (start multiple i/o is not supported by the ESDI portion of the Venus controller). The start I/O command byte contains a mail box number. This mail box number is an index into an array of longwords, in host DMA space. The associated longword contains the address of a command descriptor block (CDB) in its 3 low order bytes. In its high order byte it contains the length in bytes of the (CDB) that it points to. The (CDB) in turn contains the information needed to specify ESDI and controller specific commands. It, also, provides empty fields that are dedicated to receiving the command completion status.

The 4 bytes contained in a mail box are accessed using the onboard DMA controller (first party DMA). The controller uses the information contained in the mail box to DMA the associated CDB into its local RAM. Once the associated CDB is obtained, the host is free to re-use the mail box, but not the CDB. The controller indicates to the host that the mail box can be re-used by clearing the high order byte in the mail box.

The CDB's are DMA'd using the same first party DMA mechanism that was used for the mail boxes. However, unlike the mail boxes and unlike the CDB's in the WD7000-ASC spec, these CDB's are not fixed length. This is due to the variable number of header pointers that the data transfer commands may contain. Therefore, as mentioned earlier the high order byte of the mail box is used to store the length of the corresponding CDB block. This information is used by the controller, both in determining how large of an internal buffer to reserve for the transfer, and in deciding how much to transfer. Once transferred, the CDB is buffered up in the controller's local RAM.

The host must not interpret the fact, that the CDB is buffered up in the controllers local RAM, to mean that it is safe for the host to re-use its own copy of the CDB. The host must not re-use the CDB until the controller has completely finished its processing of the command. This is due to the fact that the response will be delivered in selected fields of the same CDB that specified the original command. The host will be notified of command completion via an ICMB, which will be accompanied by an interrupt. The ICMB will be pointing to the same CDB that the original OGMB was pointing to at the beginning of the request. After receiving the ICMB, the host is free to re-use the CDB.

For any read or write operation to the ESDI device additional bytes of data are DMA'd using first party DMA as needed. The CDB specifies the actual location in host i/o space that serves as a destination or source for a DMA transfer.

5.1.1. Power-Up Sequence

The timings on the power-up sequence have not yet been determined. For now assume that they will be the same as for the WD7000-ASC. Of course, there will be no initialization of the SBIC.

5.2. Host Hardware Interface

The description of the location and layout of the host registers in the WD7000-ASC spec applies equally to the Venus ESDI board. Of course, the actual address of the 4 bytes of register space may very well be in a different part of the 256 byte area allowed for them. In fact, in the final Venus board, there will be two sets of such registers and they are guaranteed to be at different locations. One set will be for the host/SCSI 1st party DMA interface and the other set will be for the host/ESDI 1st party DMA interface. The host/SCSI 1st party DMA interface will of course be identical to the WD7000-ASC host/SCSI 1st party DMA interface.

5.2.1. Status Port

Identical to 5.2.1 in WD7000-ASC spec.

5.2.1.1. Interrupt Image Flag

Identical to 5.2.1.1 in WD7000-ASC spec

5.2.1.2. Command Port Ready

Identical to 5.2.1.2 in WD7000-ASC spec

5.2.1.3. Command Byte Rejected

There are a number of differences in the actual treatment of and the suggested usage of this bit between the Venus/ESDI controller and the WD7000-ASC. For starters in the Venus/ESDI controller, this bit will never indicate that an internal controller queue is full. In fact, it will always indicate either a bad command, a bad parameter, or a bad return status from a non-mail box type command.

There is one exception, which can occur when a start express command in OGMB #n command byte is written to the command register by the host. An express command is a new type of command that exists on the Venus/ESDI controller that did not exist on the WD7000-ASC. It presents a mail box to the controller for non-queued immediate execution. In this way certain mail box type commands can be presented to the controller as an express request. (See table 5 to find out which commands may be sent as express commands.) The controller will execute an express request ahead of all the queued commands using a separate internal buffer that is not a part of the queued commands' buffer pool. It will only execute one express request at a time, and it will not queue up express requests. Any express command sent by the host while the controller is still processing a prior express request will be rejected. The controller will set the byte rejected bit to notify the host of this fact.

In the case of all other mail box type commands, the recommended host usage of this bit is opposite that proposed in the WD7000-ASC spec. For the Venus/ESDI controller, the host should **NOT** check this bit, after writing a mail box type command to the command register. The status and/or error information will always be provided by the controller for mail box type commands via fields in the original CDB that are reserved for this purpose. This information will always be brought to the host's attention via an incoming mail box containing the address of the original CDB in its lower 3 bytes. The host will always receive an interrupt when an incoming mail box is delivered.

In the unlikely event that the incoming mail for some mail box type command is never delivered, the host should use a command time-out and an express inquire command to determine what happened. The inquire command is a special mail box type command whose sole purpose in life is to inquire about the progress of other mail box type commands. It can find any command that is still being processed by the controller, and can determine whether or not the controller is capable of doing work on it. This technique provides us with a high level sanity check on the controller, and a mechanism for dealing with command register transmission errors after the fact.

5.2.1.4. ASC initialized flag

Identical to 5.2.1.4 in WD7000-ASC spec

5.2.2. Interrupt Status Byte

There is only one difference. The SBIC bit is not applicable to the Venus/ESDI controller.

5.2.3. Command Port

The host issues controller control commands via the first write port to get its attention. All control commands are 1 byte instructions except the initialization command which must be sent to the controller after power on or controller reset.

The command port allows the host to queue commands at any time desired. The start multiple i/o command is not supported by the Venus/ESDI controller. There is no limit to the number of host commands that can be queued at one time.

The functionality of the start i/o command can be implicitly achieved by writing a host driver, which treats the OGMBs as a ring and spends as little time as possible dealing with the command register when setting up mail box commands. On the controller end this functionality can be achieved by maximizing the extent to which it can buffer up requests.

The Venus/ESDI controller has a new command that is not supported by the WD7000-ASC controller. This is the express command. The bytecode for the express command is 01NNNNNN, where the 01 in the upper two bits identify the command as the express command and the lower six bits are a mail box number. The express command is a variant of the start i/o command that differs in the way that the controller queues the corresponding CDB. The controller will execute an express command immediately. It does not queue express commands like it does other mail box type commands. Therefore, it can only accept one express command at a time. Any attempt by the host to send more than one express command at a time will result in all commands but the 1st being rejected by the controller. The host is notified of each rejected command via the rejected command bit in the controller status byte.

The controller will always be able to handle one and only one express command regardless of whether or not it has enough buffers to handle queued commands. Only certain mail box type commands can be issued as an express command. Diagnose, Inquire and Abort must always be sent as express commands. Esdi and Read Parameters may be sent as either express or nonexpress commands. However, when they are sent as express commands they can neither be aborted, nor inquired about. The reset of the commands must be sent as non-express commands.

5.2.4. Interrupt Acknowledge

Identical to section 5.2.4 of the WD7000-ASC spec

5.2.5. Host Control Register

The SCSI Port Reset bit is not applicable to the Venus/ESDI controller. Everything else in this section applies equally to both controllers. These statements also apply to all subheadings of this heading.

5.3. Mail Boxes

In order to fetch and execute a CDB setup by the host in it's system memory, the host must provide the I/O channel with the starting address of the block. Likewise, when the controller completes a command, it must indicate to the host the address of the CDB it has just completed. These handshakes are accomplished via Outgoing and Incoming mail box systems. The terms Outgoing/Incoming are with respect to the host system. At system startup the host issues the initialization command which sets the number of outgoing/incoming mail boxes and the starting address of the mail box block. The relative mail box number is given in the command port control opcodes. The format of a typical mail box in memory is shown below:

Mail Type	Contents
O.G.M.B. #n	Size of CDB in bytes, or 0 for empty. Pointer to CDB (MSB) Pointer to CDB Pointer to CDB (LSB)
I.C.M.B. #m	Action Status Pointer to CDB (MSB) Pointer to CDB Pointer to CDB (LSB)

Table 1. Table 5-5: Mail Box Format

OGMB Address = Starting address of mail block + mail box number n x 4

```
ICMB Address = Starting address of mail block + mail box number m x 4
               + 4 (total number of OGMB's)

     where     0 <= n <= p-1  &  0 <= m <= q-1

               p,q = number of OGMBs/ICMBs resp.   (max. = 64)
               m,n = mail box offset numbers from 0 to 63 (max.)
```

5.3.1. Outgoing Mail

An outgoing mail box is marked as full or empty by the mail box size-of-CDB byte. A value of zero means that the controller has picked up any mail that may have been in the box. This means that the controller has DMA'd the longword contents of the mail box into its local RAM and has retrieved the associated CDB. It tells the host that it can re-use the mail box, but it does not indicate that the associated CDB can be re-used. The host must wait until the entire command has completed before re-using the associated CDB. This is due to the fact that the response is written back to the host via selected fields of the original CDB in host memory. A value other than zero in the size-of-CDB byte indicates three things. First it tells the host that the mail has not yet been picked up. Second, it tells the controller that there is mail to be picked up. Third, it tells the controller how large the associated CDB is.

The host has set up the OGMB CDB pointer first and then set the size-of-CDB byte. The host then commands the controller to check a specific mail box by writing its number into the command register. The host does this on a per mail box basis. The scan mail box byte command is not supported by the Venus/ESDI controller. The mail boxes are emptied by the controller on a first come first serve basis. However, the processing of the associated CDB's within the controller is potentially nonsequential. This fact is reflected in the order that the corresponding responses are delivered, via ICMBs, which does not have to be the same as the order in which the original commands were sent. Many of the performance enhancements under consideration for this product will involve rearranging requests internally to reduce device latency. Therefore, to remain compatible with this upward migration path, the host should not assume that the responses are going to come back in the same order as the commands were sent.

The Venus/ESDI controller in contrast to the WD7000-ASC controller will never reject a byte command at the command register because its internal buffers are full. It accomplishes this by maintaining a buffer, which only buffers up byte commands, and is larger than the maximum possible number of OGMB's. In other words it is greater than 64 bytes. Since non-mail box commands are executed immediately and therefore require no buffering and since the number of outstanding mail box type commands can't exceed the number of outgoing mail boxes, this buffer can't overflow. This does not mean, however, that there are no limits to the buffering capability of the controller. It turns out that the LCPU is limited internally when it comes to storing the CDB's. When it no longer has enough internal memory to store the next CDB, it refrains from clearing the high order byte of the associated OGMB. This limits the host to a maximum of one outstanding command per not yet buffered mail box. It also makes it possible for the host to make the inference when all of its OGMBs are full, that the LCPU has exhausted its buffers. Typically the host will request an "Interrupt on Free OGMB" command at this point. This might also be a good point to look for a lost or garbled mail box type command via the express Inquire command.

5.3.2. Incoming Mail

The host is always notified about the completion of a mail box type command via an ICMB. Whenever an ICMB is used in this way it contains the host address of a CDB in its low order bytes. This CDB is of interest to the host in three ways. First, it contains the original command. Second, in dedicated fields it contains the associated response. Third, it can now be re-used.

This implies a certain protocol between the host and the controller with respect to CDB usage. When the host sends a mail box type command, he must not release the CDB until after he receives an ICMB command with the address of the CDB in its low order bytes. The controller will remember the address of the host CDB for the duration of the command by storing this information in the internal buffer used by the command. When the controller completes processing for the command, it uses the stored address of the hosts CDB to DMA status and/or error information back into the original CDB. Following this it places the address of the hosts CDB into an available ICMB which is eventually delivered to the host.

Delivery occurs when the controller writes the number of the ICMB into the Interrupt status byte. This both interrupts the host and informs him of the number of the ICMB that needs to be serviced. After the host picks up the mail he must strobe the Interrupt acknowledge line. This completes a handshake that allows the controller to present the next ICMB to the host. The controller keeps track of ICMB's internally. It knows which ones are available, which ones are in use, and how to queue them up for delivery to the host.

In addition to delivering responses to mail box type commands, ICMB's can also be used to deliver unsolicited attention messages from the controller to the host. This type of ICMB has no associated CDB. It has a command code in the upper byte of the mail box. The lower bytes of the mail box are not used. Currently, no unsolicited attention messages are defined for the Venus/ESDI controller.

5.3.2.1. ICMB Return Status

The ICMB return status bytes are broken down into two classes. The first class (00-7F) is known as solicited interrupt status. These are codes accompanying interrupts which were in response to a host command to the controller. The second class, unsolicited interrupts, accompany an interrupt as a result of a controller detected condition. These interrupts do not have an associated CDB. There are no unsolicited interrupts currently defined.

Code	Definition / Effect
00	Reserved
01	Command complete; No errors
02	Command complete; Hard error
03	Command complete; Soft error
04->0F	Reserved
10	Invalid command
11	Aborted command
12->7F	Reserved
80->FF	Reserved (No unsolicited commands defined yet)

Table 2. ICMB Return Status Values

5.4. Command Blocks

None of the information on CDB's/SCB's/ICB's etc. for the WD7000-ASC has any relevance for the Venus/ESDI controller. There is absolutely no commonality between the two controllers in this area.

In the case of the Venus/ESDI controller all command blocks are referred to as CDBs, or Command Descriptor Blocks. Each CDB has a command opcode in its first byte. The command opcode defines the general category or type of command. For the most part each type of command has a CDB with a unique format, or layout of fields. Furthermore, many have CDB's with variable formats.

The host is responsible for creating and retiring CDB's. The controller will use the same CDB through which it receives a command to return the corresponding response. Therefore, the host must not retire a CDB until the response has been received. Also, the host should take advantage of the fact that both the original command fields and the response fields will be together in the same CDB at i/o post processing time.

6. COMMANDS

There are two broad categories of commands. Those that have associated CDBs and those that do not. Let's call the former mail box commands and the later immediate commands. The mail box commands always use the mail box system. The immediate commands never do. Both use the PIO command register byte.

6.1. Command Port Control Opcodes

These control commands, in the table below, are issued by the host by writing to the command port. Note that only the INITIALIZATION command is multi-byte and that the rest just require a single byte.

This table differs from the corresponding table in the WD7000-ASC spec in two of its entries. The 01NNNNNN entry in the table below, which specifies opcodes for starting express commands in OGMBs, does not exist in the WD7000-ASC spec. Furthermore, the 11NNNNNN entry, which is reserved in the table below, specifies scan mail boxes opcodes in the WD7000-ASC spec.

Opcode	Definition
00	No Operation
01	Initialization (8 Byte sequence)
02	Disable Unsolicited Interrupts
03	Enable Unsolicited Interrupts
04	Interrupt on Free OGMB
05-7F	Reserved
01NNNNNN	Start Express Command in OGMB #N
10NNNNNN	Start Command in OGMB #N
11NNNNNN	Reserved

Table 3. Command Port Opcodes.

6.1.1. No Operation (00)

Identical to 6.1.1 in WD7000-ASC spec

6.1.2. Initialization

All control commands are 1 byte instructions except the initialization command which must be sent to the controller after power on or controller reset. It may be re-issued any number of times thereafter if any of the parameters defined need to be changed. Any commands outstanding at the time of initialization must be re-issued by the host immediately following initialization. Furthermore, all re-initializations that occur with outstanding commands present must be accompanied by a reset. The host can send the command byte when the command port ready bit of the status port is a set. Writing a byte to the command port will reset this bit. The controller sets the bit when it is ready for the next byte. Using this programmed I/O technique, the remaining 8 bytes of the initialization command can be sent to the controller. These 8 bytes establish the number of outgoing and incoming mail boxes, the starting address of the entire mail box block, and the bus on/off times.

It is assumed that the OGMB's are followed by the ICMB's so that the relative mail box number, specified by the control port command byte, can be added to the 3 byte base address specified in the table below to compute the absolute mail box address. Further, it has been assumed that the mailbox or any of the data blocks do not wrap around memory from the highest 24 bit memory address to the lowest memory address of zero.

In the table below, a value of zero in bytes 8 and 9 will be taken to imply that at least 1 OGMB and 1 ICMB are allocated by the host. Hence the maximum value that the host can specify in these 2 bytes is 64 instead of 63.

Similarly if byte 2 is set to zero and a DMA transfer is required at least 1 word will be transferred by the first party DMA controller before giving up the host bus. A value of zero in byte 3 implies that the DMA controller will relinquish the bus for only 125 nanoseconds before re-arbitrating for it again. The maximum on time used should be less than 15 micro-seconds less ALL overhead time required to allow the host to service memory refresh cycles, including DMA bus arbitration time. The controller transfers a word every 375 nanoseconds via the first party dma interface, once it has control of the IBM-PC/AT bus.

Byte Num.	Bit Pos. 7654 3210	Definition
00	0000 0001	Command Opcode
01	0000 0000	Reserved
02	TTTT TTTT	Bus On Time (125 ns per count value)
03	TTTT TTTT	Bus off Time (125 ns per count value)
04	0000 0000	Reserved
05	AAAA AAAA	Starting Address of Mail Block (MSB)
06	AAAA AAAA	Starting Address of Mail Block
07	AAAA AAAA	Starting Address of Mail Block (LSB)
08	0NNN NNNN	Number of OGMB (max. 64) (0,1 = 1)
09	0NNN NNNN	Number of ICMB (max. 64) (0,1 = 1)

Table 4. Initialization sequence

6.1.3. Disable Unsolicited interrupts (02)

Forbids the controller to send unsolicited interrupts to the host.

6.1.4. Enable Unsolicited interrupts (03)

Allows the controller to send unsolicited interrupts to the host.

6.1.5. Interrupt On Free OGMB (04)

Identical to 6.1.5 in WD7000-ASC spec

6.1.6. Start express command (40 - 7F)

This command initiates a singular OGMB type command for express processing by the controller. The lower six bits specify the outgoing mailbox number. This command is identical to the Start command in every respect except one. It differs in the way that the controller will internally queue the corresponding CDB. The controller will service an express command immediately. Furthermore, it will not queue up express commands, like it will regular start commands. It will only process one express command at a time.

6.1.7. Start Command (80 - BF)

Identical to 6.1.6 in WD7000-ASC spec.

6.1.8. Start Multiple I/O (C0-FF)

This command is not supported by the Venus/ESDI controller. These bits are currently reserved.

6.2. Host initiated CDB commands

The following commands are sent from the host to the controller using a CDB. The commands are distinguished by an opcode, which is written into the 1st byte of the CDB by the host before sending the command. The supported opcodes are described below.

Command	Type	Bits 7 6 5 4 3 2 1 0
Read	nonexpress	0 0 1 0 0 0 L T
Write	nonexpress	0 0 1 1 0 0 L T
Format	nonexpress	0 1 0 1 0 0 0 0
Read verify	nonexpress	0 1 0 0 0 0 0 T
Seek	nonexpress	0 1 1 1 X X X X
Restore	nonexpress	0 0 0 1 X X X X
Diagnose	express	1 0 0 1 0 0 0 1
Load Param Block	nonexpress	1 0 0 0 1 0 1 0
Abort	express	1 1 1 1 0 0 0 1
Inquire	express	1 1 1 1 0 0 1 0
Esdi	either	1 1 1 1 0 0 1 1
Read Parameters	either	1 1 1 1 0 1 0 0
Read Defects	nonexpress	1 1 1 1 0 1 0 1

Table 5. OPCODES

T = 0 : Enable retries.

T = 1 : Disable retries.

L = 0 : Normal mode, selected ECC function is performed.

L = 1 : Sector is extended by 7 bytes. No ECC is generated or checked.

6.2.1. Generic CDB fields for command phase

The following generic descriptions provide default explanations for some common CDB command fields. There is a lot of commonality in the way that some command fields are defined and used. Rather than repeat all the details of these fields for each seperate CDB, the common information has been extracted and brought together in this section.

6.2.1.1. Sector count field

The sector count field defines the number of sectors to be transferred during a Verify, Read, Write, or Format_track command. During a multi-sector operation, the sector count is decremented and the sector number is incremented. When the disk is being formatted, the number of sectors per track must be loaded into this field for each Format command. The controller supports multi-sector transfers across track and cylinder boundaries. The sector count field must be loaded with the number of sectors to be transferred for any data-related command.

6.2.1.2. Cylinder Number fields

The target number for Read, Write, Seek, and Verify commands is loaded into these fields as shown in the following figure. The cylinder-number fields address up to 2048 cylinders.

	Cylinder High	Cylinder Low
Field bits	76543210	76543210
Cylinder bits	-----A98	76543210

Table 6. Cylinder Number Registers

6.2.1.3. Drive/Head field

Bit 7	Set to 1
Bit 6	Set to 0
Bit 5	Set to 0
Bit 4	Drive Select--This bit selects the drive. A 0 indicates the first fixed disk drive, and a 1 indicates the second.
Bit 3-0	Head select Bits--Bits 3 through 0 specify the desired read/write head. Bit 0 is the least-significant(0101 selects head 5). The controller supports up to 16 read/write heads.

6.2.1.4. Plo length register

This field is used to specify the length of the Data PLO sync field during WRITE commands and to specify the length of the ID PLO sync field during FORMAT commands. During read commands it specifies the ISG delay.

6.2.2.
Generic CDB fields for response phase

The following generic descriptions provide default explanations for some common CDB response fields. There is a lot of commonality in the way that some response fields are defined and used. Rather than repeat all the details of these fields for each seperate CDB, the common information has been extracted and brought together in this section.

6.2.2.1. Status field

The contents of this field will correspond to the description in the IBM/PC-AT technical reference manual for all commands that have AT equivalents (i.e. read,write,format etc.). For other commands, these fields will be described in the command specific descriptions of the command in this manual.

Bit 0 Error--A 1 on this bit indicates that an error occurred in the execution of the command, and that one or more bits are set in the error register.

Bit 1 No generic definition.

Bit 2 Corrected Data--A 1 on this bit indicates that the data read from the disk was successfully corrected by the ECC algorithm. Soft errors will not end multi-sector operations.

Bit 3 No generic definition.

Bit 4 Always 1.

Bit 5 Write Fault--A 1 on this bit indicates that a write fault occurred during the execution of the command.

Bit 6 Drive ready.

Bit 7 No generic definition.

6.2.2.2. Error field

The contents of this field based on the description in the IBM/PC-AT technical reference manual for all commands that have AT equivalents (i.e. read,write,format etc.). For other commands, these fields will be described in the command specific descriptions of the command in this manual.

The data in this field is valid only when the error bit in the status register is set, unless the processor is in diagnostic mode. Diagnostic mode is the state immediately after power is switched on or after a Diagnose command. In these cases, the register must be checked regardless of the status register indicator. The following are bit values for the diagnostic mode.

Diagnostic Mode:

01	No errors
02	Controller error
03	Sector buffer error
04	ECC device error
05	Control processor error

The following are bit definitions for the operational mode.

Operational Mode:

Bit 0 Data Address Mark (DAM) Not Found--This bit indicates that DAM could not be found within 16 bytes of the ID field.

Bit 1 No generic definition.

Bit 2 Aborted Command--A command is aborted based on the drive status (Write Fault, Not Seek Complete, Drive Not Ready, or an invalid command). The status and error registers may be decoded to determine the cause.

Bit 3 Set implies successful error recovery. Type of recovery will be as described by the table below.

Description	Error Bit 5	Status Bit 2
Single re-read	0	0
ECC corrected	0	1
Move head then read ok	1	0
Move head then ECC correct	1	1

Table 7. Types of Soft errors.

Bit 4 ID Not Found--The ID field with the specified cylinder, head, and sector number could not be found. If retries are enabled, the controller attempts to read the ID 16 times before indicating the error. If retries are disabled, the track is scanned a maximum of two times before setting this error bit.

Bit 5 Track/strobe offset.

Bit 6 Data ECC Error--This bit indicates that an uncorrectable ECC error occurred in the target's data field during a read command.

Bit 7 Bad Block Detect--This bit indicates that the bad block mark was detected in the target's ID field. No read or Write commands will be executed in any data fields marked bad.

6.3. Specific CDB descriptions

The following subheadings contain detailed descriptions of CDBs for host initiated mailbox type commands. Each section contains a CDB format table that describes the layout of the fields for a particular command. Following each format table is a description of the individual fields. Occasionally a field in the format table will not be contained in the following field descriptions. These fields assume the default definitions provided in the preceeding generic descriptions.

6.3.1. Read & Write commands

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	(Plo length register)*
6	Sector count
7	Sector number
8	Cylinder number LSB
9	Cylinder number MSB
10	Drive/Head
11	Sectors Actually Transferred
12	Starting Address of Data MSB
13	Starting Address of Data
14	Starting Address of Data LSB
15	Header Number 1 Starting Address MSB
16	Header Number 1 Starting Address
17	Header Number 1 Starting Address LSB

o
o
o

Header Number N Starting Address MSB
Header Number N Starting Address
Header Number N Starting Address LSB

Table 8. READ & WRITE commands

NOTE: The fields above in parenthesis and followed by an asterisk receive default values, if specified as 0. A non zero value in any of these fields will override the default. It is recommended that the driver writer, use the defaults.

6.3.1.1. PLO Length Register

When writing it specifies the data field PLO length. When reading it specifies the ISG (Inter Sector Gap) delay.

If a value of zero is given then a default value is provided by the controller. A non zero value, will override the default. It is recommended that the driver writer use the default.

6.3.1.2. Sectors Actually transferred

This value is returned to the host when the ICMB is delivered. It contains the number of sectors actually transferred. This may not be the same number as the number of sectors requested.

6.3.1.3. Starting Address of Data

This points to the beginning of a contiguous sequence of data blocks in the host DMA address space. The data portion of the transfer will be to or from this location.

6.3.1.4. Header Number 1...N

A list of Header addresses in host DMA space. There is one header for each block of data that is to be transferred. The headers are at random locations in the host DMA space. On the disk the headers and the data blocks are combined. Each header in the list is prepended to the block that occupies the same block position in the data blocks, as the header does in the header list. A maximum of 32 headers may be specified.

6.3.2. Read verify

This command is similar to a Read command except that no data is sent to the host. This allows the system to verify the integrity of the fixed disk drive.

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	Plo length register
6	Sector count
7	Sector number
8	Cylinder number LSB
9	Cylinder number MSB
10	Drive/Head
11	Number sectors successfully verified

Table 9. Read verify command

NOTE: See generic descriptions for all fields in this command.

6.3.3. Seek command

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	Reserved
6	Reserved
7	Reserved
8	Cylinder number LSB
9	Cylinder number MSB
10	Drive/Head

Table 10. Seek command

6.3.3.1. Cylinder fields

Specifies cylinder to seek to.

6.3.3.2. Drive/Head

Selects drive to perform seek on.

6.3.4. restore command

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Drive/Head

Table 11. Restore command

6.3.4.1. Drive/Head
Selects drive to perform restore on.

6.3.5. diagnose command

0	Opcode
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved

Table 12. Diagnose command

This information is confidential and proprietary to WDC and shall not be reproduced or further disclosed to anyone other than wdc employees, without written authorization from Western Digital Corporation.

6.3.6. Format

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	(Plo length register)*
6	Sectors/Track
7	(Gap Size)*
8	Cylinder number LSB
9	Cylinder number MSB
10	Drive/Head
11	Reserved
12	Interleave table MSB
13	Interleave table
14	Interleave table LSB

Table 13. Format command

NOTE: The fields above in parenthesis and followed by an asterisk receive default values, if specified as 0. A non zero value in any of these fields will override the default. It is recommended that the driver writer, use the defaults.

6.3.6.1. Plo Length Register

This field is used to specify the length of the id plo sync field. If a value of zero is given, then a default value is provided by the controller. A non zero value, will override the default. It is recommended that the driver writer use the default.

6.3.6.2. Sectors/Track

This field is usually known as the sector count field. In the case of the format command, its meaning is altered slightly. It still means a number of sectors. However, instead of denoting the number of sectors to read or write etc., it specifies the total number of sectors to format onto the track.

6.3.6.3. Gap size

Controls the Gap 1 and Gap 3 sizes during format commands. Ordinarily this field would specify the starting sector number. Since the entire track gets formatted according to the order specified in the interleave table, the starting sector number has no relevance when formatting.

If a value of zero is given, then a default value is provided by the controller. A non zero value, will override the default. A non zero value will be interpreted as the desired number of gap bytes. It is recommended that the driver writer use the default.

6.3.6.4. Interleave table fields

The interleave table is composed of two bytes per sector. The 1st two byte pair describes the 1st sector after index, the 2nd pair describes the 2nd sector after index, and so on. The first byte of each pair specifies whether the sector is good or bad. A value of 00 indicates a good sector, a value of 80 indicates a bad sector. The second byte specifies the interleave number for the sector.

The values for any specific interleave table will of course vary depending on the number of sectors/track, the interleave, and the bad blocks on any specific track. The table for a 2-to-1 interleave, for a track with 17 sectors and no bad sectors is: 00, 01, 00, 0A, 00, 02, 00, 0B, 00, 03, 00, 0C, 00, 04, 00, 0D, 00, 05, 00, 0E, 00, 06, 00, 0F, 00, 07, 00, 10, 00, 08, 00, 11, 00, 09. The actual number of two byte entries is specified in the Sectors/Track field.

6.3.7. Load Parameter Block

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	(ISG)*
6	(Data written in gaps)*
7	(Data written in pads)*
8	Sector size LSB
9	Sector size MSB
10	Drive/Head

Table 14. Load parameter block

NOTE: The fields above in parenthesis and followed by an asterisk receive default values, if specified as 0. A non zero value in any of these fields will override the default. It is recommended that the driver writer, use the defaults.

6.3.7.1. ISG

The ISG field specifies the delay from the index/sector pulse to the read gate.

6.3.8. Read Defects

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Drive/Head
11	Reserved
12	Defect list buffer MSB
13	Defect list buffer
14	Defect list buffer LSB

Table 15. Read Defects command

6.3.8.1. Defect list buffer

This field contains the address of a buffer in host address space. The defect list will be written out to this location. The buffer must be able to receive 2048 bytes of information.

6.3.9. Abort command

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	target_cdb_address MSB
6	. . .
7	LSB

Table 16. Abort command

6.3.9.1. Status

Status = 0 implies that command was aborted. Status =2 implies that command was not aborted.

6.3.9.2. Target_CDB_address

This is the address of the associated CDB in host DMA space for the command that we want to abort. Internally, the controller points every command to its associated host CDB. This allows this information to be used as a unique tag for every command at any point in its journey through the controller.

6.3.10. Inquire command

0	Opcode	
1	Esdi Status Msb	
2	Esdi Status Lsb	
3	Status	
4	Error	
5	target_cdb_address	MSB
6		⋮
7		LSB
8	Progress	MSB
9		LSB

Table 17. Inquire command

This command must be sent as an express command.

6.3.10.1. Target CDB_address

This is the address of the associated CDB in host DMA space for the command that we want to inquire about. Internally, the controller points every command to its associated host CDB. This allows this information to be used as a unique tag for every command at any point in its journey through the controller.

6.3.10.2. Progress

The controller keeps an internal progress number for every command. It initially sets this number to 1. When an inquiry is made the internal progress number for the target command is written into the progress field of of the Inquire command. Following this the internal progress number of the target command is incremented. The next time an Inquire command is sent for the same target the progress command that comes back should be greater than the previous one. If this ever fails to be the case, then this means that the controller, either can't access the target, can't do useful work on it, or has a fatal error.

6.3.11. Esdi command

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Drive/Head
11	Esdi_command MSB
12	Esdi_command LSB

Table 18. Esdi command

6.3.11.1. Esdi_command

The raw ESDI command contained in these sixteen bits is passed without interpretation to the ESDI device. The controller will provide the necessary parity bit. See ESDI document no. x3t9.3/86 for detailed description of ESDI commands.

6.3.11.2. Esdi_response

The raw ESDI response contained in these sixteen bits is passed without interpretation back to the host. See ESDI document no. x3t9.3/86 for detailed description of possible ESDI responses.

6.3.12. Read parameters

0	Opcode
1	Esdi Status Msb
2	Esdi Status Lsb
3	Status
4	Error
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Drive/Head
11	General Configuration MSB
12	General Configuration LSB
13	Number of Fixed Cylinders MSB
14	Number of Fixed Cylinders LSB

15	Number of Removable Cylinders MSB
16	Number of Removable Cylinders LSB
17	Number of removable drive heads
18	Number of fixed drive heads
19	Unformated bytes/track MSB
20	Unformated bytes/track LSB
21	Unformated bytes/sector MSB
22	Unformated bytes/sector LSB
23	Reserved
24	Number of sectors/track
25	Bytes per ISG
26	ISG Bytes after Index/Sector Pulse to WSplice
27	Reserved
28	Min bytes in PLO
29	Reserved
30	# words of vendor unique status
31	Reserved
32	Reserved
33	Reserved
34	Reserved
35	Reserved
36	Reserved
37	Reserved
38	Reserved

39	Cylinder switch skew
40	Head switch skew
41	Vendor ID
42	Extended Vendor ID
43	Controller ID MSB
44	Controller ID LSB

Table 19. Read Parameters

7. Suggested host communications protocol

Broad is the road and many are the host drivers that can be written to go down it, that will work with this controller's communications protocol. But narrow and cramped is the road and few are the host drivers finding it, that will work WELL with it. This is all the more true in the case of multi-tasking hosts. My goal in writing this section is to help the driver writer find the narrow and cramped road and to show the reasonableness of following it. Although, not a part of the formal communications interface, the following discussion is a very valuable addendum that, if followed, should greatly improve the extent to which the host and the controller work together. The end result should be an increase in peripheral performance, a decrease in host overhead, an increase in i/o reliability, and a reduction in the complexity of the host software.

At this point I'd like to introduce driverX. DriverX is a hypothetical host driver. It happens to possess what I consider to be an optimal communications interface to the Venus/ESDI controller. What follows is a description of DriverX's communications interface.

DriverX has n OGMB's where n is greater than 2 and preferably close to the maximum allowed, which is 64. In other words the more mail boxes the better. DriverX exclusively dedicates the nth OGMB to the task of sending express mail box type commands. The rest of the OGMB's are treated as a ring, which is dedicated exclusively to the task of sending non-express mail box type commands.

DriverX maintains a ring_pointer into the OGMB ring that always points to the OGMB to use for the next non-express mail box type command. Following initialization or reset DriverX sets ring_pointer to point to the 1st OGMB in the ring. When a new host request needs to be sent to the controller, DriverX first determines if there are any OGMB's available. It determines this by looking at the high order byte of the mail box pointed to by the ring_pointer. If it is not zero, then it knows that all the OGMB's are in use. This is a natural result of the way in which the host fills the ring of OGMB's and the order in which the controller empties it. The net result is that mail boxes are always filled and emptied in FIFO order, with the filler, namely the host, always remaining ahead of or even with the emptier, namely the controller. Furthermore, the ring is always filled by the host and emptied by the controller, in such a manner, that the ring position of the former is always ahead of or even with the latter. What we have is a tail chases head scenario, in which the host is the head and the controller is the tail. The tail will always be emptying the oldest of the filled mail boxes next, the head will always be filling the oldest of the emptied mail boxes next. When there are no more emptied mail boxes left, the host is pointing to what will be the next empty mail box, when it is emptied. Therefore, it is pointing to the mail box at the head of the ring of mail boxes, that are scheduled to be emptied by the controller. Therefore, it is pointing to a mail box for which the high order byte is currently non-zero. When the host finds itself in this situation it typically requests the controller to interrupt it when an OGMB frees up. If the OGMB that frees up is not the one that the ring_pointer is pointing to, then a communication error occurred somewhere along the line. Such an error can probably be found with the express Inquire command. The typical outcome of such a mismatch will be a reset. However, these are all unusual cases. Typically, he will find that the byte pointed to by the ring_pointer is 0, which means it's available for his use. Finding this to be the case DriverX proceeds to send the command.

To send the command he must obtain a place to put the command. Therefore, he 1st acquires a CDB from either a heap or a free list. Next he links the CDB to the tail of the completion_pending list, using pointer(s) that are just in front of the address that the controller will think is the beginning of the CDB. He then places the address of the beginning of the CDB sans linkage pointer(s) into the low order bytes of the current OGMB pointed to by ring_pointer. He then places the size of the CDB sans linkage pointer(s) into the upper byte of the OGMB. He then writes the bitwise or of the mail box number and 80 hex into the programmed i/o command register. He then advances the ring_pointer to the next OGMB. This will either be the next larger one, or the 1st one, if he's at the end of the ring. If there are any more outstanding host requests requiring a non-express mail box, then he repeats the process using the newly advanced ring_pointer, and so on. In this way the OGMB usage occurs in a very rigorously defined and easily traceable manner.

Eventually, command completions come in via ICMB's, which are accompanied by host interrupts. In a highly optimized controller, the completions are unlikely to come back in the same order as the original commands went out. When the completion comes back DriverX removes the CDB pointed to by the ICMB from the completion_pending list. He then passes it on to the next stage by linking it into a post_processing FIFO list. (Due to the random order in which the CDB's get removed from the completion_pending list, it would probably be best to use doubly linked lists). After removing the CDB from the completion_pending list, DriverX writes to the Interrupt acknowledge strobe to enable the controller to send the next ICMB.

Eventually, the CDB is pulled off the post_processing FIFO. When whatever postprocessing that needs to be done with it is completed, it is returned either to the free list, or the heap.

You might be wondering why DriverX bothers to link the CDB's into the completion_pending list. It turns out, that when the Venus/ESDI controller is reset, it loses all knowledge of commands in progress. This is different from the WD7000-ASC, which has a selective reset capability, that usually allows it to restore its own command stream. Because, the Venus/ESDI controller doesn't know how to re-issue the commands it was working on, or about to work on, prior to a reset, the host must do this for it. The completion_pending list provides the host with a historical record of all pending commands which is organized according to the order that they where issued. Therefore, re-issuing the pending commands after a reset, becomes a simple matter of processing all the CDB's on the completion_pending list, in FIFO order.

DriverX will occasionally issue an express mail box command. To do this he uses mail box n. He, also, uses a dedicated CDB that is permanently allocated and is only used for express commands. He, also, uses an internal flag that tells him, whether or not an express command is currently in progress. If no other express command is in progress, then DriverX will fill the CDB, with whatever information is necessary to specify the command that he wants to send via express mail. Next he will write the address of the CDB into mail box #n. Then he will write the size of the CDB into the high order byte of the mail box. At this point he will set an internal flag that tells him that he must wait for command completion before issuing another express command. Following this, he will write the bitwise or of 40 hex with the mail box number and write the result into the programmed i/o command register.

Eventually, he will receive an ICMB along with an interrupt. If the number of the ICMB is the same as the one that is dedicated to express commands, then he will proceed immediately to command post-processing. Afterwards he will reset his flag, thereby indicating that he's free to send another express command. Finally, he will write to the interrupt acknowledge strobe to re-activate the ICMB pipeline.