5. Rigid Disk Subsystem TABLE OF CONTENTS

5. Rigid Disk Subsystem

Subsystem Architecture

The IOP rigid disk subsystem provides support for the Dove workstation rigid disk operation. The subsystem has three main components:

- the disk drive, which provides local permanent file storage, virtual memory swapping for the Pilot operating system, and system booting
- the rigid disk controller (RDC), which supports labels and the various disk operations required by the Pilot operating system
- the DMA controller and the rigid disk FIFO, which transfer control, data, and status blocks between main memory and the rigid disk controller

Figure 5.1 illustrates the rigid disk subsystem.

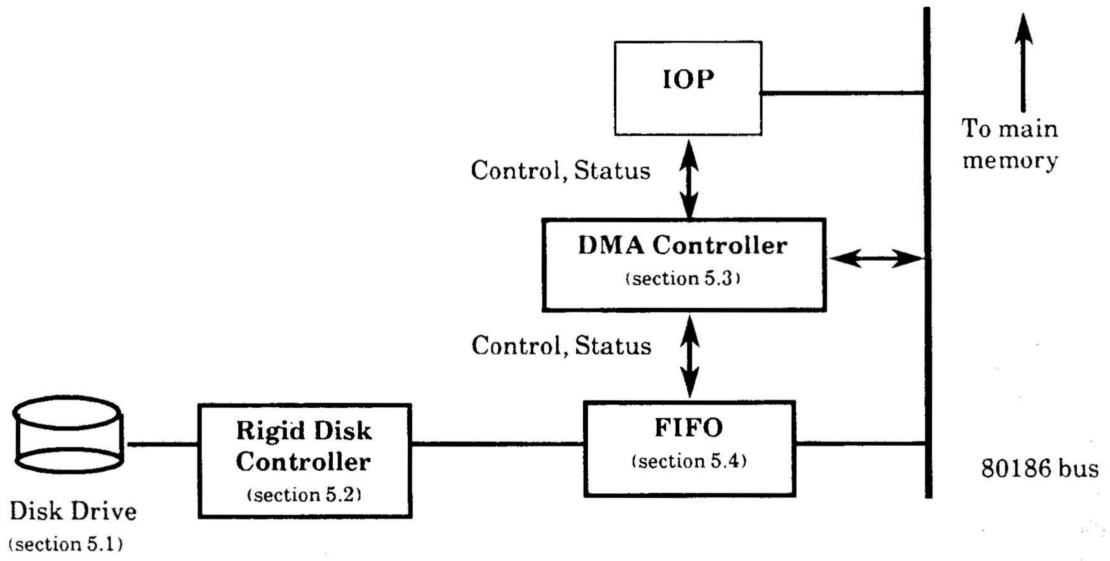


Figure 5.1. Rigid disk subsystem

Subsystem Programming Overview

The rigid disk subsystem is programmable by the IOP 80186. This section describes the steps and rules required for programming the subsystem.

The Rigid Disk Controller (RDC) and DMA Controller/FIFO are programmed independently. Software should ensure that all the programming steps are taken before issuing a data transfer request between main memory and the rigid disk.

Unlike the DMA/FIFO, programming the RDC is done via a control block called the Disk Control or Command Block (DCB) or Disk Operational Block (DOB). For this discussion, DCB and DOB are used interchangeably. The block contains: disk type; size and format information; the number and location of disk sectors to be accessed; the type of specific operation (read, verify, or write) to be performed on each field of the sector (address, label, data). The RDC performs its share of data storage based on the DCB contents.

Terms

Terms used in this overview are defined as follows:

transfer	– the action of transporting data (often in the form of a block) from one place of data storage in the system to another.
operation	– a collection of transfers that, when completed, moves a block of data between the two final points of main memory and the disk medium.

Information Types

The rigid disk subsystem deals with two types of information: control information and true data.

Control information is the information used to program the RDC and to query the status of disk accesses. An example is the Disk Operational Block.

True data is the information that travels between the two final points of main memory and the disk medium. Examples are: customer data, Mesa application programs, and information describing data structures.

DMA Controller

Data for the DMA controller is of both types; that is, true data and the DOB (control information). The DMA controller treats control information like a block of true data, except that DOBs are of a different length. In other words, to transfer DCBs to or from main memory and the RDC, the IOP 80186 programs the DMA controller as if a block of true data of a different length is being transferred.

Rigid Disk Controller

In contrast, the RDC receives the DCB (containing instructions and control information for the RDC) as its own control block and stores it in its scratch pad memory. The RDC treats the true data as the information to be transferred between the FIFO and the disk medium.

It is critical, then, for the RDC to know the nature of the information it finds in the FIFO, as it is critical for the IOP 80186 to know the nature of the information the RDC puts in the FIFO. It is not necessary for the RDC to know the real address of the true data or of the DCB in main memory.

Data Transfer Path

In reading this overview, note the path of data flow among the points of data storage; that is among main memory, the FIFO within the subsystem, the scratch pad of the RDC, and the rigid disk drive (magnetic medium used for long term storage). Direction of data transfer among these points must always be consistent to ensure a successful operation

For true data, the data transfer path between main memory and the disk medium, regardless of its direction, is divided into two segments: one segment between main memory and the FIFO, the other segment between FIFO and the disk medium.

Data transfer along the first segment (between main memory and the FIFO) is controlled by the DMA controller after it is programmed. Data transfer along the second segment is controlled by the RDC.

Figure 5.2 illustrates the control partitioning.

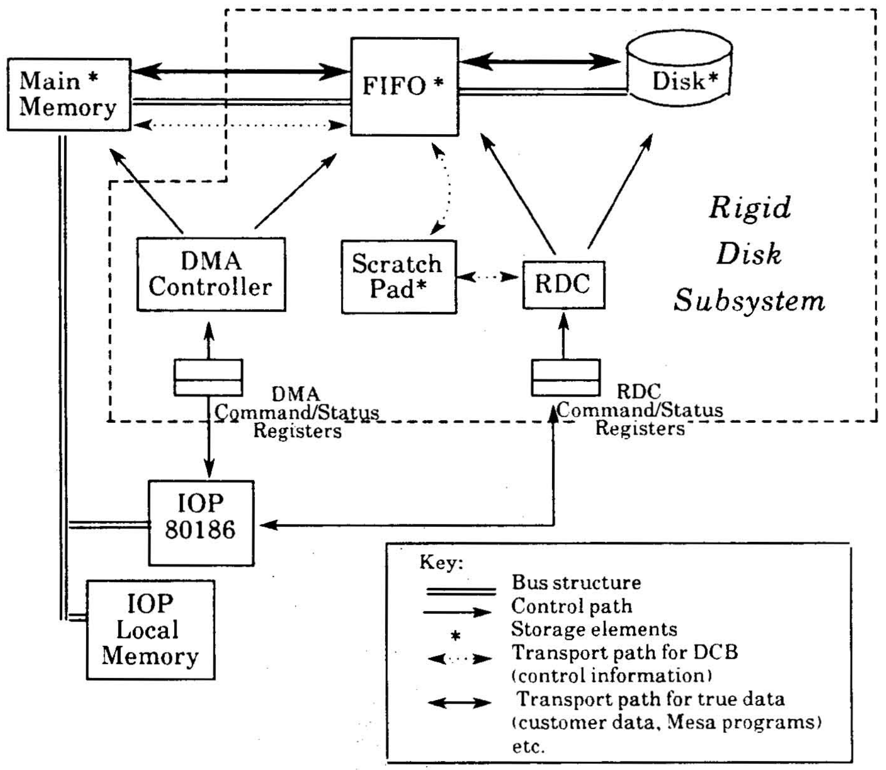

Main * Memory
FIFO *
Disk*
Rigid Disk Subsystem
DMA Controller
Scratch Pad*
RDC
DMA Command/Status Registers
RDC Command/Status Registers
IOP 80186
IOP Local Memory
Key:
Bus structure
Control path
* Storage elements
Transport path for DCB (control information)
Transport path for true data (customer data, Mesa programs) etc.

Figure 5.2. Data transfer path between main memory and the rigid disk

Communication Registers

Two registers provide communication between the RDC and the IOP 80186: the RDC command register and the RDC status register. The information exchanged through these registers includes:

- indication of sending or receiving a DCB, via the FIFO
- instruction to the RDC to begin its share of the operation, based on the DCB already available to it in its scratch pad
- notification to the IOP 80186 of the success or failure in performing the given command.

Table 5.2 in section 5.2.2.7 describes the available commands.

Within the scratch pad, the RDC maintains an updated DCB containing details of the error conditions as well as complete details of the functions performed. The IOP 80186 retrieves the DCB and examines its contents in order to determine details about the functions that the RDC has performed.

RDC and IOP 80186 Coordination

The RDC receives both the DCB and real data, and distinguishes between them by means of communications it makes with the IOP 80186 regarding their existence in the FIFO. **Therefore, the RDC and the IOP 80186 must be in phase with each other at all times regarding what is next available in the FIFO.** This becomes apparent when considering the required steps for programming the DMA controller and the possibility that the DMA controller will intervene in the process of information exchange between the RDC and the IOP 80186.

DMA Transfer Requirements

Consider the DMA controller requirements for programming. The DMA controller can transfer up to 256 words (or 512 bytes) of data in either direction between the two final destinations that it controls; that is, main memory and FIFO. The true data is always transferred in pages of 256 word; DCBs are always 34 words long.

When the DMA controller is programmed and enabled for a transfer, it proceeds to completion without distinguishing between DCB and true data. If another peripheral requires communication with the IOP 80186 during this transfer, then the DMA controller temporarily suspends the transfer, according to an arbitration priority scheme. The IOP bus is then open for the interrupting requestor, and later the DMA controller resumes the DMA transfer. Now, the RDC can also interact with the FIFO concurrently with DMA transfer between main memory and the FIFO, so the DMA controller will tolerate FIFO full or empty conditions during the transfer by relinquishing the IOP bus control and resuming later on.

Under these circumstances, the following situations are possible: (Software should be aware of these pitfalls)

The RDC finds the FIFO empty while it is expecting more data.

At the same time, the DMA controller has been suspended by an interrupt to the IOP 80186.

The IOP 80186, after serving the interrupt, ignores the right of the DMA controller to regain control of the bus.

The transfer does not proceed.

Another possibility is this:

The DMA controller is programmed for a true data block transfer of 256 words.

The RDC expects a DCB in the FIFO, and uses part of true data as a DCB block.

A random bit pattern of true data is interpreted as a disk operation command, and good data on the disk is overwritten with meaningless data.

General Example of a Data Transfer Process

The following is a general example of a suitable implementation of a data transfer process. For detailed programming steps for the DMA controller, FIFO, and RDC, refer to the corresponding programmer interface section.

1. The IOP programs the DMA controller to move DCB from main memory to the FIFO. Direction: from memory to FIFO.

2. The IOP programs the RDC to accept a DCB.

Note: The hardware for the RDC and the DMA controller is tolerant to data availability in the FIFO. Therefore, steps 1 and 2 need not be executed in this order, but to ensure reliability, perform the two steps as given. Moreover, it is advisable to wait until step 1 is completed and the DCB resides in the FIFO before initiating step 2.

3. The IOP waits until the RDC acknowledges the receipt of DCB. Since DCB is smaller than 256 words, the DMA controller need not be reprogrammed in the interim in order to complete transfer of the DCB to the FIFO. IOP 80186 code can guarantee a successful operation by waiting.

4. The IOP programs the DMA controller and the direction of the FIFO to move a page of data between main memory and the FIFO. The programming is based on information about the nature of the true data transfer; for example, its direction, types of functions required of the RDC, the number of memory pages and disk sectors involved, and so forth.

5. The IOP then instructs the RDC to begin its function, based on the DCB now maintained in the RDC scratch pad memory.

6. If the DCB instructions involve transfer of more than one page, then, when the DMA controller interrupts the IOP with an indication that one page has been transferred, the IOP 80816 verifies the DMA controller status for a successful transfer. The IOP 80186 then reprograms the DMA controller for

transfer of the next page, in the exact page order designated in the DCB.

Note: The DMA controller can be reprogrammed many times during execution of a DCB. Before reprogramming, the status of the DMA controller should be checked for a successful transfer.

7. After programming the DMA controller for the final page as designated in the DCB, the IOP waits for the interrupt from both the RDC and the DMA controller. Depending on the transfer direction, one of the two interrupts indicates completion of the transfer for all data pages, and hence completion of the instructions given in the current DCB.

8. The IOP then requests the RDC to store the updated DCB in the FIFO.

9. The IOP next programs the DMA controller to transfer the contents of the FIFO (that is, the updated DCB) to a designated destination in main memory. Direction: from FIFO to memory.

Note: For reliability, the FIFO status should be verified to be empty before step 8, and step 8 should be completed before step 9 is initiated.

10. The IOP, in response to completion of transfer sent by the DMA controller, accesses the updated DCB in main memory and examines its contents for the status of each sector accessed.

Error handling

The RDC always stops for fatal errors (hard errors). The RDC may stop before a disk sector data transfer or after correction. During data transfer, the RDC or DMA controller will interrupt the IOP for hard errors.

Non-fatal errors (soft errors) detected by the RDC are communicated at the end of transfer via the updated DCB.

Summary

In summary, it is important to note again these points.

1. A DCB often governs the transfer of more than one memory page at a time (512 bytes, which is equal to one disk sector). The DMA controller transfers only one page at a time. Hence, the IOP 80186 programs the DMA controller many times before it changes the DCB, or even verifies the results of a DCB. Therefore, the IOP 80186 code must be up-to-date regarding the sequence of events within the rigid disk subsystem.

2. The IOP is likely to run two processes in order to carry out a transfer operation. One process programs the DMA controller; the other process monitors the RDC activities. Each process accepts interrupts from its hardware, and must coordinate activities with the other process to ensure a successful transfer operation.

5.1 Rigid Disk Drive

A single rigid disk drive is supported, either half- or full-height. Performance requirements of the drive meet or exceed the Shugart SA1004 performance standards.

5.1.1 Hardware

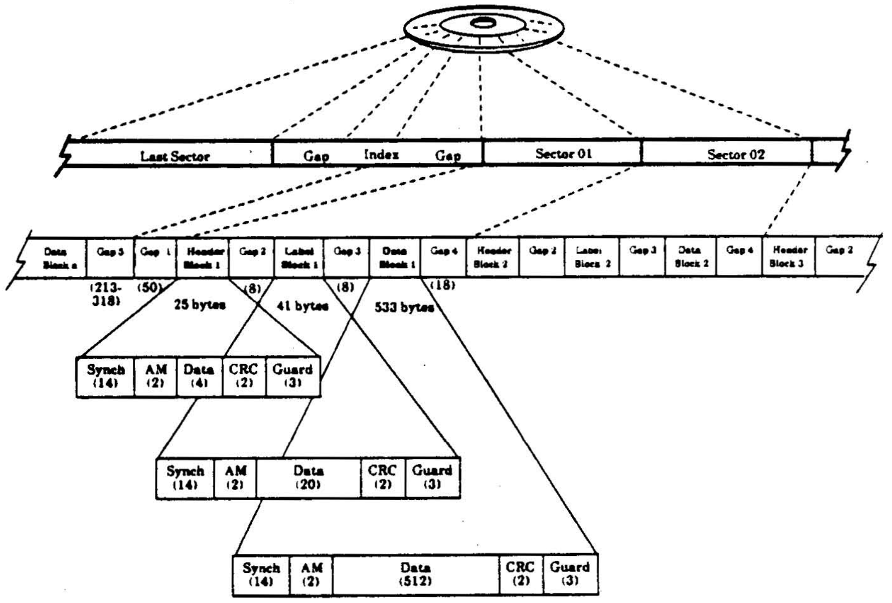
The drive is a 5¼-inch Winchester drive with 3 bits of head select (eight heads maximum), four drive select lines (four drives maximum), and 5 Mbits/sec transfer rate with data encoding and decoding done by the controller. The drive interface is the ST412/St506 interface. The drives use dc spindle and head motors. If the power supply can handle the drive's current requirements, then varying line voltages and frequencies should not cause line problems.

Spindle speed is required not to vary by more than one percent from nominal over the entire range of specified operating conditions.

The interface to the drive supports one departure from the S7412/506 specifications, viz., 4 bits of head select are provided. This change allows 16 heads maximum as long as the drive does not require an external reduce write current signal. The bit normally used for reduce write current is reassigned to head select 3. (This bit can be used as a reduce write current line, if desired.)

5.1.2 Theory of Operations/Programmer Interface

The format of a disk is the pattern written on the disk to define the sector numbers; that is, the format labels the sector, and provides a defined space for the data. Figure 5.3 illustrates the format for the rigid disk drives.



Notes:

Number in parentheses is number of bytes.

During Write Long the guard becomes 1 byte; the other two bytes are for additional data.
Read Long is not allowed.

Nominal Track length = 10416 bytes
1 minimum track = 10311 bytes (nominal - 1 %)

```
1 track  = Gap 1 + [Header + Gap2 + Label + Gap3 + DataField] * 16 + Gap4 * 15 + Gap6
         = 50 + [615] * 16 + 18 * 15 + Gap6
         = 50 + 9840 + 270 + Gap6 = 10160 + Gap6
```

Since a nominal revolution = 10416 bytes, Gap6 (Nominal) = 256
Since a minimum revolution = 10311 bytes, Gap6 (Minimum) = 151
(Gap6 must be at least Gap4 + 1% of the track length excluding Gap6 = 120 bytes)

Figure 5.3. Rigid disk format

Cylinders

The disks of a drive are divided into cylinders. A cylinder is the surface of the disks swept by all the heads as the disk revolves without moving the heads.

Tracks

A cylinder is further divided into tracks. A track is the area on one surface of a disk swept by one head as the disk revolves. A track extends from the leading edge of the index pulse to the leading edge of the next index pulse. The index gap of 50 bytes follows the index pulse. The gap is filled, as are all gaps, with zeros or E5(H). This gap allows for the format operation to overrun the index pulse by a few bytes. It also provides room for the read amplifiers to recover after switching heads.

Sectors

A sector is a subdivision of a track. In a soft-sectored drive, each sector begins with a header block. The header marks the beginning of a sector and uniquely numbers the sector. Each sector has a unique combination of cylinder number, track number within the cylinder (head number), and sector number within that track. Each sector is divided into three blocks: a header block, a label block, and a data block. Each block has the same parts: a synch field, an address mark byte, a block ID byte, a data field, a CRC field, and a Guard field. The address mark byte and the block ID byte are frequently called the address mark.

Each block may be written at a different time, but once a block is written, the successive blocks in that sector must be written at the same time. The header block can only be written when all headers of a track are written. In this controller there is no "Write Header" command. Headers may only be written using the "Format" command, and the minimum length for a format command is one track.

Notes

The sync field provides room for the phase locked loop to lock.

The guard field eliminates problems caused by ending write current too close to the CRC field.

The gap between blocks provides time to set up the next operation and allows for variations in disk speed.

The gap between one data block and the next header is long enough to allow write current to be turned off without affecting the next header. This spacing takes into account all possible variations in disk speed.

The gap at the end of the track provides space enough that a track formatted at the fastest allowable disk speed will fit completely between index pulses.

5.2 Rigid Disk Controller

The rigid disk controller directly controls the drive and read/write logic, and consists of a 2K x 24 PROM control store, a fast 8-bit microcontroller (8x305) with a 256 byte scratchpad memory, and read/write logic.

The rigid disk controller can function with any 5¼-inch drive that supports the ST412/ ST506 interface. The controller reads, writes, or verifies any number of contiguous sectors and will switch heads if necessary. The commands supported are: Restore (Recalibrate), Format, Write Data, Write Label and Data, Read Data, Read Label and Data, Read Label and Skip Data, and Verify Data. The diagnostic commands that are supported are: Read Header, Read Label and Data.

5.2.1 Hardware

The rigid disk controller components are off-the-shelf components. The non-industry-standard format and operations are supported by the controller microcode.

Figure 5.4 illustrates the 50-pin 8x305 microcontroller chip. Table 5.1 lists the pins and signals.

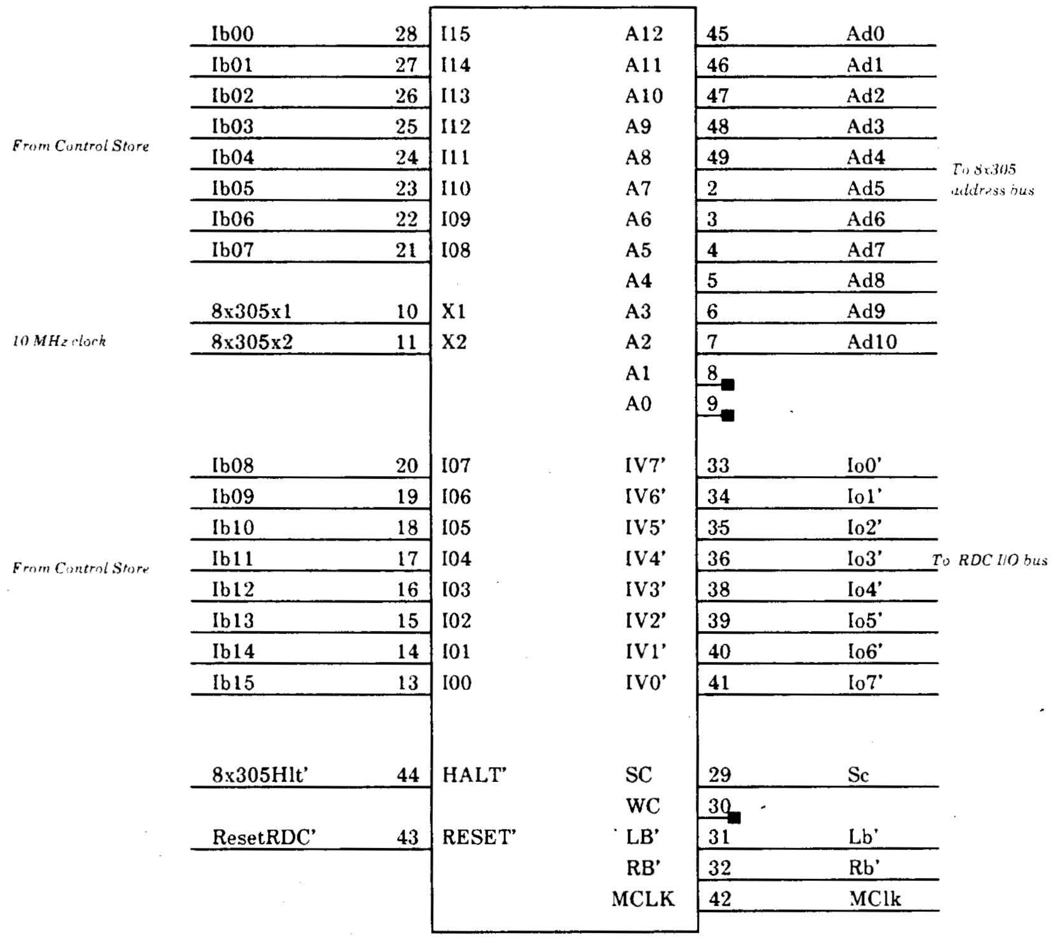


Figure 5.4. 8x305 microcontroller pin-out

Table 5.1. 8x305 Pin Description

Symbol	Pin #	Type	Function
A0-A12	2-9 45-49	Output	Program address lines to 8x305 bus. A0 and A1 (pins 8 and 9) are not used.
HALT'	44	Input	Not used in the RDC.
I0-I15	13-28	Input	Instruction lines for 16-bit instructions from program control store.
IV0'-IV7'	33-36 38-41	I/O	Bidirectional 3-state lines for data and/or addresses to RDC I/O bus. A low voltage level is binary 1.
LB'	31	Output	When low, allows access to devices connected to the left bank.
MCLK	42	Output	(Master Clock) clocks I/O devices and/or synchronizes external logic.
RB'	32	Output	When low, allows access to devices connected to the right bank.
RESET'	43	Input	When low, initializes the 8x305; that is, sets program counter/address register to zero and inhibits MCLK. While RESET' is low, LB' and RB'are forced high asynchronously.
SC	29	Output	(Select Command) Indicates that data is being written.
WC	30	Output	Not used
X1, X2	10, 11	Input	10 MHz clock source with complementary output.

Pin # 1 = VCR, input from series-pass transistor.
Pin #12 = Ground
Pin # 37 = Vcc, +5V power supply
Pin #50 = VR, output reference voltage for external series-pass regulator transistor.

5.2.2 Theory of Operations

Figure 5.5 illustrates the rigid disk controller.

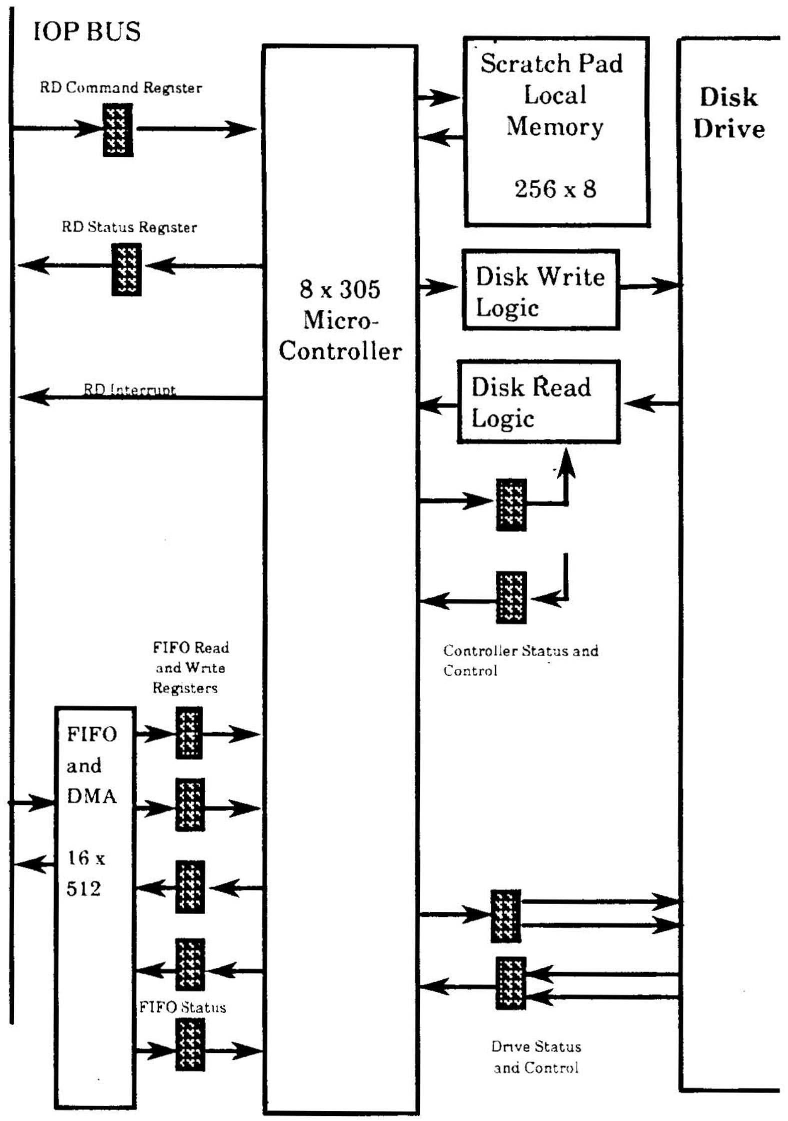


Figure 5.5. Rigid disk controller block diagram

The 8x305 microcontroller controls the rigid disk controller. The communication with the IOP and the disk control hardware is also shown.

Blocks in the diagram are discussed below.

5.2.2.1. Command and Status Registers

The 8-bit command and status registers permit the IOP and the 8x305 microcontroller to synchronize communication. The command register is loaded by the IOP and read by the 8x305. The status register is loaded by the 8x305 and read by the the IOP. The 8x305 interrupts the IOP, but is itself not interruptible. The disk command block and data are passed to and from the controller via the FIFO and the DMA.

5.2.2.2. Scratch Pad and Local Memory

The scratch pad stores the image of the header and label and other information needed by the controller.

The scratch pad memory consists of two high speed 256 x 4 bit RAMs. The scratch pad address register consists of two 4-bit parallel-loadable counters. The WriteMA instruction parallel-loads this address counter. Subsequent reads and writes to and from the scratch pad use this address. Any read command with the appropriate bit set increments the address. This increment MA feature is used only with a read into 8x305 command.

Note: The address counter connection appears to decrement the address instead of incrementing it. The counters are defined for positive-true logic while the 8x305 I/O bus is negative-true. With negative-true logic, the positive-true decrement becomes a negative-true increment.

5.2.2.3. Drive Status and Drive Control Registers

The drive status and drive control registers are used by the 8x305 microcontroller to monitor and index the current state of the drive. Control signals are: Head Select, Drive Select, Step Direction In, Step, and Write Enable. Status signals are: Track00, Ready, Seek Complete, Write Fault, and Index.

5.2.2.4. Microcontroller Data Paths

Figure 5.6 illustrates data paths of the 8x305 microcontroller.

The 8x305 sends address signals to three PROMs. Two of the PROMs send instructions back to the 8x305. The third PROM stores input/output portions of the commands. The I/O portion of each instruction and 8x305 portion are sent to I/O control and to the 8x305 at the same time.

The 8x305 communicates with the rest of the system via an 8-bit, bi-directional I/O bus. The connections to the 8x305 illustrated in Figure 5.5 are connections to this I/O bus.

The 8x305 sends out timing signals to synchronize the system with its own internal timing. The 8x305's clock is the 10MHz 2x write clock used by the disk write circuits. The 8x305 executes each instruction

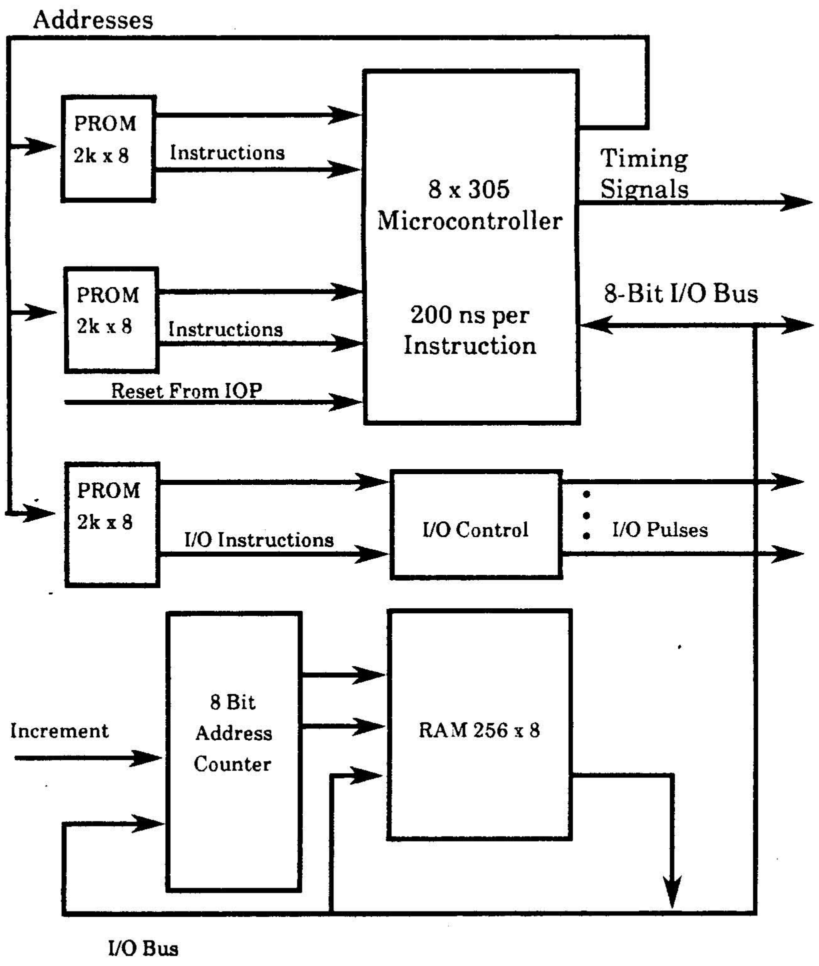


Figure 5.6. RDC microcontroller data paths

in 200 ns; each bit to or from the disk also takes 200 ns. The IOP 80186 controls the reset line to the 8x305.

5.2.2.5. Write Logic Data Path

Figure 5.7 illustrates the write logic data path.

The 8-bit parallel data from the I/O bus is loaded into the parallel-to-serial converter. The serial output from this chip, in NRZ form, is sent to the CRC chip, where the CRC is accumulated. From the CRC chip, the serial data goes to the MFM generator, when the data is converted to MFM format.

The MFM generator also provides the early, nominal, and late signals necessary for precompensation. If precomp is not enabled, then only nominal is used. The MFM signal goes through the delay line which produces three outputs, each 12 ns apart. These early, nominal, or late MFM signals from the MFM generator are gated with the early, nominal, or late signals from the delay line to produce the correct

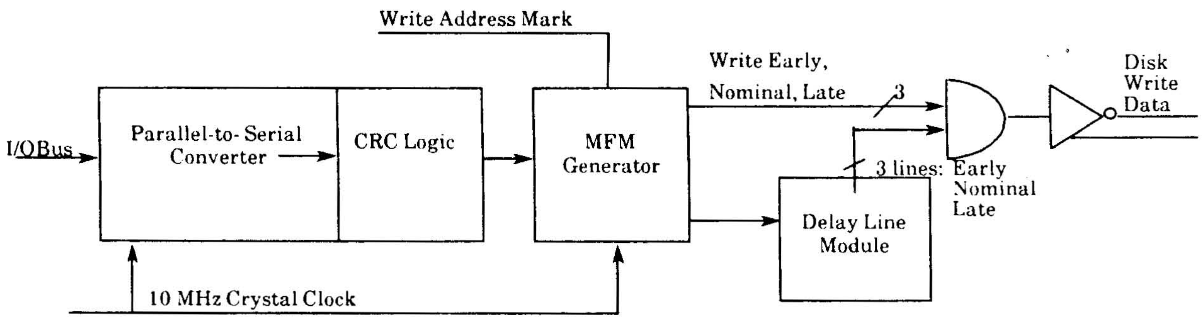


Figure 5.7. RDC write logic data path

output. The write data signal is converted to a differential signal for noise immunity, and is sent to the drive.

When the data has completely passed through the CRC generator, a control signal from the 8x305 changes the internal configuration of the CRC chip. This change ensures that the check bytes are not accumulated, but are instead sent out following the data. The MFM generator also has a provision to delete a clock, producing the missing clock needed to generate the address mark.

Note: Two control signals have names that include "write": write and write gate. "Write" sets chips that have both a read and a write function to the write function. Write gate enables write current in the drive head and is turned on when all components of write logic are ready to write.

5.2.2.6. Read Logic Data Path

Figure 5.8 illustrates the read logic data path. Refer to section 5.2.3.2 for timing of this data path.

The disk is normally in a read condition with read gate off. A chip with both read and write functions is normally in read mode. When write is enabled, the chip switches to write mode; read is enabled with a read gate.

Differential data from the disk enters as disk read data, is converted to single-ended data, and goes both to the phase locked loop. When read gate is off, the PLL locks on the 2x write clock, and thus is near the frequency required to read data. When read gate is turned on by the 8x305, the PLL switches to disk read data. When the PLL locks, it generates the signal LockDet.

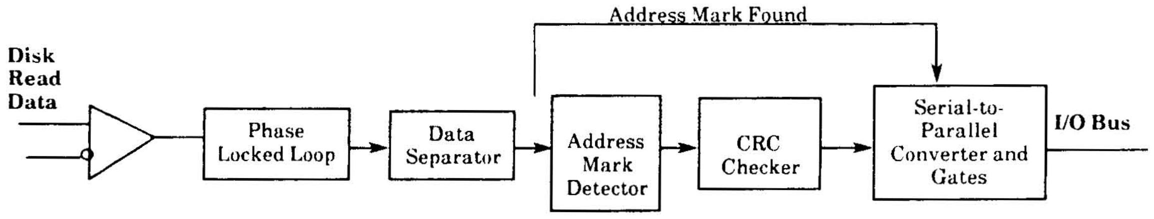


Figure 5.8. RDC read logic data path

The data separator separates the data pulses from the clock pulses and sends the separated pulses to the address mark generator. The data separator produces the signal RdDataFnd when the first "1" bit is detected.

The address mark detector looks for a certain pattern in the data and clock output of the data separator. This unique pattern is the address mark. The address mark found signal (AmDet) enables the serial-to-parallel converter.

Data passes through the CRC checker, which is the same chip that generates the CRC check bytes for a write operation. The 8x305 sends a signal to the checker after the entire data stream, including the check bytes, has passed through the checker. When the checker is finished, the internal shift register should contain all zeros. Anything else is an error. The contents of the internal shift register are shifted out, following the data, to the serial-to-parallel converter and then to the 8x305.

The serial-to-parallel converter accumulates the data, including the status of the CRC register, into 8-bit words and makes these words available to the 8x305. The converter has an internal counter that controls the transfer of data between the internal shift register and the internal buffer register. This counter also signals the 8x305 that another byte is available. The 8x305 now has seven instruction times to read the data before the next byte is loaded on top of the buffer.

5.2.2.7. Reading and Writing the DMA and FIFO

The 8x305 reads from and writes to the FIFO, but has no control over FIFO direction. The IOP initializes the FIFO to give the FIFO the proper direction and to clear the FIFO to empty, when appropriate.

Disk operations are set up in the Disk Command Block by high level programs. Table 5.2 lists disk commands and rigid disk controller operations.

Table 5.2. Disk Commands and RDC Operations

Disk Commands		RDC Operations
NOP-IDLE		Restore-Recalibrate
GetCommandBlock		Format Tracks
ExecuteCommandBlock		Read Data
LoadCommandBlock		Write Data
		Write Label and Data
		Read Label and Skip Data
		Read Label and Data
		Verify Data

5.2.3 Programmer Interface

The following subsections describe the rigid disk controller registers, normal operation sequence and error recovery sequence, and timing.

5.2.3.1. Registers

Two 8-bit hardware registers communicate between the rigid disk controller running on the IOP and the microcode running on the 8x305.

The disk controller command register is hardware port 0214H in the rigid disk controller. The controller treats the command register as a write-only port, and issues commands through this register. (The opposite is the case on the 8x305.)

The disk controller status register is also hardware port 0214H. The controller treats the status register as a read-only port, and obtains status information from this register. (The opposite is the case on the 8x305.)

Disk Controller Command Register

Figure 5.9 illustrates the disk controller command register. Bits 08 through 15 are not used; bits 00 through 07 are programmable.

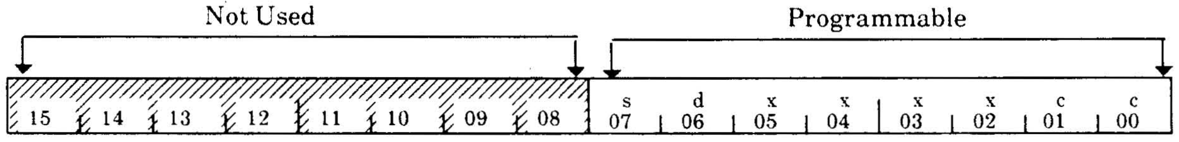


Figure 5.9. RDC command register (write-only I/O Addr = 0214 hex)

The bit-level definition of the command register is: **sdxx xxcc,** where

- **s**, when s = 1, is a request to stop at the end of the present sector on the commands read/write/verify. When s = 0, this bit is in normal running mode.
- **d**, when d = 1, indicates that the given command is a diagnostic function.
- **x** bits are unassigned bits, currently set to zero; that is, 0000.
- **cc** defines the command from the IOP's handler to the 8x305 microcode, as follows:

00 = >	Go to and stay in idle mode
01 = >	Fetch the disk command block (DCB)
10 = >	Execute the DCB
11 = >	Store the DCB

Disk Controller Status Register

Figure 5.10 illustrates the RDC status register. Bits 08 through 15 are not used; bits 00 through 07 are programmable.

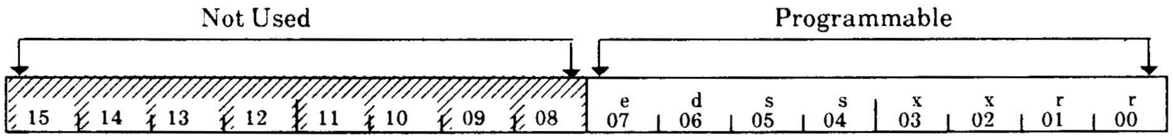


Figure 5.10. RDC status register (read-only I/O Addr = 0214 hex)

The bit-level definition of the status register is: **edss xxrr,** where

- **e**, when e = 1, indicates that an error has occurred. When e = 0, assume that no error condition has occurred.
- **d**, when d = 1, indicates that the operation has been completed. Otherwise, d = 0.
- **ss** bits indicate that a special operation has occurred. Normally ss = 00.
  ss = 01 indicates a FIFO error has occurred: the FIFO is not empty at the start of the control block loading operation. The Disk Control Block contents will not start at the normal location in the FIFO.
  Note: Errors are always shown in the DCB. However, if DCB cannot be read, the FIFO error is pointed out in the ss bits.

  Note: ss = 10 and ss = 11 are unassigned.
- **xx** are currently unassigned bits. The controller does not assume that the bits are 00; however, the 8x305 places 0s in these bits.

- **rr** repeats (echoes) the command bits (cc) in the disk controller command register and corresponds one-to-one to the cc definitions given above.

Register Sequences

Table 5.3 lists the contents of both registers as the controller goes through a normal command sequence. Table 5.4 lists the registers' contents during an error recovery sequence.

Table 5.3. Normal Register Sequence

Disk Controller Command Register	Disk Controller Status Register	Remarks
0000 0000	0000 0000	In idle state.
0000 0001	0000 0000	IOP issues Fetch DCB.
0000 0001	0000 0001	8x305 receives fetch command and begins fetch.
0000 0001	0100 0001 *	8x305 has fetched DCB and has issued an interrupt.
0000 0010	0100 0001	IOP issues Execute DCB.
0000 0010	0000 0010	8x305 receives execute command and begins execution.
0000 0010	0100 0010 *	8x305 finishes the command and interrupts the IOP.
0000 0011	0100 0010	IOP issues Store DCB .
0000 0011	0000 0011	8x305 receives command and begins storing DCB.
0000 0011	0100 0011 *	8x305 has completed store of DCB and interrupts the IOP.
0000 0000	0100 0011	IOP issues Go Idle.
0000 0000	0000 0000	8x305 goes to idle mode (no interrupt is given).

* 8x305 interrupts IOP

Table 5.4. Error Recovery Sequence

Disk Controller Command Register	Disk Controller Status Register	Notes
0000 0000	1100 00xx	IOP issues Go To Idle Mode.
0000 0000	0000 0000	8x305 goes to idle mode (no interrupt is given).
0000 0011	0000 0000	IOP issues Store DCB .
0000 0011	0000 0011	8x305 sees command and begins storing DCB.
0000 0011	0100 0011 *	8x305 has completed store of DCB and interrupts the IOP.
0000 0000	0100 0011	IOP issues Go To Idle Mode.
0000 0000	0000 0000	8x305 goes to idle mode (no interrupt is given).

* 8x305 interrupts IOP

5.2.3.2. Timing

Figure 5.11 illustrates the timing for the 8x305 described in section 5.2.1. Timing is based on the fall of MClk as 0 or 200 ns. For minimum/maximum timing information, please see the 8x305 specification sheet.

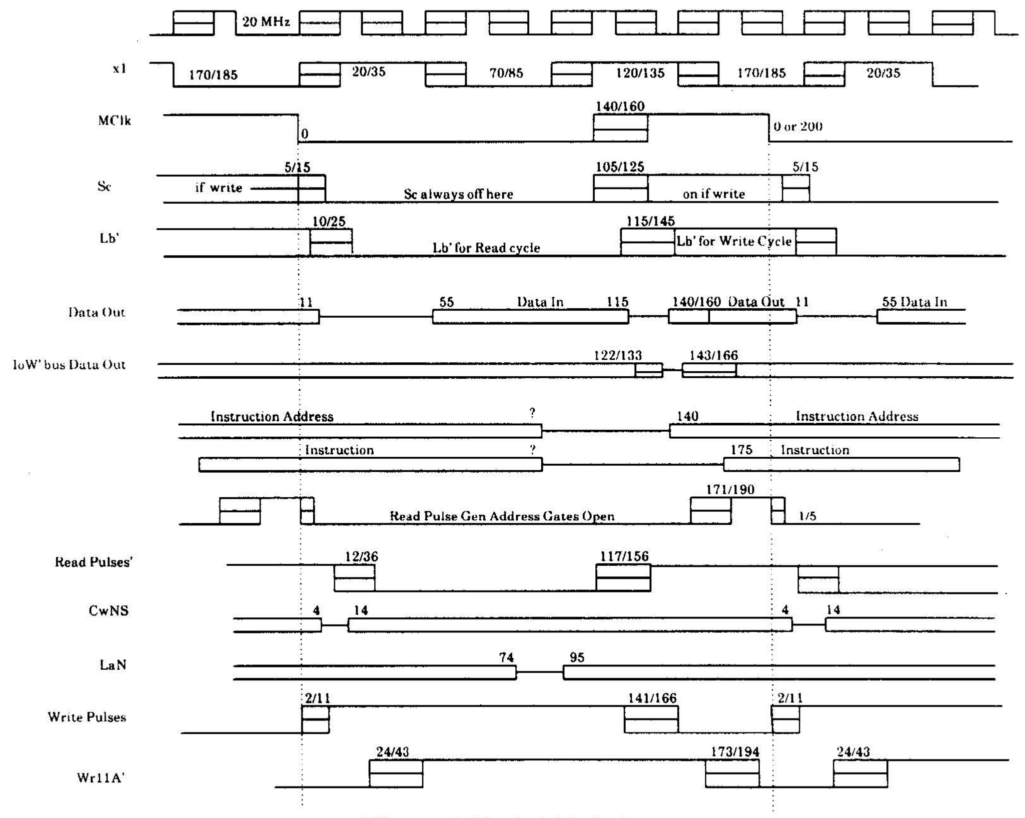


Figure 5.11. 8x305 timing

Figure 5.12 illustrates timing for reading data, data separation, and address mark detection, described in section 5.2.2.6. In the figure, C indicates a clock pulse and D indicates a data pulse. The number below D is the content of the cell; the character x below C in column 21 is a missing clock, which is interpreted as an address mark.

Bits occur at 5 MHz or 200 ns/bit with no variation. A write operation is fixed at 200 ns; a read operation is plus/minus 1% of 200 ns.

Signals in the figure are:

DOUT	- Disk data out
DtaInDlyd	- Data in delayed
VFOClk	- VFO clock (from Phase Locked Loop)
PLLLocked'	- Phase Locked Loop locked
Search	- Search for address mark
DHld	- Separator hold locked in 0
RDi	- Read data within data separator
RDta	- Read data out of separator
RCi	- Read clock within data separator
RCkS	- Read clock out of separator
AmDet	- Address mark detected

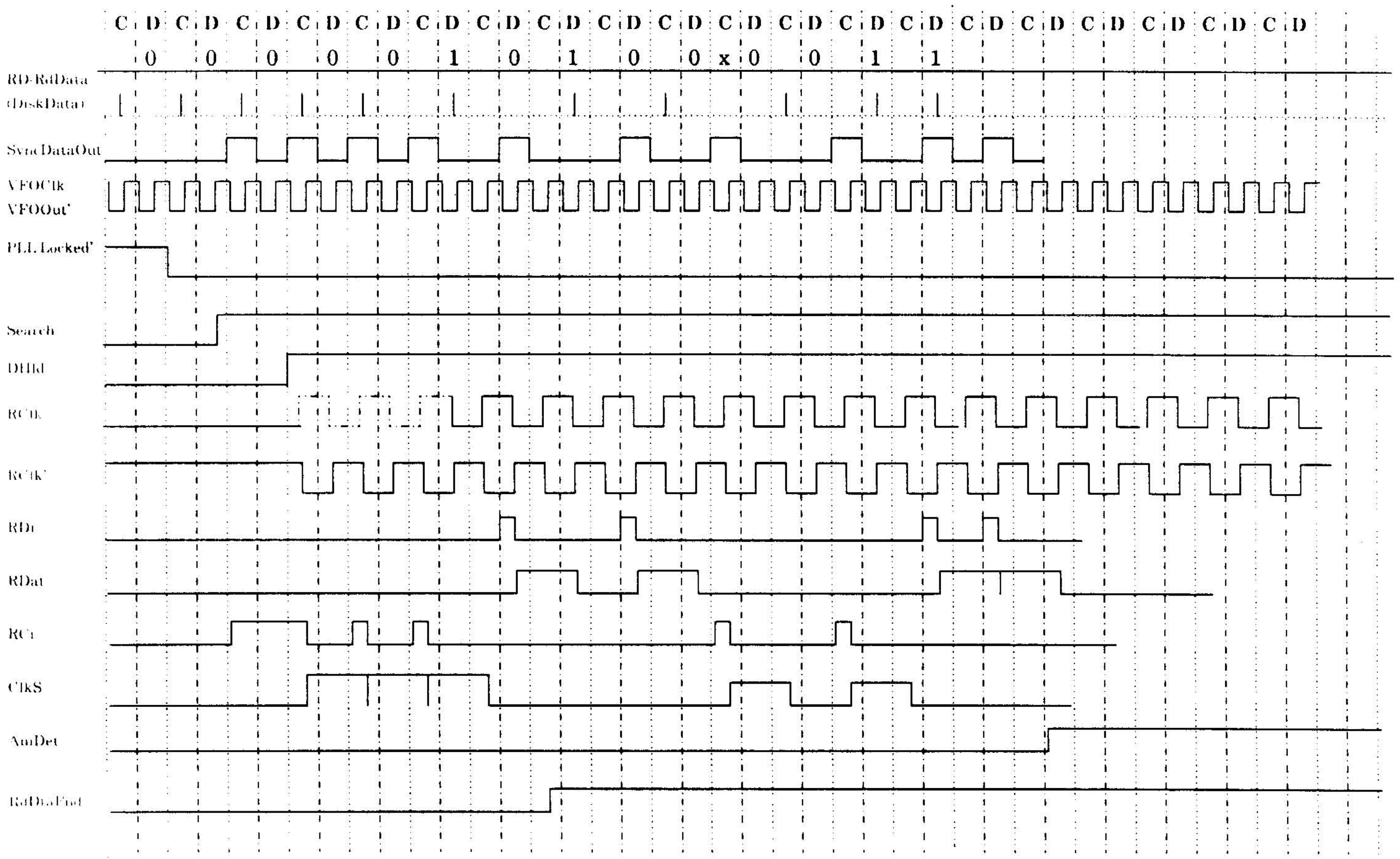

C D C D C D C D C D C D C D C D C D C D C D C D C D C D C D C D C D C D C D C D
0 0 0 0 0 1 0 1 0 0 x 0 0 1 1
RD-RdData
(DiskData)
SyncDataOut
VFOClk
VFOOut'
PLL Locked'
Search
DHld
RClk
RClk'
RDi
RDat
RCi
ClkS
AmDet
RdDtaFnd

Figure 5.12. Timing for read data, data separator, and address mark detection

5.3 DMA Controller

The DMA controller controls high speed data transfers between the FIFO and main memory, and consists of three sections: a control section, a data path section and a third section which is the data and control route by which the IOP programs the DMA and FIFO.

The control section includes a control register, a status register, a state machine, and other control logic required to make the system fully functional. The data path section includes the buffers, registers, and latches necessary to direct the data back and forth between the FIFO and the 80186 bus. The program section includes a buffer and other logic.

Figure 5.13 illustrates the DMA controller.

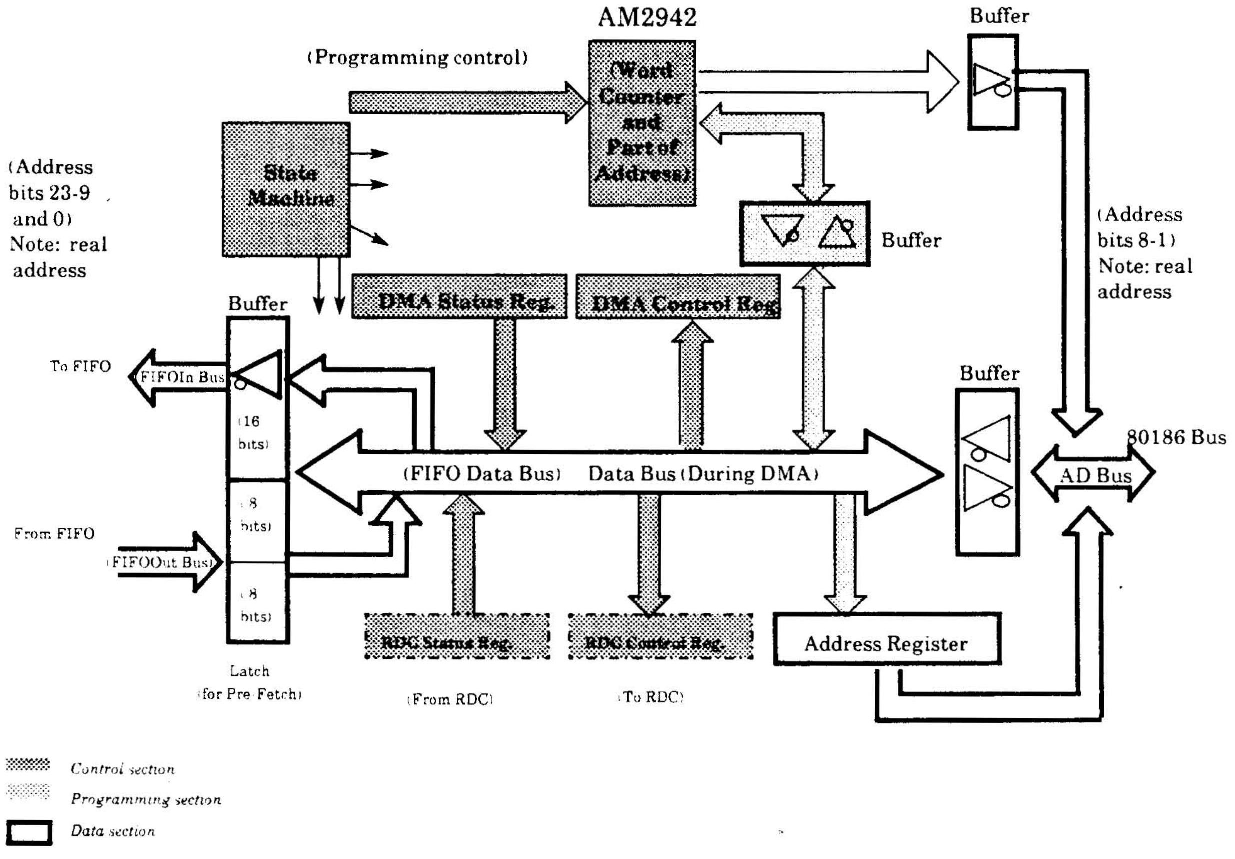


Figure 5.13. DMA controller block diagram

The main features of the DMA controller are:

- is programmed by the IOP (starting address, word count, and transfer direction)
- transfers up to 256 16-bit words at a time
- provides 24 bits of real address to the main memory controller
- transfers data at the full 80186 bus rate (one word per four T-state memory cycle); can also tolerate slower memories by inserting wait states in the memory cycle
- provides an end-of-transfer interrupt to the IOP
- dynamically uses the available 80186 bus cycles

5.3.1 Hardware

The controller for the DMA function is a custom implementation and provides the controlling task as programmed. The controller interfaces to main memory through the 80186 bus and is one of the 80186 bus masters.

5.3.1.1. DMA Signals

Tables 5.5 – 5.8 list and explain significant signals of the DMA and bus interfaces. Refer to these tables when studying the schematics contained in appendix D.

Table 5.5. DMA Signal Description

Symbol	Type	Function
StartDMA'	Output	Active low pulse generated when the IOP issues a StartDMA command.
DMAActive'	Output	Negative true signal that indicates whether or not DMA is active; that is, DMA operation in progress. The signal is cleared (DMA not Active) when DMA operation is temporarily suspended by outside factors, such as Ethernet, a qualified interrupt to IOP, or the IOP itself.
Run SM (Run State Machine)	Output	Goes high in response to IOP's StartDMA request, and allows the DMA to run. The signal remains high until the completion of DMA operation and automatically goes low when the DMA operation is complete. During the time that DMA operation is temporarily suspended by outside factors, the signal maintains its active (high) level, indicating that DMA transfer is not yet finished. This signal is available as one bit of the status register.
EndOfXfer'	Output	Negative true signal indicates that the total number of words transferred between main memory and the FIFO is equal to the number of words originally requested for DMA transfer.
ADDr'/Data	Output	Identifies the type of information on the bus during DMA transfer. Low only during the T1 states of the bus cycle while DMA operation is in progress. Remains high when DMA is not active.

Table 5.6. 80186 Bus Control Interface Pin Description

Symbol	Type	Function
RDCHoldReq	Output	DMA Hold Request (equivalent to HLD in 80186) sent by the RDC DMA to the bus arbiter.
RDCHoldAck	Input	DMA Hold Acknowledge (equivalent to HLDA of 80186) sent by the bus arbiter to the RDC DMA.
S.2'-S.0'	Output	Provides status lines for the 80186 bus, as follows: S2' - S0' — IOP Function 000 — Interrupt Acknowledge 001 — Read I/O 010 — Write I/O 011 — Halt 100 — Instruction Fetch 101 — Read Memory * 110 — Write Memory * 111 — Passive (no action)
186BHE'	Output	Indicates high byte.

* The DMA controller uses these functions and their corresponding s'lines. Other functions are not supported by the DMA controller

Figure 5.7. 80186 Data Bus Interface Pin Description

Symbol	Type	Function
AD.15-AD.00	I/O	Address /Data bus bits 15-00.
A.23-AA.16	Output	Address bus bits 23-16.
IOPARDY	Input	Indicates whether the main memory subsystem requires that additional wait states be introduced in the memory cycle.
ALE	Used indirectly	Latches the address for each memory cycle.

5.3.1.2. DMA as a Peripheral for IOP 80186

The IOP also treats the DMA controller as a peripheral device. Under this condition, dedicated buffers and logic are used to facilitate the DMA programming. Table 5.8 lists the signals used by the IOP when it programs the DMA controller or queries DMA controller status.

Table 5.8. 80186 DMA Program Signals Description

Symbol	Type	Function
Rd'	Input	Read pulse from 80186 indicating that status information is being requested by 80186.
WrL'	Input	Pulse used for writing the low order byte of a control register.
186DEn'	Input	Pulse requesting the data to be made available. (Refer to 80186 specifications.)
RDMASel	Input	One of the Peripheral Chip Select lines that is dedicated to the RDC/FIFO/DMA generated by the IOP.
RDiskDmaIntr	Output	Causes an interrupt signal from the DMA controller to the 80186, and informs the IOP that it has finished the DMA transfer.

5.3.2 Theory of Operations

Included in the control section of the DMA controller is a PROM-based state machine which provides appropriate control signals to the data path section and the FIFO, communicates with main memory and the bus arbiter, and sends interrupts to the IOP.

The DMA state machine also receives external status information, such as FIFO status or end-of-transfer indication.

Sequencing through consecutive addresses as well as maintaining the word count and parts of the address bits are made possible using an AM2942 chip.

Operation details are described in the following subsections after a brief explanation of the state machine structure.

5.3.2.1. DMA State Machine

Figure 5.14 illustrates the state machine hardware. The DMA state machine is PROM-based; the control information is stored in the form of a bit pattern within the PROM. The figure corresponds to the state machine block of the control section in Figure 5.13.

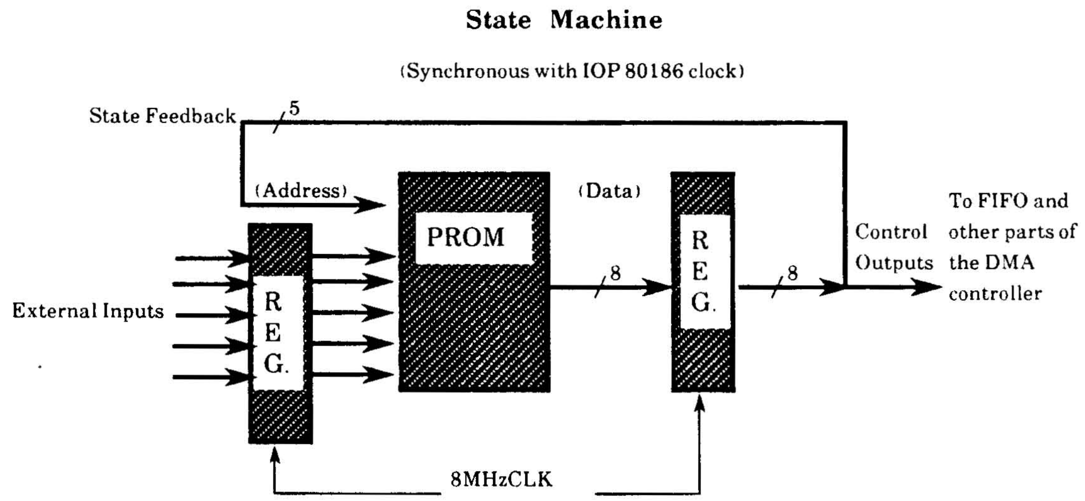


Figure 5.14. Block diagram of DMA state machine

Address inputs to the PROM are signals from different parts of the DMA controller and FIFO that are clocked into a register; they provide a stable address to the PROM for a period of 125 ns. The data vector saved in this location of the PROM is clocked into another register to provide a stable pattern for 125 ns for use as control signals by other parts of the rigid disk subsystem. Some of the data outputs are also fed back as address bits to the same PROM. The state machine marches through the state sequences, as determined by its previous state and by external information.

If an illegal condition is presented to the inputs of the state machine, then the machine creates an error condition. The machine does this by: 1) setting the error bit in the DMA status register; 2) freezing the clock for the status register to preserve the latest status just before the error condition; and 3) sending an interrupt to IOP.

5.3.2.2. States of the State Machine

Figures 5.15 – 5.17 illustrate the three phases of the states of the state machine. The circled numbers in the figures refer to the steps as described in the text.

See Appendix A, Table A.4 for the state machine PROM contents and the relationship between the states and the PROM outputs. Timing for the states is given in section 5.3.2.4.

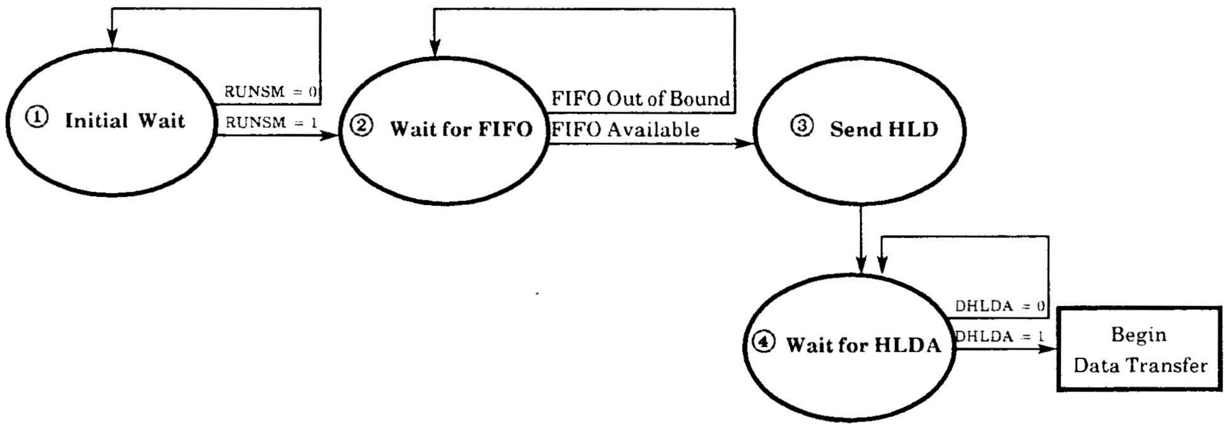


Figure 5.15. DMA states: I - Transfer initiation

1. The state machine normally is in the "initial wait" state. This state occurs at power-up time when the DMA controller is reset, and also occurs after successful completion of a DMA transfer.
2. The StartDMA command from IOP causes the state machine to go to the wait-for-FIFO state, during which FIFO availability for DMA operation is verified.

   If the FIFO is not available, then the DMA waits in the same state until the FIFO is available. The FIFO may be unavailable for one of two reasons:

   FIFO is full AND the transfer direction is memory-to-disk.
   FIFO is empty AND the transfer direction is disk-to-memory.

   If the FIFO is available, then the DMA goes to the next state.
3. A HoldRequest (HLD) is sent to the bus arbiter.
4. The DMA waits for HoldAcknowledge (HLDA). When the acknowledgment is received, the 80186 bus is clear and available

for DMA transfer. The DMA then begins the bus cycles for the transfer.

Figure 5.16 illustrates the bus cycles.

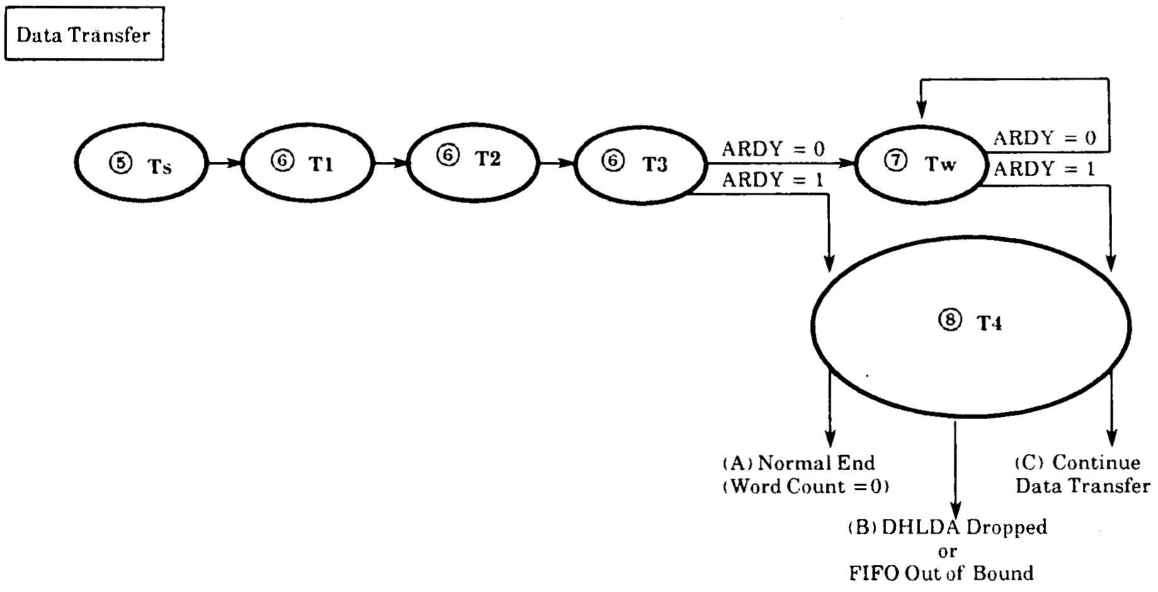

Data Transfer
⑤ Ts
⑥ T1
⑥ T2
⑥ T3
ARDY = 0
ARDY = 1
⑦ Tw
ARDY = 0
ARDY = 1
⑧ T4
(A) Normal End (Word Count =0)
(C) Continue Data Transfer
(B) DHLDA Dropped or FIFO Out of Bound

Figure 5.16. DMA states: II. Bus cycles

5. An initial T state, Ts, is first introduced. Ts is equivalent to the last T state of an ongoing memory access.
6. The DMA goes consecutively to T1, T2 and T3 states, while providing signals according to the 80186 bus protocol for memory access and providing appropriate control signals to the FIFO and address/word count logic. If main memory can complete memory access in four T cycles, then the DMA goes to T4 state (step 8).
7. If main memory needs more time to complete the access, then memory informs the DMA controller by pulling ARDY signal low. In response, the state machine, after T3, goes to a Tw state and introduces Tw cycles until the main memory announces completion of access by pulling the ARDY signal back high. The DMA controller then moves from Tw state to T4 state and completes a one-word transfer.
8. At T4 state, the state machine branches into three possible operating states: A) normal end; and B) and C) more data to be transferred. Figure 5.17 illustrates these states, which are described below. The error state, labeled D on the figure, is also described. The circled number refers to the step number previously described.

   A) If the total number of words requested for DMA transfer has been reached (normal end), then the state machine drops

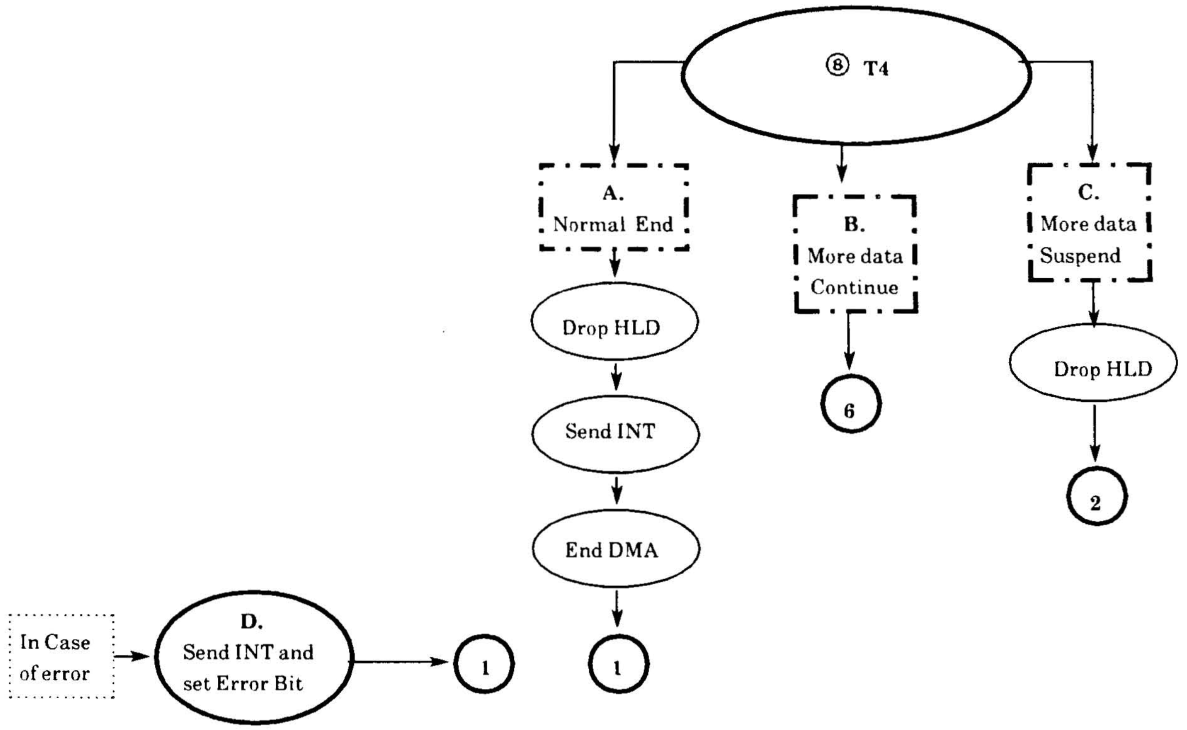


Figure 5.17. DMA states: III. Transfer conclusion

the HLD line, and branches to send an interrupt request to IOP. The branch occurs when an end-of-transfer condition is reached, regardless of other factors, such as unavailable FIFO or other bus master requesting the bus.

B) If more data is to be transferred, and the FIFO is available, and no other bus master is requesting the 80186 bus, then the state machine branches to T1. (Refer to step 6.)

C) If the FIFO is not available for the next word to be transferred or if another 80186 bus master is requesting the bus, then the DMA drops the hold request line (HLD) and relinquishes the 80186 bus. After dropping the HLD line, the DMA goes to the state of waiting for FIFO to become available (refer to step 2).

If the FIFO is available, then, after one more T state (delay through the state described in step 2), DMA moves to SendHLD state to send the hold request again.

D) The DMA state machine is designed around a PROM, but not all of the addressable locations within the PROM are used in implementing the state machine. An error state occurs when the address lines point to an unused portion of the PROM.

When an error state occurs, an error bit is set in the DMA status register, and an interrupt is given to the IOP. The error bit does not disable the state machine. Instead, the state machine goes to the Initial Wait state (step 1) until it receives further directions from the IOP. The status register is frozen until an explicit reset clears it. No further DMA transfers occur until the IOP intervenes, takes recovery measures, and re-issues StartDMA.

For the DMA controller, presence and absence of the HLD and HLDA lines, in addition to the normal meanings under 80186 bus protocol, are a means of communication with the bus arbiter regarding the bus availability. The arbiter communicates the need for the bus evacuation to the DMA controller by dropping HLDA for the DMA controller while DMA is in operation. The controller then leaves the 80186 bus and informs the arbiter of bus availability by dropping the HLD line.

5.3.2.3. Operating Sequence

In a rigid disk DMA operation, one of two possible sequences of operation is exercised. Table 5.9 lists the operating sequence for transferring data from FIFO to memory and for transferring data from memory to FIFO. In the table, actions that are the same in both directions are centered. Actions that are specific for the direction are placed in the appropriate column.

Table 5.9. Data Transfer Operating Sequence

Device	From FIFO to Memory	Same	From Memory to FIFO
IOP		1. Programs the DMA for CR register, word count (two's complement), and starting address (24 bits), not necessarily in the order given.	
	2. Sets direction bit to 0.		2. Sets direction bit to 1.
		3. Issues StartDMA.	
		4. Issues AllowRDC to bus arbiter. Note: IOP will have communicated with RDC regarding the I/O control block and the data transfer between rigid disk and FIFO.	
DMA		5. Checks if FIFO is available for transfer.	
		6. Sends hold request (HLD) and waits for HLDA.	
		7. When HLDA is received, transfers one word per memory cycle, as follows:	
	7a. Moves S'lines active (memory write)		7a. Moves S' lines active (memory read)
		7b. Puts 24-bit address on the bus (AD15-00 and AA23-16) during T1 state.	
	7c. Pre-fetches data from FIFO (during last T4 and present T1)		7c. (no equivalent step)
	7d. Puts data on the 80186 bus during T2 and T3 (plus Tw caused by IOPARDY)		7d. Waits until data from memory is stable on the bus (delays for IOPARDY).
	7e. Data written into main memory at T4 state.		7e. Data written into FIFO at T4 state.
		8. Decrements the word count (increments the two's complement).	
		9. Increments the address (24 bits).	
		10. Transfers another word (during the next T1 through T4).	
		11. If HLDA drops, then DMA drops HLD, delays for two cycles, and sends Hold request again.	
		12. If the FIFO is unavailable, then DMA drops HLD, waits for FIFO available, then sends HLD.	
		13. DMA continues to transfer until End-of-Transfer.	
		14. At End-of-Transfer, sends Interrupt to IOP and returns to ready (initial wait) state.	
IOP		15. Once interrupted, will check for End-of-Transfer or error.	
		16. If IOP is interrupted by other I/O during the DMA, then it issues AllowRDC again. This action is required when ready to continue the DMA transfer.	
		17. After DMA interrupt, IOP has control for the next DMA transfer.	

5.3.2.4. State Machine Timing

DMA timing behavior is explained in Figures 5.18 - 5.26. A description of each diagram precedes the figure.

The timing diagrams explain:

- typical transfers with zero, one, and two wait states
- starting DMA operation
- suspending DMA transfer (and releasing the bus) in response to an Ethernet request or other interrupt to the IOP
- resuming DMA transfer after such a suspension
- delaying and postponing DMA operation in response to unavailability of FIFO
- continuing the operation after FIFO becomes available
- completing DMA operation after all the requested words are transferred

No Wait State

Figure 5.18 illustrates timing for No Wait state. Use this diagram as a reference for the other diagrams as well as for an independent timing chart. The diagram enumerates possible delays in the subsystem when the component variations are taken into account. To better illustrate a starting operation, a Ts state just prior to the first T1 state is also shown.

Delay values are expressed in ns and are measured from the falling (and in a few instances the rising) edge of the ClkOut signal (same as the 8MHzClk on the schematics). S'lines illustrate the S1' or S0' that is low during a memory cycle. If the lines are high, then the signal remains in its default level of high without going to low level.

S'lines and ADBus specify the timing tolerances experienced on the IOP Bus while the DMA controller is the bus master.

ALE is not generated or used by this subsystem, yet is essential for the proper function of the DMA transfer. Its timing in any given system provides enough setup and hold time so that it can be used for latching the address that is given by the DMA controller during the T1 state.

IOPARDY and SARDY (synchronized version of IOPARDY with DMA state machine) are assumed to be high at all times. This naturally yields the no-wait condition.

ADDr'/Data is low during those clock cycles in which the AD bus contains an address.

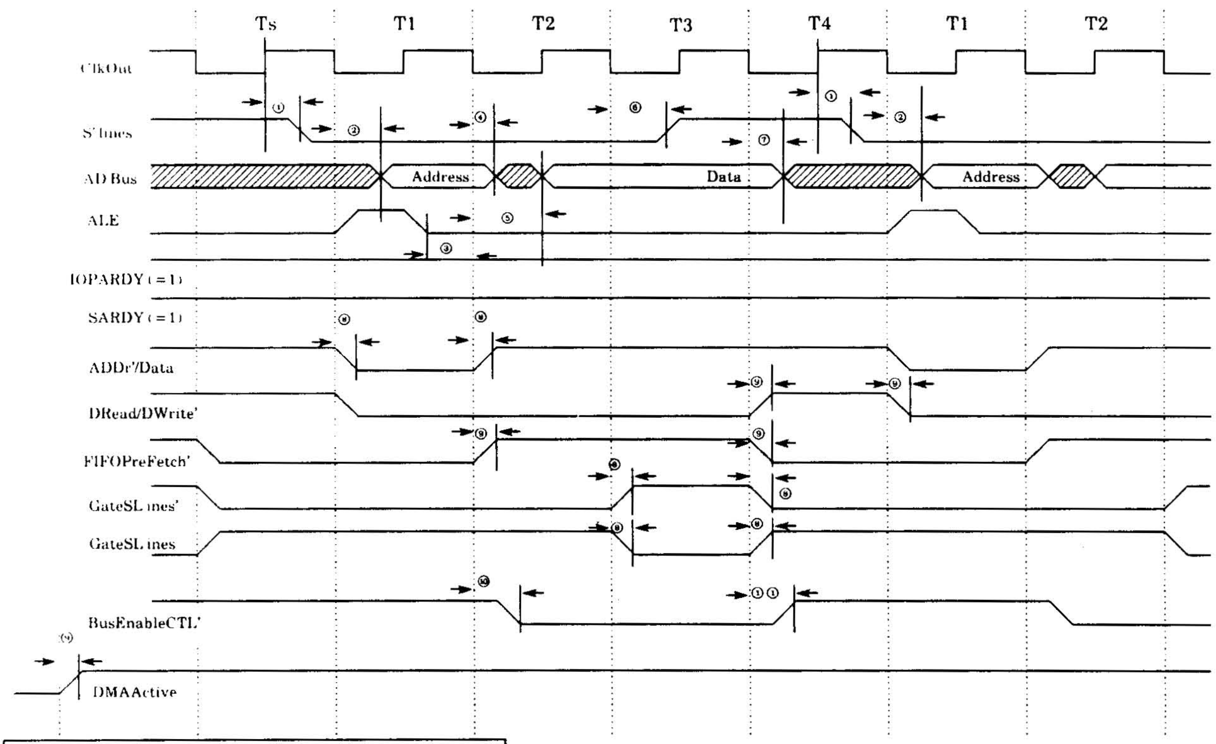
DRead/DWrite' and FIFOPreFetch' are activated by the state machine as specified in the diagram. The signals are generated regardless of the transfer direction, and are only used by logic external to the state machine, as determined by the programmed transfer direction.

GateSLines and its inverted counterpart provide appropriate IOP bus status on the S2', S1' and S0' lines and are also used by other parts of the controller.

BusEnableCTL' enables the AD buffer and delivers the data to or accepts data from the IOP bus, depending on the direction. The signal is generated by the DMA controller while a DMA operation is in progress, and is generated by the IOP 80186 when the IOP attempts to program or query the DMA controller.

Note: The timing specifications on the diagram associated with this signal are valid only during DMA operation and are no longer true when the signal is controlled by the IOP.

DMAActive is a state-machine-generated signal used in various areas of the controller to indicate that a DMA cycle is running.



Key: Minimum/Typical/Maximum in ns. N = not defined.		
① 15/N/87	⑤ 14/N/34.5	⑨ 4/N/20
② N/N/27	⑥ 14/N/85	⑩ 4/N/25
③ 2/N/N	⑦ 10/N/43.5	①① 8/N/37
④ 8/N/N	⑧ 3/N/19	

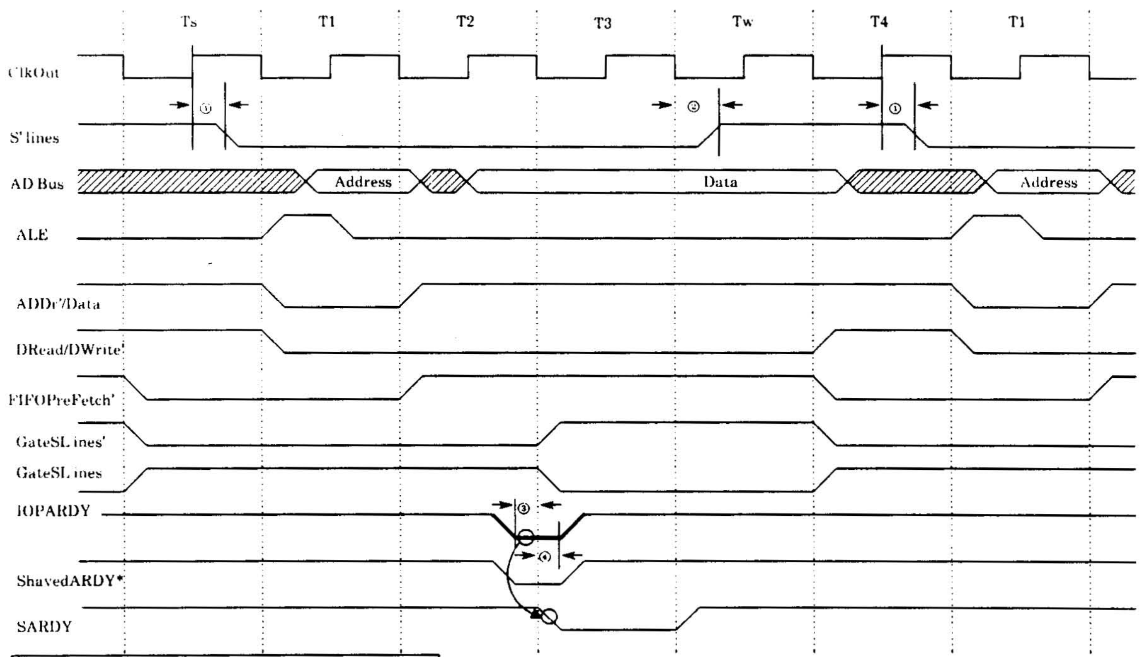
Notes: 1) The signal "S'lines" represents only the signals among S1', S2', and S0' that go low during a memory cycle. (For the case like S2' = 1, obviously the signal remains high.)
2) In all timing values, neither the effect of clock skew nor the delay between ClkOut and ClkOut' is included.

Figure 5.18. DMA timing: No wait states

Figure 5.19 illustrates timing for one wait state. One wait state is similar to no-wait state, except that a narrow pulse on the IOPARDY signal has produced a ShavedARDY signal. Later, ShavedARDY creates a suitable pulse on SARDY. This refined signal is input to the state machine and forces it to introduce one Tw (wait) state between T3 and T4.

Note that in order to create the Tw state, the IOPARDY signal must meet the minimum requirements given on this diagram. IOPARDY can go low earlier than is required (e.g., early in T2 or sometime during T1), without causing a problem and will simply be ignored by the state machine. To achieve one Tw state, the signal must be low at the end of T2 and must be high prior to the second half of the T3 state (see Figure 5.19).

See also section 5.3.2.5 for more timing on IOPARDY behavior and its impact.



Key: Minimum/Typical/Maximum in ns. N = not defined
① 15/N/87 ③ 11/N/N
② 11/N/71 ④ 6/N/N

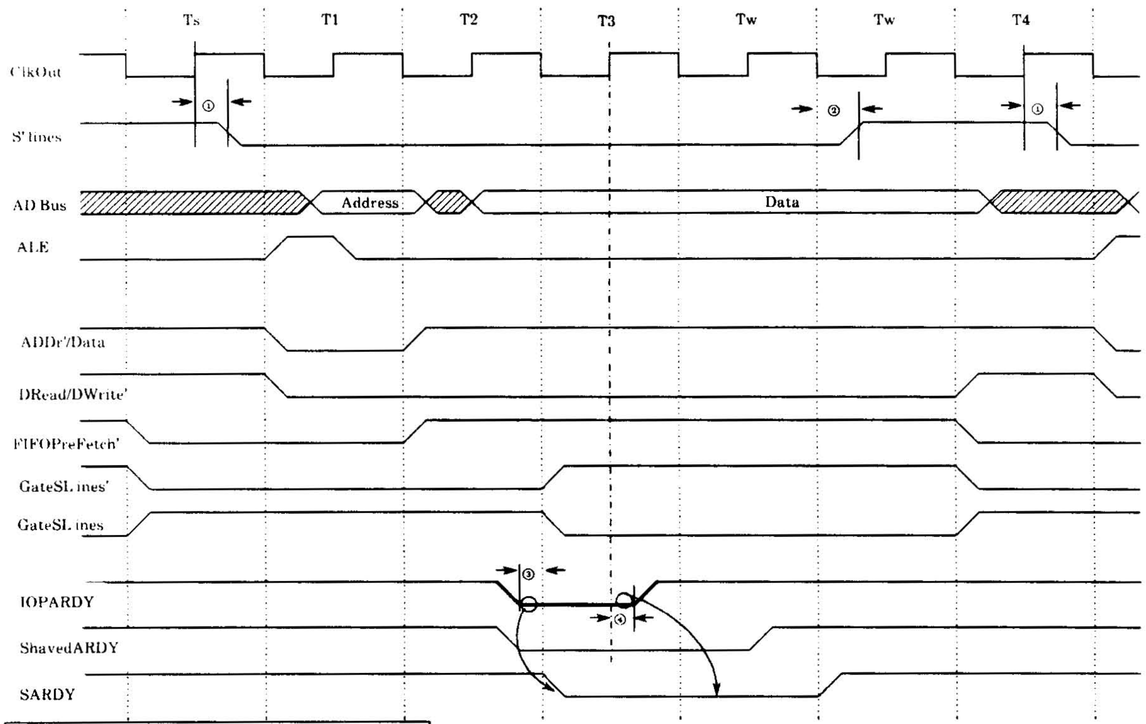
Notes: 1) For other timing information, refer to the No Wait States case.
2) * For further information please refer to schematics, appendix D.

Figure 5.19. DMA timing : One wait state

Figure 5.20 illustrates timing for two wait states. The timing is similar to the two previous figures. Two Tw states are introduced by the length of the negative pulse on IOPARDY signal (which has maintained its low level during the first half of the T3 state).

This pulse in turn has been modified to become a synchronized pulse on the SARDY signal for a period of two T cycles. The state machine responds by introducing two Tw cycles. A pulse longer than these cycles on the IOPARDY signal yields more Tw states accordingly. The number of Tw states that can be introduced has virtually no limit. The memory cycle can be extended for a long period of time. Only when the low level is removed from the IOPARDY signal will the system resume the normal operation as before without loss of data.

See also section 5.3.2.5 for more timing on IOPARDY behavior and its impact.



Key: Minimum/Typical/Maximum in ns. N = not defined
① 15/N/87 ③ 11/N/N
② 11/N/71 ④ 3/N/N

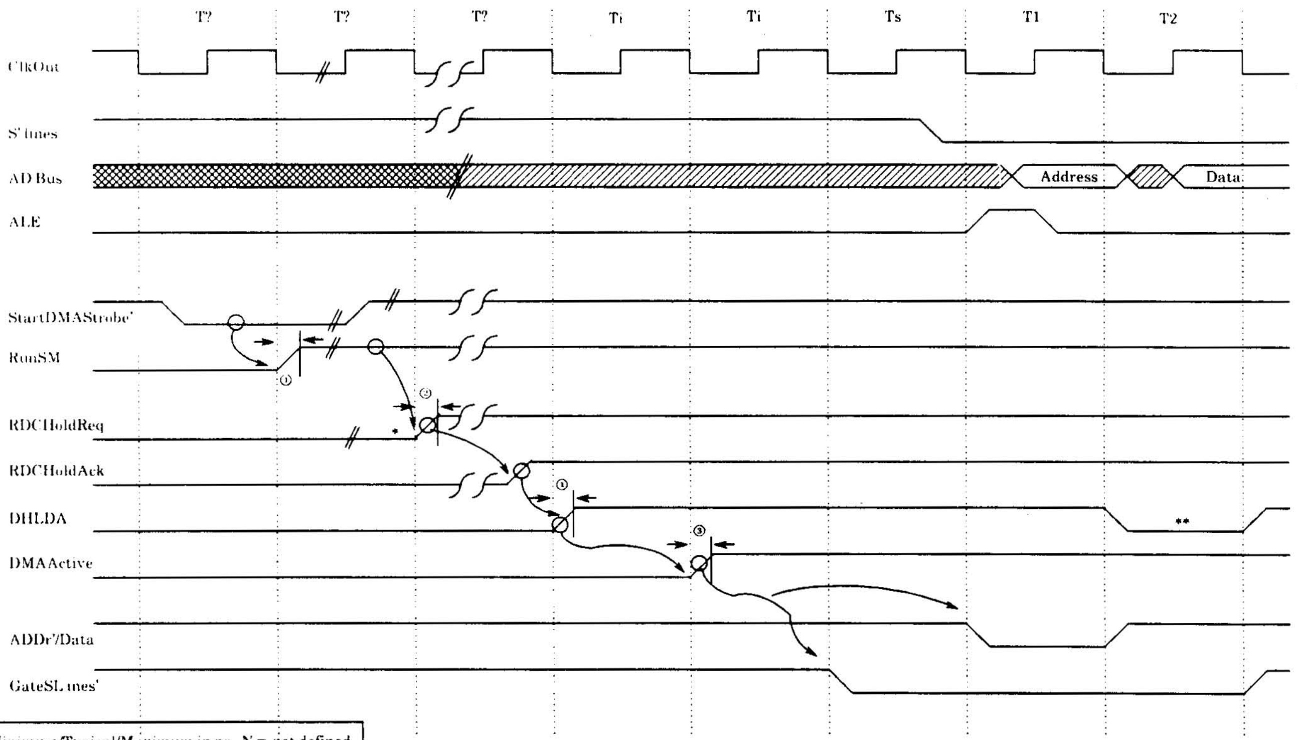
Note: For other timing information, refer to the No Wait States case.

Figure 5.20. DMA timing : Two wait states

Figure 5.21 illustrates starting DMA operation. Issue of a StartDMA command by the IOP causes a negative pulse on StartDMAStrobe' that in turn moves RunSM to a high level.

If the FIFO is available, then the state machine moves RDCHoldReq high to request the bus mastership. After delay for a number of cycles (depending on bus traffic and on the status of the bus masters), the bus arbiter responds by moving RDCHoldAck high.

A refined version of this signal, DHLDA, informs the state machine of IOP bus availability. The state machine initiates the DMA operation activity by moving DMAActive high. The next cycle is a Ts cycle; the following cycles will then proceed as shown in the previous figures.



Key: Minimum/Typical/Maximum in ns. N = not defined
① 3/N/19
② 3/N/19
③ 4/N/20
* Delay if FIFOOB' = 0
** Ignored by State Machine
(Assuming No Wait States case)

Note: For other timing information, refer to the No Wait States case.

Figure 5.21. DMA timing: Starting DMA operation

Figure 5.22 illustrates Ethernet- or interrupt-originated DMA suspension, which results in dropping the RDCHoldReq line.

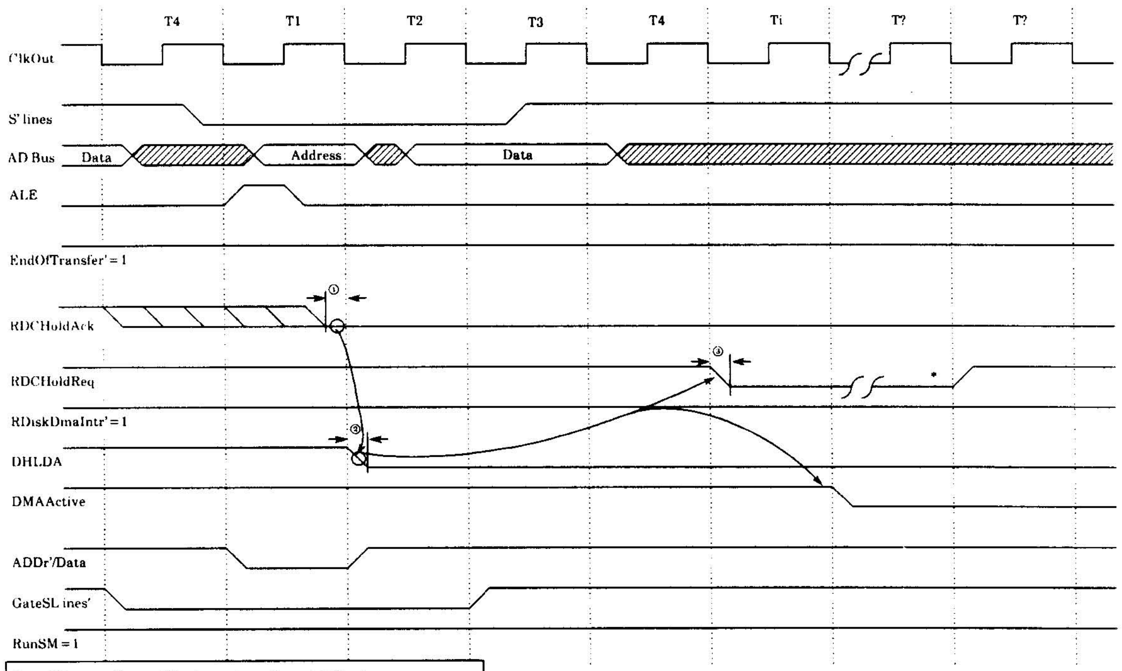
If Ethernet requests the bus mastership while DMA is in operation or if an interrupt requires the IOP's attention to the interrupt source, then the bus arbiter informs the DMA controller by moving RDCHoldAck low, to signal the need to relinquish the bus.

This change is relayed to the state machine as DHLDA. The state machine then proceeds with the current transfer until reaching T4 state. After that, without inserting another bus cycle (note the S'lines), the state machine drops the RDCHoldReq line. One cycle later it clears the DMAActive signal.

The state machine then waits one more T cycle. If the FIFO is available for additional transfers, then the DMA raises the RDCHoldReq and waits for a response from the bus arbiter (without exercising any bus activities).

This delay allows the bus arbiter to determine if the bus is available for another bus master. Reinsertion of RDCHoldReq has no immediate impact on the bus mastership, but instead confirms that the DMA controller can resume its operation as soon as the bus is available again.

Note that while DMA controller leaves the bus, RunSM remains high to inform the system (in particular, the IOP) of the incomplete DMA operation.



Key: Minimum/Typical/Maximum in ns. N = not defined
① 6/N/N
② 3/N/19
③ 3/N/19

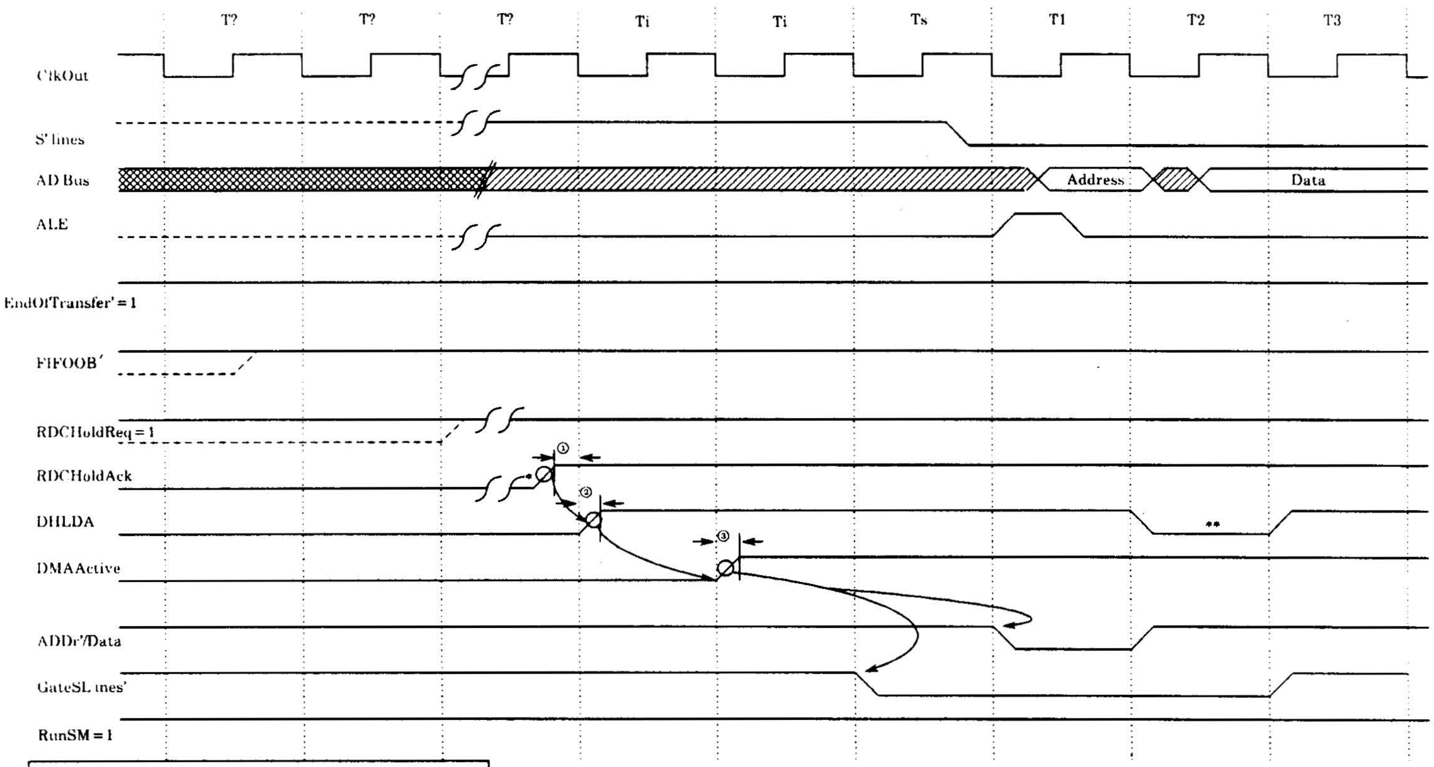
* A minimum of two clock clockcycles for "RDCHoldReq = 0" is guaranteed. Delay beyond that is introduced only if the FIFO is Out of Bound (Assuming No Wait States case)

Note: For other timing information, refer to the No Wait States case.

Figure 5.22. DMA timing: Ethernet- (or any interrupt-) originated Drop Disk Hold request

Figure 5.23 illustrates the continuation of DMA operation after the Ethernet- or interrupt-originated DMA suspension.

At an Ethernet-originated DMA suspension, the DMA controller re-inserts its RDCHoldReq shortly after it leaves the bus (if the FIFO is available) to wait for the acknowledge. As soon as the arbiter responds with RDCHoldAck, and after the other bus master has completed its task and relinquished the bus, the state machine activates the DMAActive signal. A normal DMA transfer is then resumed at the beginning of the next cycle.



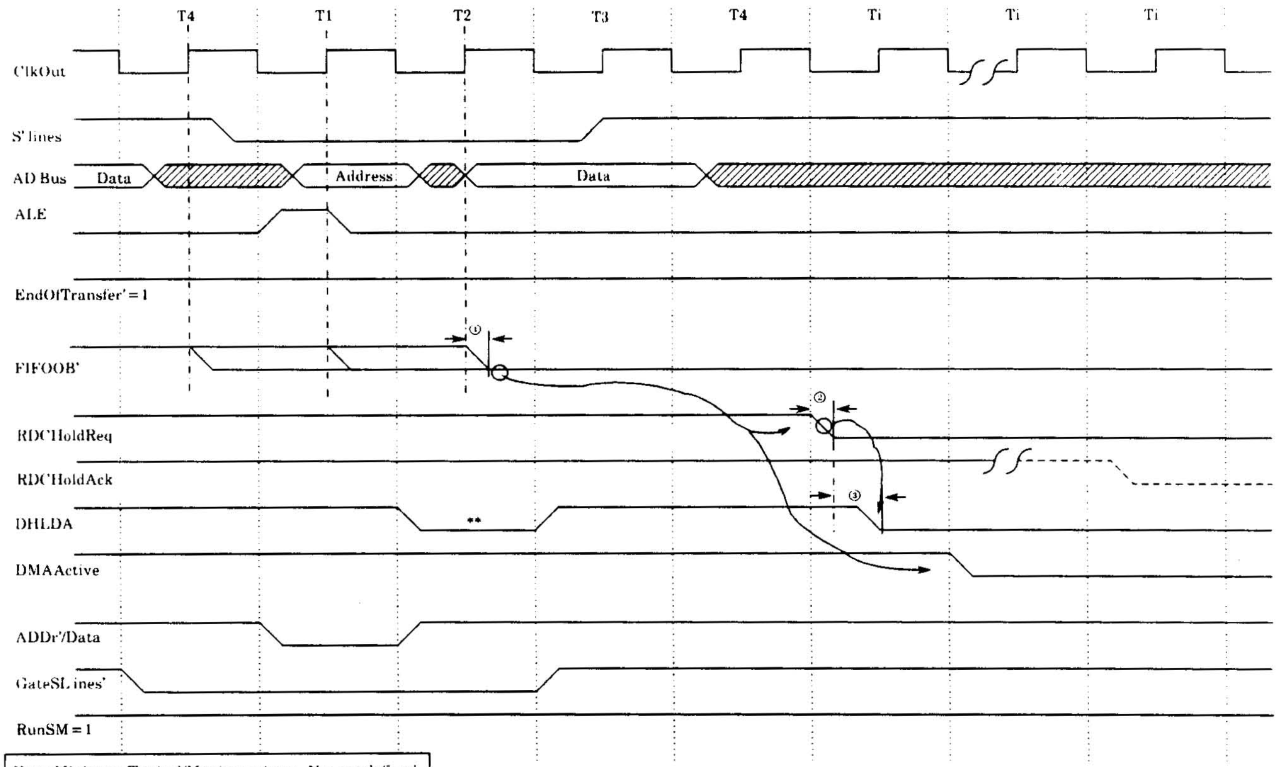
Key: Minimum/Typical/Maximum in ns. N = not defined
① 6/N/N
② 3/N/19
③ 4/N/20

* RDCHoldAck goes High only after Ethernet leaves the bus and "AllowRDC" command is given by IOP (see Arbiter logic)
** Ignored by State Machine
(Assuming No Wait States case)

Note: For other timing information, refer to the No Wait States case.

Figure 5.23. DMA timing: Continuing DMA operation after Ethernet- (or interrupt-) suspension

Figure 5.24 illustrates a FIFO-originated (FIFOOB) DMA suspension, which results in dropping the RDCHoldReq line. In the event the FIFO is not available (FIFOOB' = 0) due to possible delays from the rigid disk, the DMA controller does not keep the IOP bus occupied. Instead, the DMA controller relinquishes the bus for use by other bus masters. The sequence is similar to the previous sequence (Figure 5.22), except that the next RDCHoldReq does not occur until after the FIFO is indeed available and a successful transfer involving the FIFO can be achieved.



Key: Minimum/Typical/Maximum in ns. N = not defined
① 3/N/17
② 3/N/19
③ 3/N/11

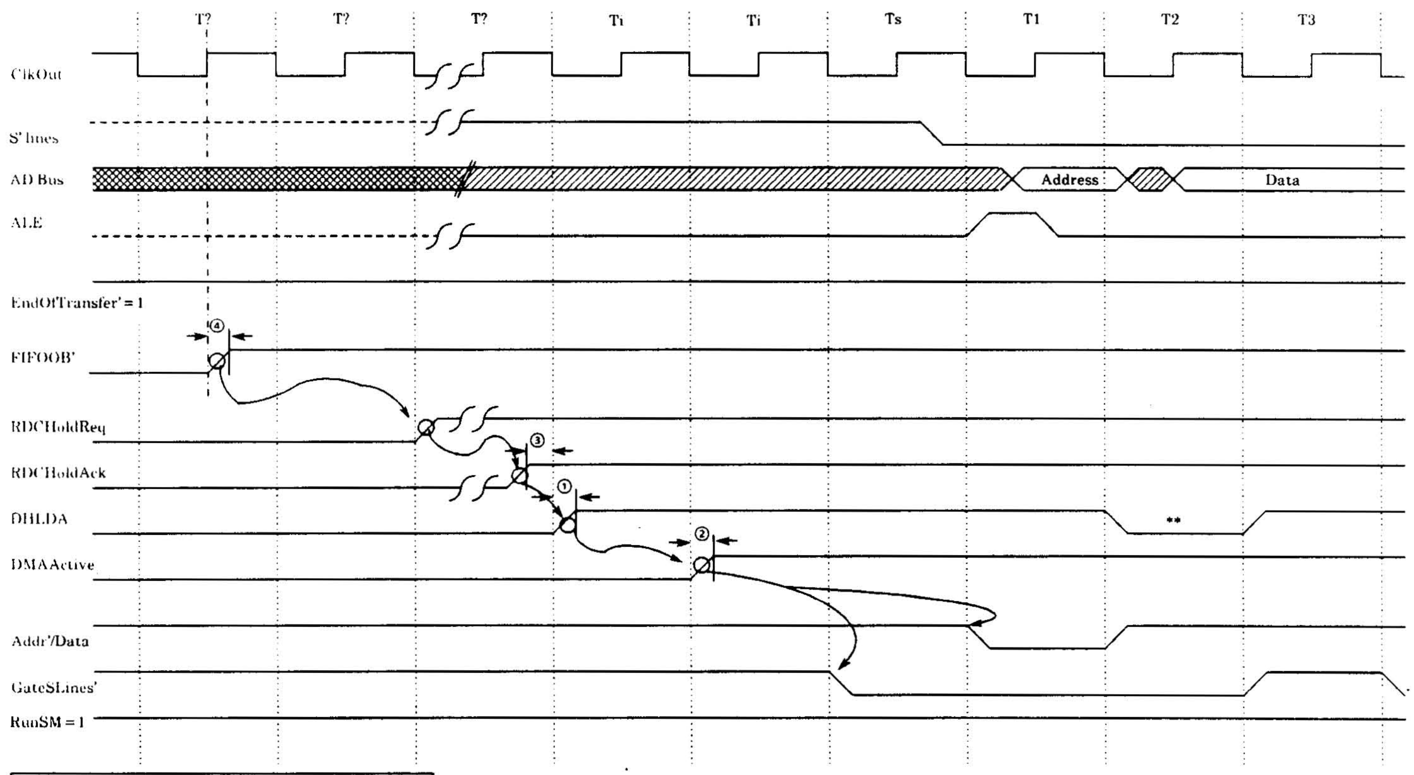
** Ignored by State Machine
(Assuming No Wait States case)

Note: For other timing information, refer to the No Wait States case.

Figure 5.24. DMA timing: FIFOOB Drop Disk Hold request

Figure 5.25 illustrates the continuation of DMA transfer after a FIFO-originated (FIFOOB) DMA suspension. The sequence is much like that shown in Figure 5.23.

In response to the FIFO available (FIFOOB'=1), the state machine moves RDCHoldReq high and, as before, waits for RDCHoldAck.



Key: Minimum/Typical/Maximum in ns. N = not defined

① 3/N/19	③ 6/N/N
② 4/N/20	④ 3/N/17

** Ignored by State Machine
(Assuming No Wait States case)

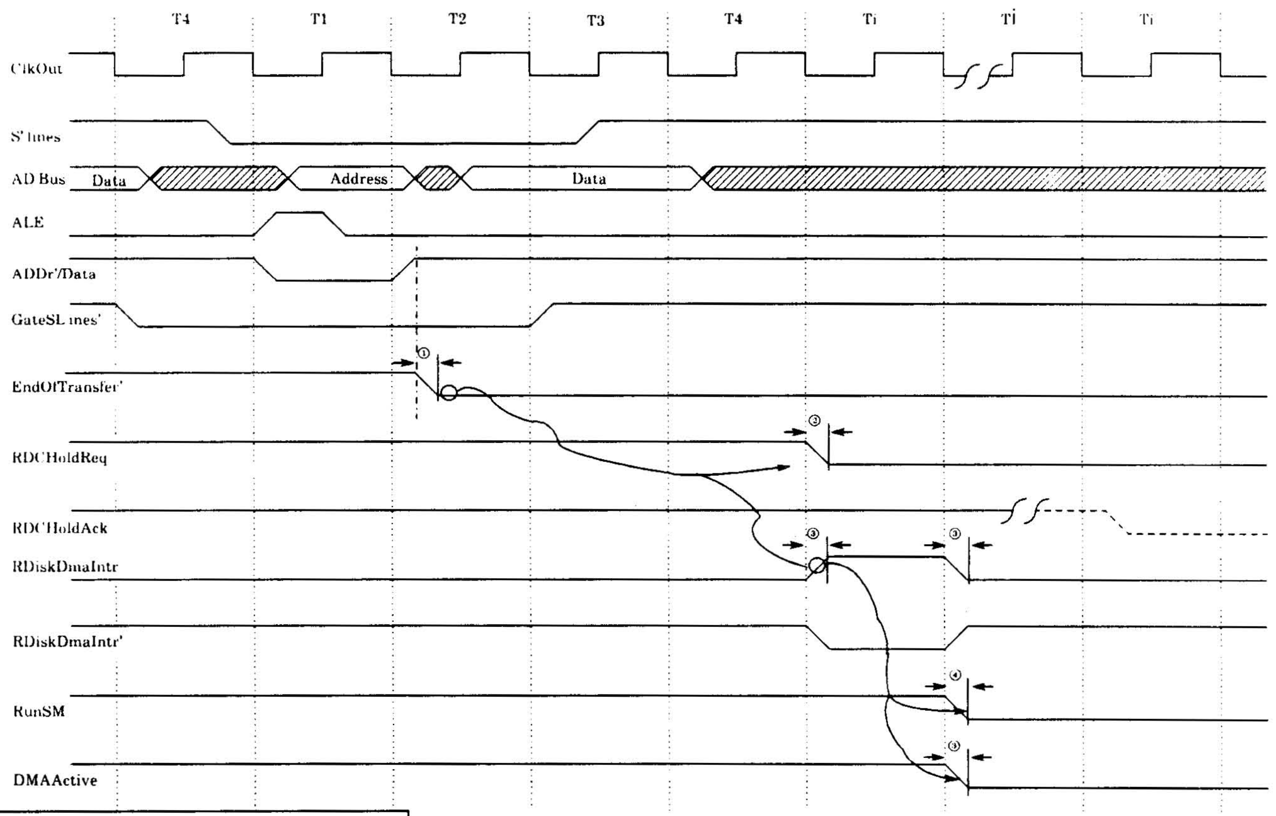
Note: For other timing information, refer to the No Wait States case.

Figure 5.25. DMA timing: FIFOOB continuing DMA operation

Figure 5.26 illustrates timing for ending DMA operation.

The end of DMA operation is signalled to the state machine as well as to the system by the signal EndOfTransfer'. The signal going low indicates that the number of words transferred is equal to the number of words programmed (Word Count).

The transition from high to low on this signal causes the state machine to complete the DMA transfer. The combination of interrupt to the IOP (RDiskDmaIntr' = 0) and RunSM = 0 is the unique feature of this time interval.



Key: Minimum/Typical/Maximum in ns. N = not defined
① 7/N/28 ③ 4/N/20
② 3/N/19 ④ 3/N/19
(Assuming No Wait States case)

Note: For other timing information, refer to the No Wait States case.

Figure 5.26. DMA timing: Ending DMA operation

5.3.2.5. Wait States for Slower Memories

During DMA transfers, the DMA controller performs the 80186 bus handshaking as bus master. It responds to timing requirements of slower memories by inserting Tw states. To do so, it accepts all three behaviors of the ARDY signal (here called IOPARDY) that could be exercised by slow memories on an 80186 bus.

The three ARDY behaviors are:

1. IOPARDY being low during low-to-high of the clock in T2 cycle (and extended n-1 clock periods for n Tw states when $n \geq 2$)
2. IOPARDY being low during high-to-low of the clock for T3 cycle (and extended n-1 clock periods for n Tw states when $n \geq 2$)
3. IOPARDY maintains low until one and a half clocks prior to T4 cycle.

Figures 5.27 - 5.32 illustrate these behaviors of IOPARDY in two systems: Daybreak and Daisy (A chip). The figures are as follows:

Figure 5.27.	Daybreak, Alt. I: One wait state (assumes two memory cycles)
Figure 5.28.	Daybreak, Alt. II: One wait state (assumes two memory cycles)
Figure 5.29.	Daybreak, Alt I: Five wait states to depict a large number of wait states (assumes only one memory cycle)
Figure 5.30.	Daybreak, Alt. II: Five wait states (assumes DMA for only one memory cycle)
Figure 5.31.	A chip (Daisy): One wait state (assumes DMA for only two memory cycles)
Figure 5.32.	A chip (Daisy): Five wait states (assumes DMA leaves the bus after one memory cycle)

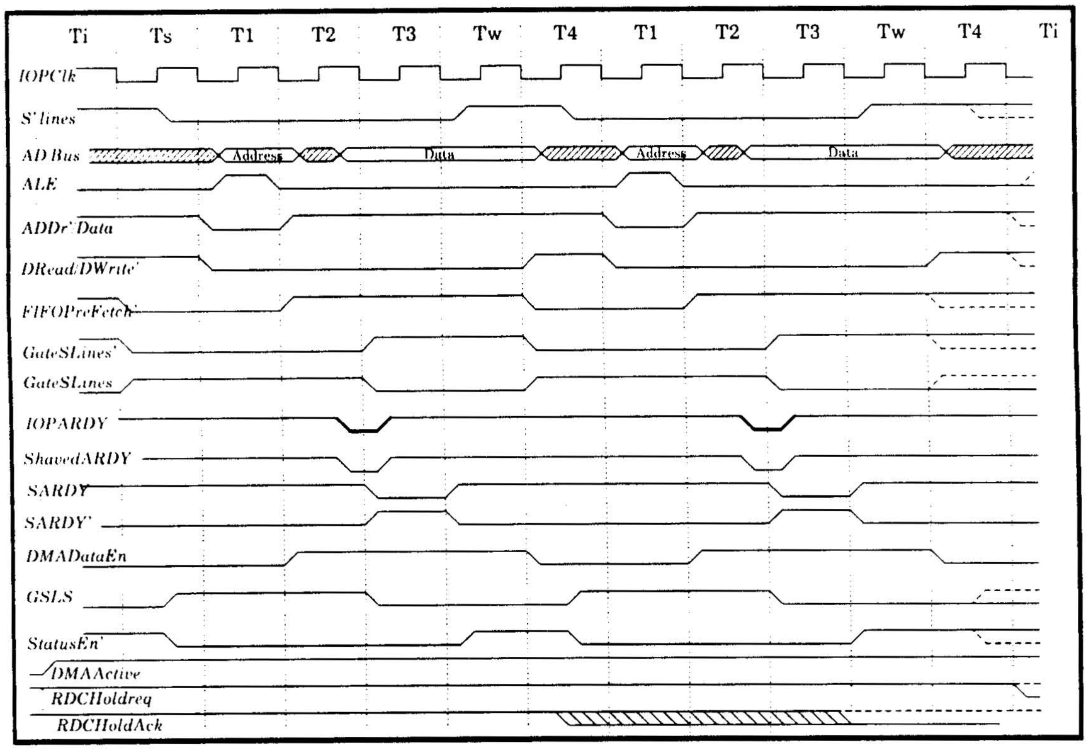


Figure 5.27. IOPARDY timing: Daybreak, Alt. I
One wait state (assumes only two memory cycles)

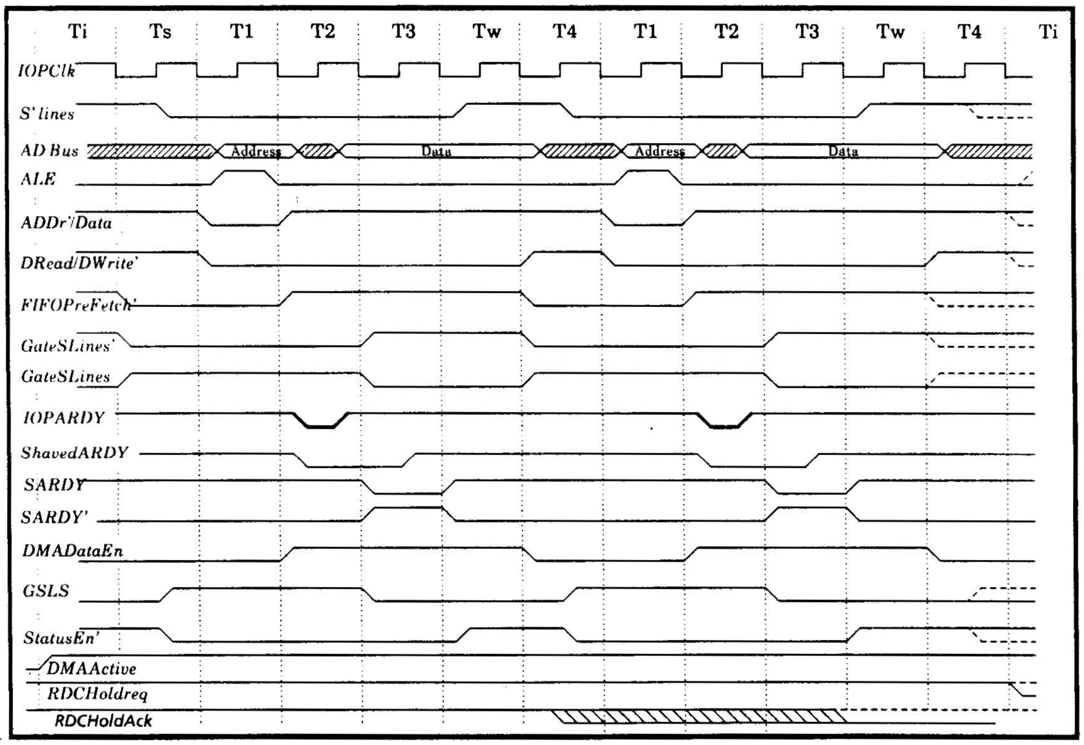


Figure 5.28. IOPARDY timing: Daybreak, Alt. II
One wait state (assumes only two memory cycles)

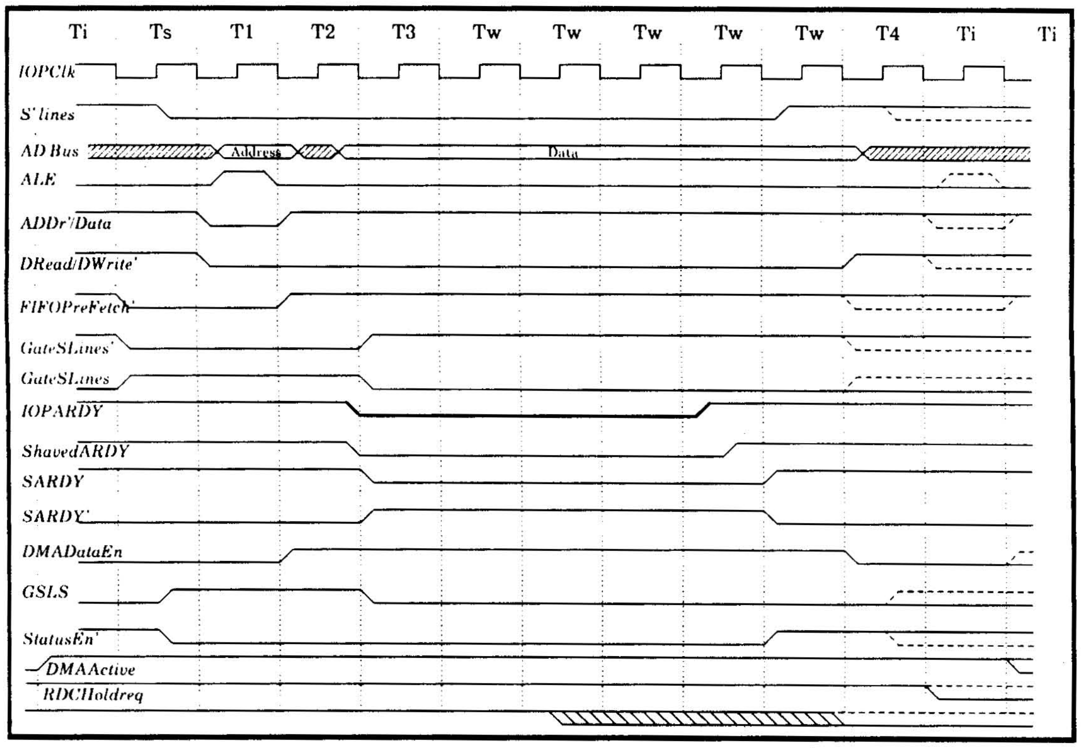


Figure 5.29. IOPARDY timing: Daybreak, Alt I
Five wait states (assumes only one memory cycle)

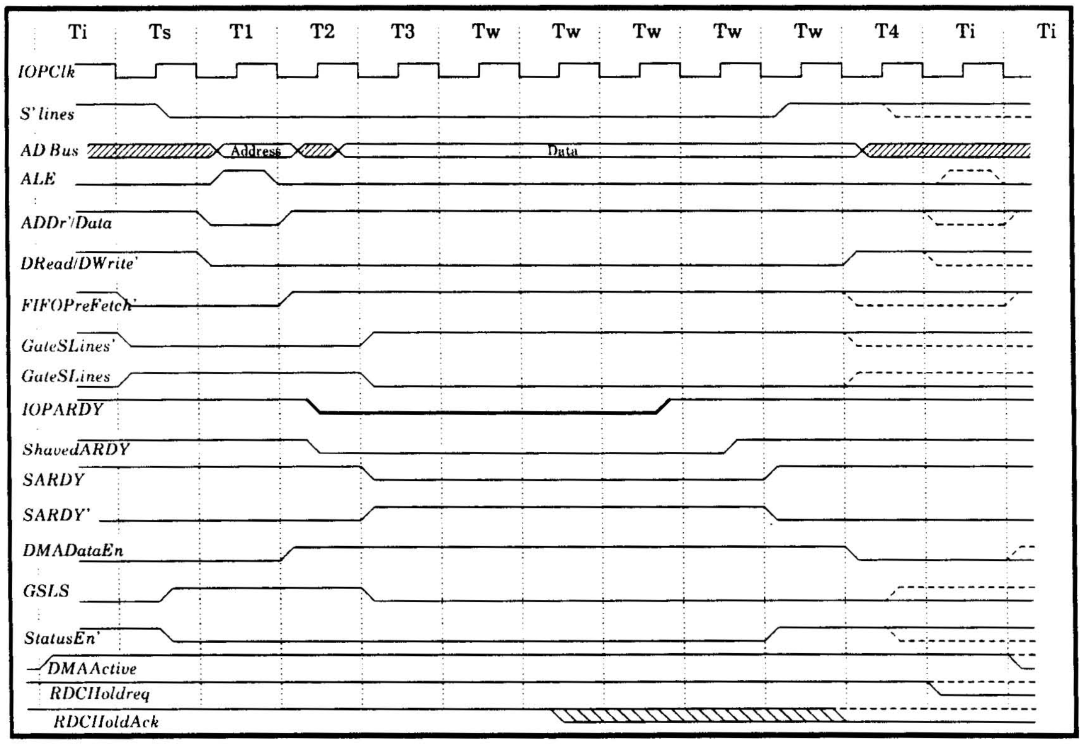


Figure 5.30. IOPARDY timing: Daybreak, Alt. II
Five wait states (assumes DMA for only one memory cycle)

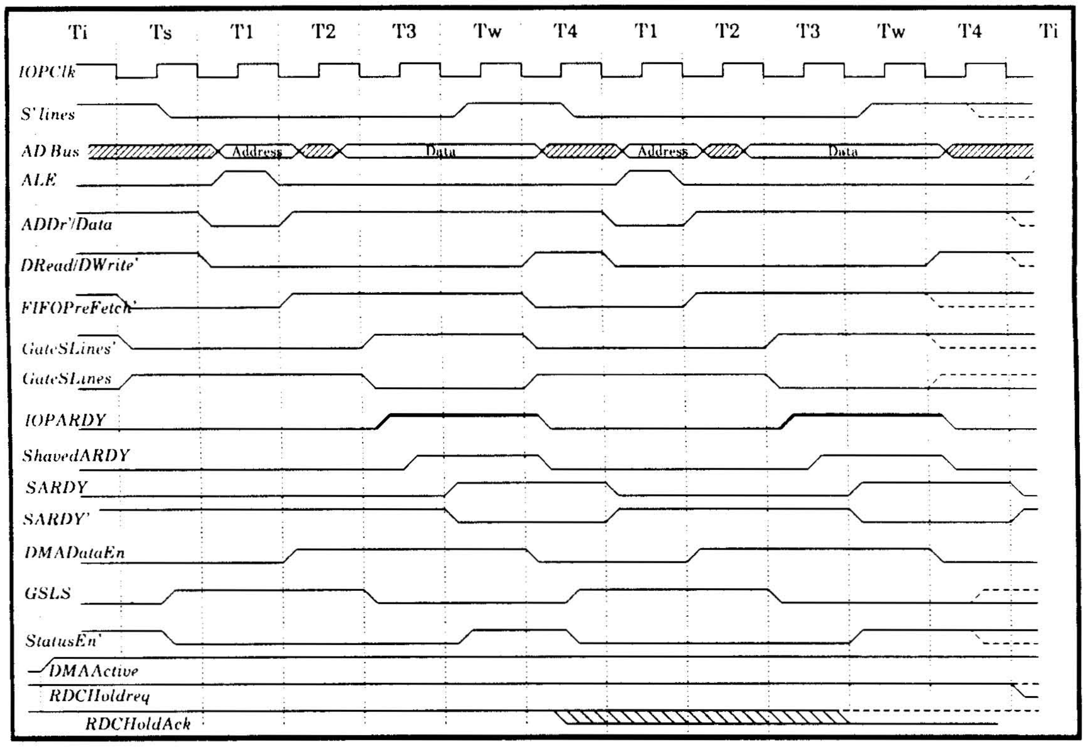


Figure 5.31. IOPARDY timing: A chip (Daisy)
One wait state (assumes DMA for only two memory cycles)

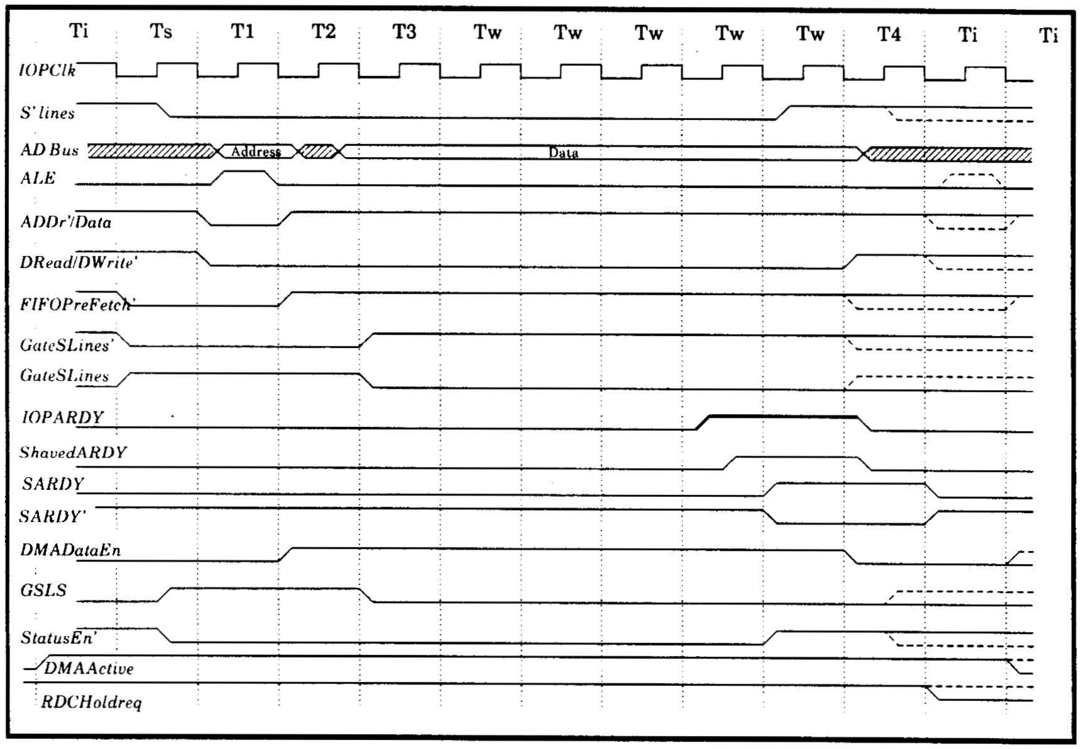


Figure 5.32. IOPARDY timing: A chip (Daisy)
Five wait states (assumes DMA leaves the bus after one memory cycle)

5.3.3 Programmer Interface

The DMA is programmable by the IOP. Registers are accessed as I/O ports of the 80186 and reside within its I/O address space. Some I/O operations perform more than one function. Since not all of the registers have read and write access, the IOP uses the "in" and "out" instructions in accessing them.

The registers that relate to the DMA function are:

- Starting address register – DMA address bits 8-1.
  Starting address register – DMA address bits 23-9 and bit 0.
- Current address register – DMA Address bits 8-1. The register contents chang during DMA transfer to provide a current address.

Note: The higher significant bits of address (bits 23-9) remain unchanged during a maximum of 256 word transfers. Transfers are designed in software to be at page boundaries. If fewer than 256 words are transferred, then the transfers are within the same page.

Note: The DMA address bit 00 for all DMA transfers must be 0. Rigid disk and DMA related address/data/control/status are at word boundaries.

- Word count register - Keeps count of the number of words remaining to be transferred.
- Word count shadow register - Maintains the original value given as word count; that is, number of words to be transferred in one DMA operation.
- Control register within AM2942 (CR2-CR0) - Used by the IOP to program the functioning mode of the AM2942.

The IOP uses four registers within the RDC/FIFO/DMA to communicate with the rigid disk subsystem. They are:

- DMA command register
- DMA status register
- Rigid disk controller command register (described in section 5.2.3.1)
- Rigid disk controller status register (described in section 5.2.3.1)

Table 5.10 summarizes the I/O addresses.

Table 5.10. I/O Addresses

I/O Address	Description
0200H	Write CR2-CR0 (inside AM2942)
0202H	Read CR2-CR0 (inside AM2942)
0204H	Read word count (in two's complement form) (inside AM2942)
0206H	Read address counter bits 8-1 (inside AM2942)
0208H	Load starting address bits 23-9 (and 0) and re-initialize counters
020AH	Load starting address bits 8-1 (inside AM2942)
020CH	Load word count (two's complement) (inside AM2942)
020EH	Enable counters (inside 2942) -- Not used
0210H	DMA command/status registers (Write = command reg, Read = status reg)
0212H	Preset FIFOIn17(Write) (for Diagnostics)
0214H	Rigid disk controller's control/status registers (Write = control reg, Read = status reg)
0216H	Start DMA (Write)
0218H	Not Used
021AH	Not Used
021CH	Not Used
021EH	Not Used

5.3.3.1. Address Register

Figure 5.33 illustrates the 24-bit starting address register.

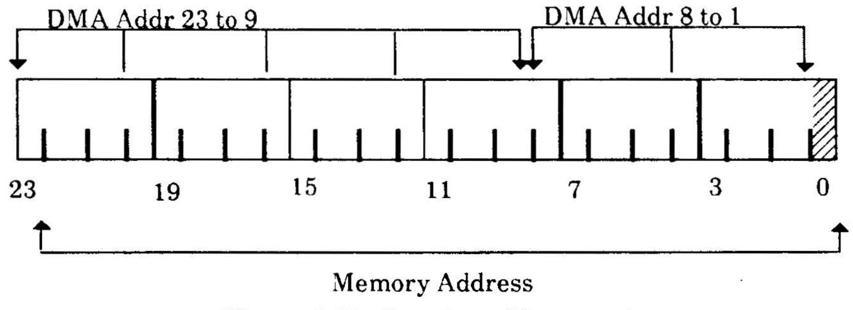


Figure 5.33. Starting address register

The 24-bit address is given by two I/O write commands:

Bits 8-1 = > Write to address 020AH while data is given on bits 8-1 of the data bus. Bit 00 of data bus is ignored and can be 0; that is, a word address can be given without a shift.

Bits 23-9 = > Write to address 0208H while data is given on bits 15-1 of the data bus. Bit 15 of the data is forwarded as the most significant bit (bit 23), and bit 1 will appear as bit 9 of the 24-bit starting address register.

Note: The instruction for writing bits 23-9 of starting address register will automatically re-initialize both the word counter and the starting address register bits 8-1. Their current value, if different from the original, is changed and becomes equal to the original.

Whenever DMA is not the bus master, IOP can read the current word count and part of the current address.

Only bits 8-1 of the latest address are accessible by the IOP. Reading is done by an I/O read "In" function from 0206H. Bits 8-1 of the current address will be delivered on bits 8-1 of the AD bus as data. The user should ignore bit 00 of the AD bus and bits 15-9. Address register bits 23-9 are not accessible.

5.3.3.2. Word Count

The word count is 8 bits wide. A maximum of 256 words can be transferred in one DMA operation. The count should be given over the AD bus bits 8-1, shifted left by one, in the form of two's complement of the actual number of words to be transferred. Table 5.11 lists word count examples.

Table 5.11 Word Count Examples

To transfer...	Load the word count register with...	This means: write...
256 words	00H	0000H
1 word	FFH	01FEH
2 words	FEH	01FDH

An I/O write command "Out" to address 020CH, with the word count on the data bus as specified above, implements the write function.

An I/O write command "In" to address 0204H results in the word count being placed on the data bus by the DMA controller as specified above.

5.3.3.3. Control Register on AM2942 Chip

The AM2942 chip within DMA is used in function mode 3; mode 3 requires the word count register and word counter to be loaded with the two's complement of the number of data words to be transferred.

An I/O write to address 0200H loads bits 3-1 of the data bus into bits 2-0 of the AM2942 control register. Because the chip is used in mode 3, the contents of CR register must be CR2-0 = 011. This translates to writing 0006H to the I/O port; note the shift to the left by one bit.

A read from this register may be done by an I/O read from address 0202H. Contents of the register are provided on bits 3-1 of the data bus; other bits should be ignored.

Note: For details on programming the starting address and word count, as well as for mode 3 specifics, please refer to the AM2942 specifications.

5.3.3.4. DMA Command Register

The DMA command register is a write-only register at I/O address 0210H. Bits 01 through 15 are not used; bit 00 indicates FIFO direction.

Figure 5.34 illustrates the DMA command register.

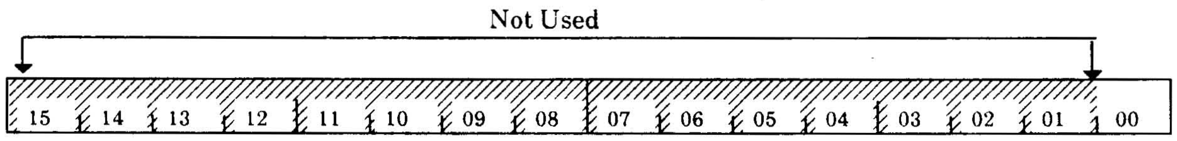


Figure 5.34. DMA command register (Read only I/O Addr = 0210 hex)

where:

Value = 1 is the direction from main memory to disk.
Value = 0 is from disk to main memory.

Note: After each reset exercised on the DMA controller, the transfer direction defaults to disk-to-memory. If the alternate direction is desired, it must be programmed prior to activating DMA operation.

5.3.3.5. DMA Status Register

The DMA status register is a read-only register at I/O address 0210H. Bits 08 through 15 are not used.

Figure 5.35 illustrates the DMA status register.

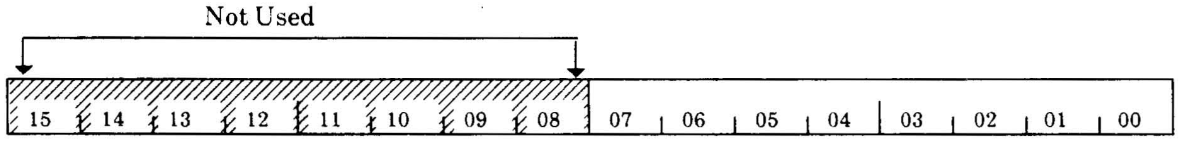


Figure 5.35. DMA status register (Read only I/O Addr = 0210 hex)

where:

Bit 00	- Error The bit normally is 0, but becomes 1 when the DMA state machine encounters an error condition. In such a case, all other bits of this register maintain the information held just prior to occurrence of the error. The bit remains 1 until a reset DMA occurs. (Bit 09 of reset control register in IOP, Table 2.5 = C0H)
Bit 01	- RunSM The bit is 1 when DMA has not completed a DMA transfer; that is, when a DMA operation is in progress. After a DMA reset, (Bit 09 of reset control register C0H) or an end-of-transfer, this bit is 0. Only a StartDMA command will change the bit to 1.
Bit 02	- EndOfXfer' The bit is 0 when a DMA transfer is completed and DMA logic has no other request pending. After DMA reset, this bit has no meaning until the first StartDMA command is issued.
Bit 03	- FIFOOutOfBound' The bit is 0 if FIFO is found empty and the FIFO is in a disk-to-memory direction, or if FIFO is full and is in memory-to-disk direction.
Bit 04	- FIFOEmpty' The bit is 0 if FIFO is empty.
Bit 05	- FIFOFull' The bit is 0 if FIFO is full.
Bit 06	- FIFOOut17 This bit is the output of the FIFO for a status bit carried along with data through the FIFO. This output has meaning only when compared with FIFOIn17, the input to FIFO for the same bit. In addition to the internal use of FIFOIn17and FIFOOut17 by the disk controller during DMA transfer, these two bits can be used to monitor transport of data words through the FIFO.
Bit 07	- FIFODIR The loopback of the direction bit from DMA command register; the bit can be used to verify the direction of transfer later in the programming cycle or during diagnostics.

5.3.3.6 Programming DMA Transfers

Programming DMA transfers begins by setting the AM2942 chip to mode 3. This step is done by writing 0006H to address 200H in the CR2-CR0 control register.

Next, the AM2942 address register A8-A1 and the page point register are set up. The direction bit in the DMA command register is also set up to notify the DMA controller that the data is going either from memory to FIFO or from FIFO to memory.

When the preceding steps are complete, AllowDMA and AllowRDC commands are issued by the IOP. AllowDMA starts the DMA but does not let it run. AllowRDC flags the arbiter for control of the bus. The IOP then waits for a DMA interrupt and then verifies the status of the DMA controller register.

When a DMA transfer occurs, the AM2942 word count register is programmed to indicate a word count between 1 and 256. The data is expressed in bits 8-1 on the IOP data bus. Note that the count is moved left by one bit, and is in the form of two's complement for the number of words to be transferred. Refer also to Table 5.11.

A step-by-step sequence of events for a DMA operation is given in Table 5.9.

5.4 Rigid Disk FIFO

To further isolate the processor/main memory subsystem from the inherent latency characteristics of the rigid disk, a temporary storage is provided in the form of a First-In First-Out (FIFO) buffer between main memory and the rigid disk. The temporary storage transfers data or disk control blocks (DCB) between the rigid disk/rigid disk controller at one end and main memory at the other end.

The main features of the FIFO are:

- Sizeable data buffering capability (512 words long = 2 Mesa pages = 2 disk sectors), which minimizes the impact of disk latencies on Ethernet and DMA operations
- Programmable bi-directional access (memory-to-disk and disk-to-memory)
- Simultaneous and asynchronous access by DMA controller and rigid disk controller
- Full, half-full, and empty indications
- Reset capability at power-up and under program control

Figure 5.36 illustrates the rigid disk FIFO.

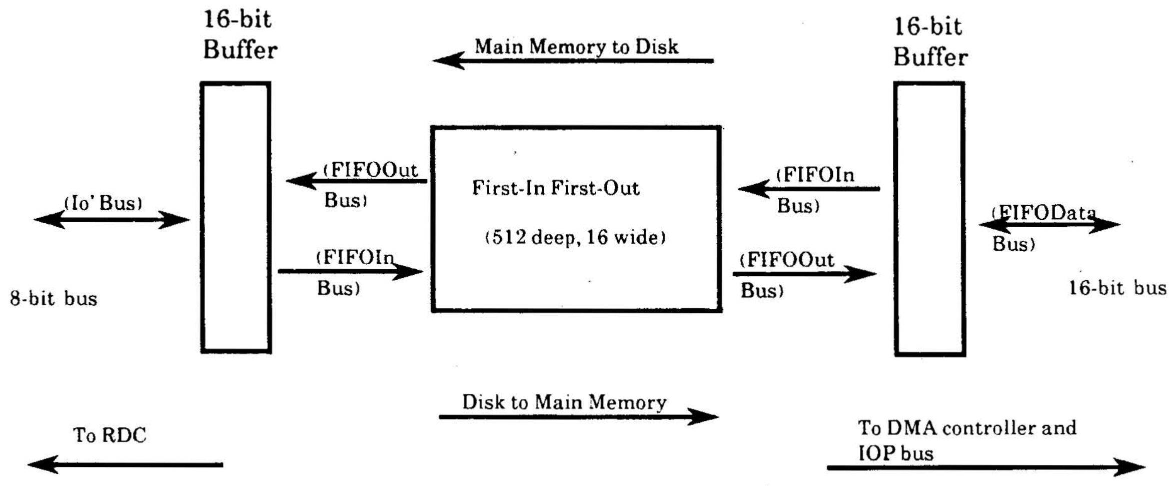


Figure 5.36. FIFO block diagram

5.4.1 Hardware

The FIFO consists of two 512 x 9 components that arrange a 512 x 18 structure, yielding a 512 x 16 logical FIFO block, plus one bit for FIFOIn17/FIFOOut17, and one unused bit. Additional logic provides the half-full capability and appropriate buffering at both ends such that data is received by the rigid disk controller in byte quantities and received by DMA and by the 80186 bus in word quantities.

Two signals that affect the FIFO are described below:

FIFODIR	Programmable by IOP, indicates the direction of DMA transfer: 1 = memory to disk 0 = disk to memory
FIFOOutOfBound	Depending on DMA transfer direction (FIFODIR), sets a flag when the FIFO is not capable of servicing the DMA operation; that is, when FIFO is full and direction is from memory to disk <u>or</u> when FIFO is empty and direction is from disk to memory.

5.4.2 Theory of Operations

Depending on FIFODIR (direction bit), either the DMA controller or rigid disk controller may write into the FIFO; the alternate source is able to read from it. The contents are readable in the same order as written. FIFO status is available to both sources. IOP software ensures that the direction bit is correct and that it does not change until a data transfer is successfully completed from the origin to destination.

After the IOP programs the DMA controller, the controller responds to a StartDMA and initiates a DMA transfer in the programmed direction. The DMA begins with the word at the location given by the starting address, and transfers words with consecutive addresses up to the total number in the word count. Transfer occurs at the rate of one word per memory cycle at the memory speed. The DMA transfer may be interrupted according to the bus arbiter's programmed priority scheme. The transfer resumes from the same point where interrupted and proceeds either to the end of DMA or until the next interruption.

For the DMA controller, transfer in the disk-to-memory direction is subject to availability of data in the FIFO. Transfer in the memory-to-disk direction is subject to availability of storage space in the FIFO. If storage space is not available during the transfer, then the DMA controller automatically relinquishes the 80186 bus and waits until FIFO is available again. The DMA controller then requests control of the bus. After control is granted, DMA transfer resumes from the point at which it was suspended. The process is described in detail in section 5.3.2.

The rigid disk controller and DMA circuitries operate asynchronously and independently. No direct communication facilities exist between them, and therefore they do not exchange information about the contents of the FIFO; that is, whether the contents are I/O control block or actual data. System software ensures data integrity, using the process information and the hardware facilities (rigid disk controller and DMA controller) at both ends of the FIFO.

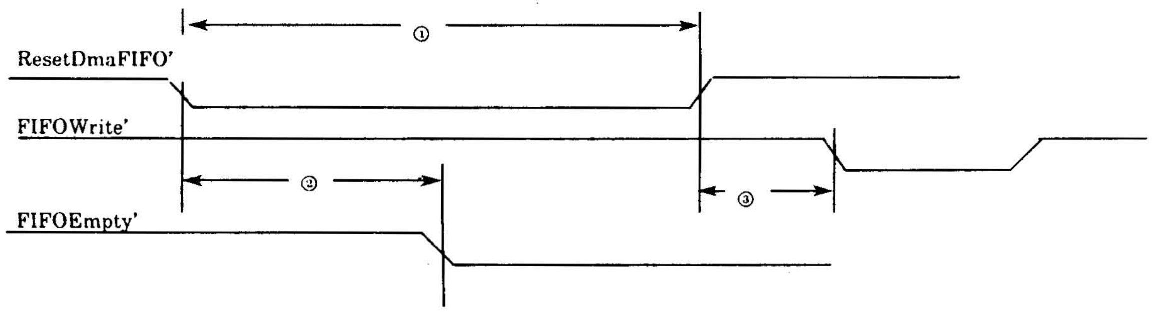
Since FIFO contents are treated in a first-in, first-out manner, particular attention should be paid to the case of reading from the disk. For example, when a DCB is sent to the rigid disk controller, the direction will be memory to disk. Then, in order for data to be transferred from the disk via FIFO to the main memory, the direction must be changed: disk to memory. Next, in order for memory to receive the updated DCB from the rigid disk controller, the direction bit must remain disk to memory.

5.4.3 Programmer Interface: Timing

The components used for the FIFO implementation of this system allow asynchronous access to both ports of the FIFO (input and output). If the timing requirements reflected on the following diagrams are met, then accesses may also occur simultaneously.

Figures 5.37 – 5.40 illustrate timing for the FIFO.

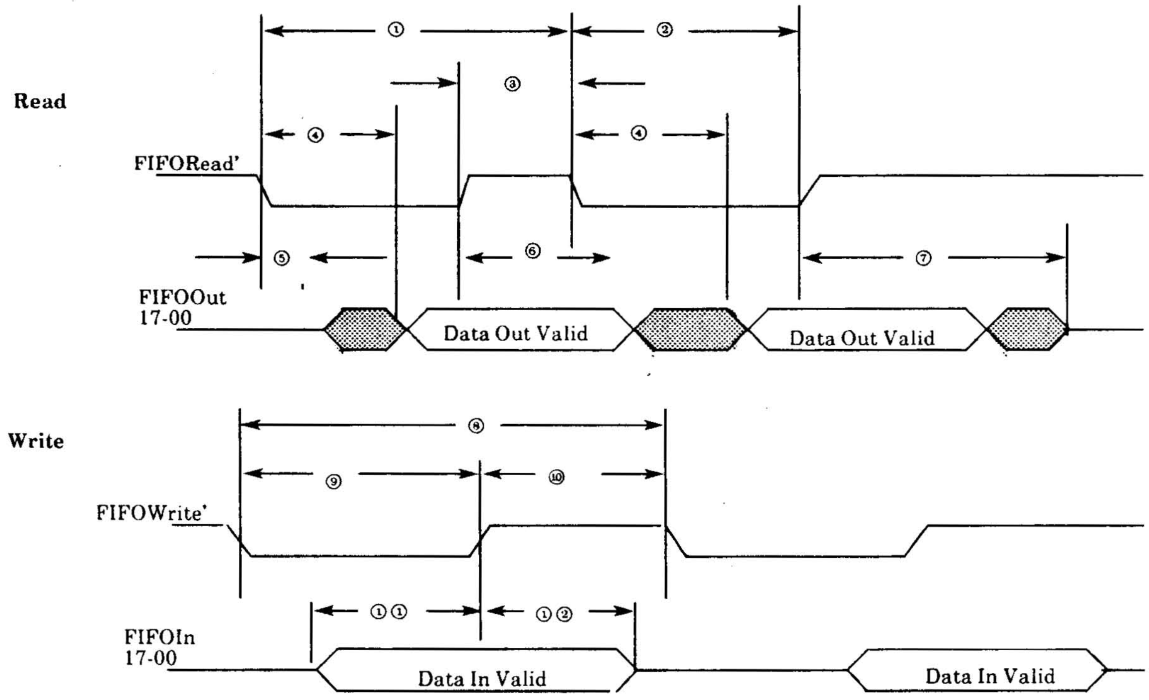
Figure 5.37 illustrates Empty Flag timing with respect to the reset condition. In order for an empty or full flag to be activated by the FIFO, certain minimum delays between the last read and the first write (for empty) and the last write and first read (for full) must occur, as shown in figures 5.39 and 5.40. Two consecutive accesses on one end allow enough time for such a flag to register. However, the system will continue to function even with no flag signal generated, because the rigid disk controller and the DMA controller operate asynchronously at each end of the FIFO. No interruption in the transfer process occurs until one side delays long enough to allow the other side to make more than one access during the delay period. To guarantee the correct operation, two consecutive read or write operations should have a minimum distance from each other and minimum pulse width. (See Figures 5.38 and 5.39.)



Key: Minimum/Typical/Maximum in ns. N = not defined

① 120/N/N
② N/N/140
③ 20/N/N

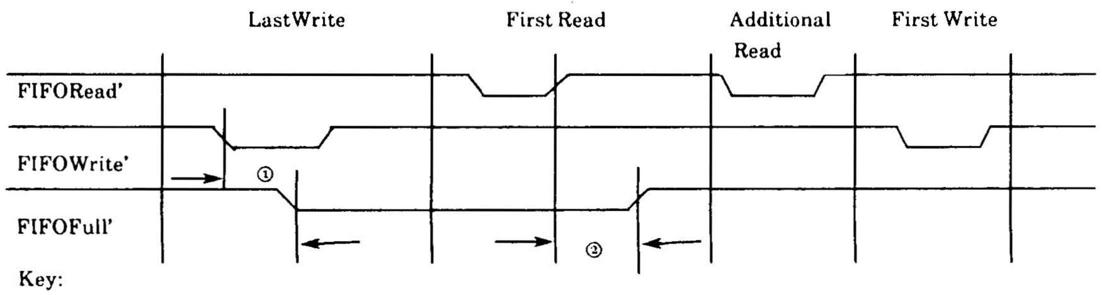
Figure 5.37. Reset timing



Key: Minimum/Typical/Maximum in ns. N = not defined

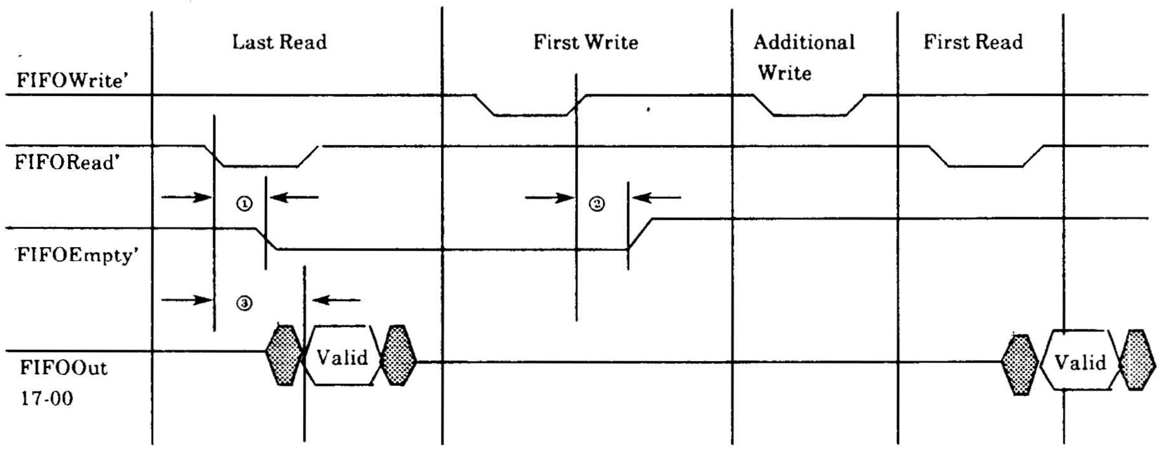
① 140/N/N	⑤ 20/N/N	⑨ 120/N/N
② 120/N/N	⑥ 5/N/N	⑩ 20/N/N
③ 20/N/N	⑦ N/N/35	①① 40/N/N
④ N/N/120	⑧ 140/N/N	①② 10/N/N

Figure 5.38. Asynchronous read/write timing



Key:
① N/N/115
② N/N/110

Figure 5. 39. Full flag from last write to first read



Key:
① N/N/115
② N/N/110
③ N/N/120

Figure 5.40. Empty flag from last read to first write