COP820CJ,COP840CJ,COP880C,COP884BC, COP888CF,COP888CL,COP888EK,COP888FH, COP888GW,COP8ACC5,COP8AME9,COP8CBE9, COP8CBR9,COP8CCE9,COP8CCR9,COP8CDR9, COP8SAA7,COP8SAC7,COP8SBR9,COP8SCR9, COP8SDR9,COP8SGE5,COP8SGE7,COP8SGG5, COP8SGH5,COP8SGK5,COP8SGR5,COP8SGR7, COP912C

AN-413 Disk Interface Design Guide and Users Manual

Literature Number: SNOA710

Disk Interface Design Guide and Users Manual

National Semiconductor
Application Note 413
James Cecil
Ramachandran Gopalan
William Llewellyn
Shakeel Masood
Pat Tucci
Larry Wakeman
January 1986

CHAPTER 1 DISK DRIVE TECHNOLOGY—OVERVIEW

1.0 INTRODUCTION—WINCHESTER DRIVES

From the start, digital computers have required some form of data storage as an adjunct to their relatively sparse main-storage facilities. Some of the early forms of storage were punched cards, paper tape and the magnetic tape storage. This was the principal storage medium, until faster-transfer, higher-capacity media became available and a direct link was established between the computer's main memory and the mass storage device. This link was the rotating memories, commonly referred to as disks.

Disk technology started a quarter-century ago, with the introduction of a large cumbersome fixed disk unit with 50 rotating surfaces 24″ in diameter, a single read/write head assembly, 600 ms seek time and a modest capacity of 5 megabytes. Half a decade later, capacities had increased by tenfold. Multiple head assemblies, one for each surface, introduced the concept of a ''cylinder'', providing simultaneous access to multiple tracks, one above the other, with a single head movement. Packing densities increased, resulting in increased storage capacity up to 100 megabytes. Head designs became more sophisticated; bits per inch increased by an order of 10; tracks per inch doubled.

Contamination-free Winchester technology was introduced by IBM in 1973. In addition to a controlled environment that eliminated dust collection on the disk surface, Winchester innovations included lightly loaded heads, an oriented iron-oxide coating to support higher flux reversal densities, and a silicone or wax coating that permitted heads to slide directly on the surface during ''takeoff'' and ''landing''—eliminating the need for complex head loading mechanisms. The Winchester technology offers a number of advantages; device reliability, data integrity, faster transfer rates and a broader range of capacities. By the early 80's, fixed disk 14″ Winchester capacities were approaching 600 megabytes. Drives with capacities of 3 to 6 gigabytes are now on the immediate horizon. Winchester innovations also served as the springboard for miniaturized rigid disk systems. First came compact single or double-platter, non-removeable 14″ units with capacities down to 10 megabytes. Then around 1975 the 8″ Winchesters appeared, closely followed by the 5¼″ units, suitable for smaller desktop computers. Today the market boasts of a continuous spectrum of small to medium Winchester sizes: 3½, 5¼, 8, 10½ and 14 inches. Capacities begin at 5 Mbytes to 900 Mbytes.

The disk drive consists of one or more platters and heads, and the control mechanism with its associated electronics. The disk is essentially a platter made of aluminum or other base material, coated with iron-oxide or other magnetizable material. Each side of the disk consists of a number of thin annular regions called *tracks*. Each track is divided into blocks referred to as *sectors*. Data and other identification information is stored in the sectors. There are two types of sectoring: hard sectored discs and soft sectored discs. The hard sectored discs have sectors demarcated by the manufacturer and are identified by a sector pulse at the start of each sector while the soft sectored discs have only an index pulse signifying the start of a track.

The more recent hard disk drives have a number of platters on the same spindle, with one head per surface. In such cases similar track position on each platter constitutes a cylinder, e.g. cylinder 0 is the cylinder corresponding to track 0 on both sides of all the platters. The reading or writing of data is accomplished by the read/write head. This head is positioned on the required track by the drives positioning control system. This process is commonly referred to as seeking and is usually less than 17 ms. The quantity of data that can be stored on a disk depends on how much of its surface area is magnetized for the storage of a bit. On a typical low cost Winchester disk track densities are around 400 tracks per inch, while flux densities range around 9000 flux transitions per inch (implying recording densities of 9000 bits per inch). The rate at which data is written on the disk or read from it is termed as *transfer rate* and ranges from 5 Mbits/sec to 24 Mbits/sec and greater. The speed at which a particular sector is found for the writing or reading of data is gauged by the access time. First the head must be positioned over the proper track referred to as seek time. Then the proper sector of the track must come under the head which is referred to as the latency time. These are some of the common terms associated with the disk drive system.

The disk selection process is a function of several factors like storage capacity, upward mobility, transfer rate, etc. Data capacity is, perhaps, the most difficult decision to make in the selection process. All questions, present and future, must be considered in the context of the application. A fail, safe option, of course, would be to select a drive design with enough potential capacity to meet any future storage requirements. Disk technology has been striving to increase capacity, with future increases taking the form of increased data densities. There is considerable room for growth. Better head and disk material techniques are being used to raise track densities. Higher track densities have resulted in replacing the head positioning stepper motors by solenoid type ''voice coil'' actuators with theoretically infinite track following resolution. Developments in disk technology can also influence the transfer rate. The transfer rate directly affects system throughput. It is the average transfer rate that counts, and again this is a function of the application.

If write/read accesses are scattered because of varied reasons, track-seeking and sector-searching delays will reduce the effective transfer rate to a fraction of the theoretical value determined from data density and rotational speed. A series of application-dependent cost-performance tradeoffs

RRD-B30M115/Printed in U. S. A.

must be individually evaluated. Higher rotational speeds reduce the latency time as the system waits for a desired sector to pass under the write/read heads. Multiple heads reduce both the number of head repositions and the distance that must be travelled. Lower cost stepper motor actuators are normally open loop—moving the heads from track to track at a constant, relatively slow rate. Voice coil actuators are more expensive but inherently faster, accelerating and decelerating in response to feedback signals from a closed loop servo system.

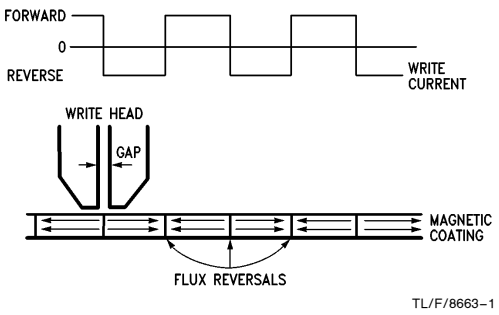
1.1 DISK STORAGE BASICS

Magnetic writing—the recording of data in a magnetic medium, is based on the principle that if a current flows in a coil of wire, it produces a magnetic field. The field is largely confined in a ring-shaped core of magnetic material, around which the wire is wound. A narrow slot is cut in the magnetic material and the field in the vicinity of the slot magnetizes the magnetic medium on the disk surface. Thus it creates alternating north-south magnets in the coated surface of the rotating disk. Thereby data is written, refer to *Figure 1.1(a).*

The head that writes the data can also be used to read it. This is done based on the principle of induction wherein a voltage is induced in an open circuit (like a loop of wire) by the presence of a changing magnetic field. In the case of a head positioned above a spinning magnetic disk on which data has been written, the magnetic fields emanate from the magnetized regions on the disk. During the time the head is over a single magnetized region, the field is more or less uniform. Hence no voltage develops across the coil that is part of the head. When a region passes under the head in which the magnetization of the medium reverses from one state to the other, i.e. a flux reversal, there is a rapid change in the field, developing a voltage pulse, refer to *Figure 1.1(b).* In this way the digital data are read as an analog signal, which can be readily converted back to digital form. The shape of this pulse and its ability to be recovered depends on various spacings. *Figure 1.1(c)* shows the spread of the coupling effect as a function of the width of the read-head gap and, equally important, the distance from the gap. The latter is, in turn, a function of both the head-surface separation and the depth of the flux reversal within the magnetic coating.

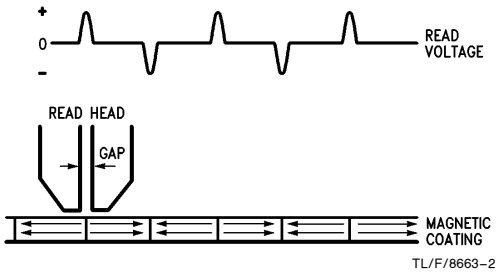
The quality of writing and reading of data depends of course on the magnetic properties of both the medium in which the data are stored and the head that writes and reads them. The common method of disk manufacture is to coat an aluminum disk with a slurry containing the gamma form of iron oxide. The iron atoms in the needle-like particles have their own minute magnetic fields and act like bar magnets with a dipole. The overall magnetization in any given region of the disk is the sum of the fields of these particles within it.

The core of most read/write heads is a ceramic consisting of spherical ferrite particles. The design of the head must conform to the design of the disk. In the case of the floppy disk(or flexible disket), which is a thin sheet of mylar plastic on which the gamma form of iron oxide is coated, the head makes contact with the surface, resulting in higher error rates and greater wear of the medium. In high performance disk drives, the magnetic medium is the coating on a rigid aluminum disk, and the head is kept from touching the medium by the so-called air-cushion effect. Consider a head that is nearly in contact with the surface of a hard disk spinning at 3600 revolutions per minute. If the length of the head along the direction of relative motion is two orders of magnitude longer than the separation between the head and the



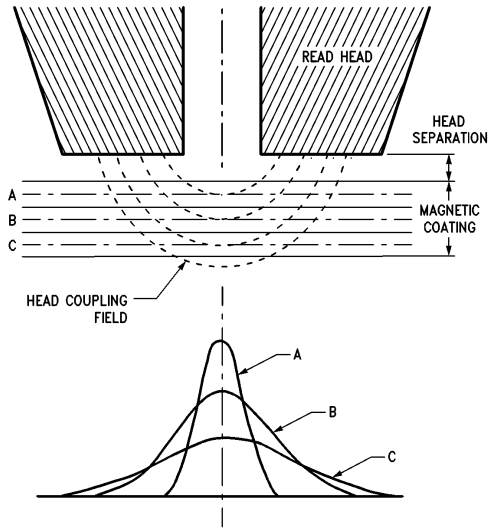
TL/F/8663–1

(a) Flux Reversals Produced by Write Current



TL/F/8663–2

(b) Read Pulses Generated by Flux Reversals



TL/F/8663–3

(c) Output as a Function of Saturation

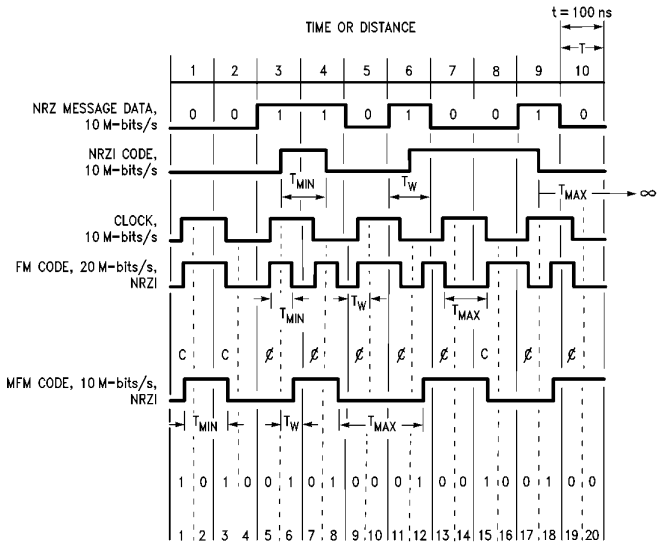
FIGURE 1.1 Flux Reversals as They Relate to "Writing To" and "Reading From" the Platter

medium, the flow of air between the head and the medium provides support for a head weighing up to several grams. Therefore because of the high cost of producing a hard disc along with its large storage capacity, it is used even if it has a bad track or sector while a floppy could be discarded. The bad sector is detected by using error checking and correction codes.

Optical Disk Technology

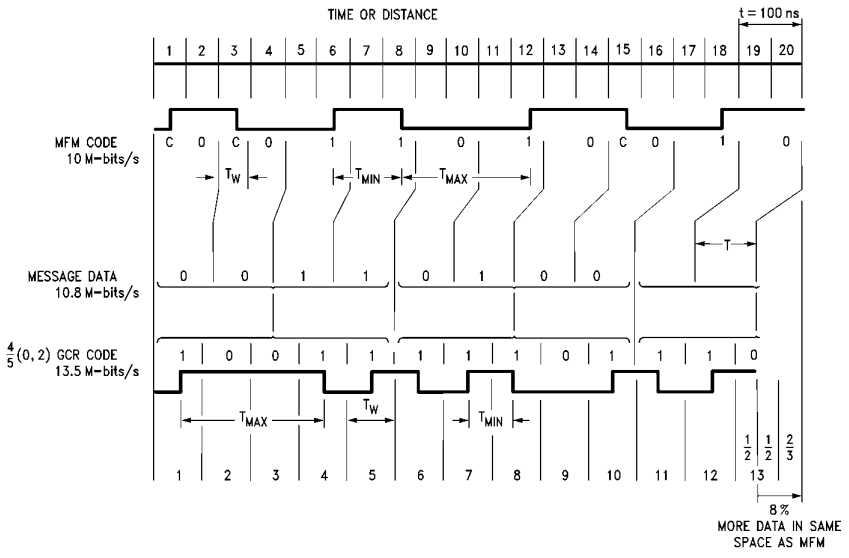
Disk drive improvements have resulted in faster data rates caused by increasing the density of the magnetic particles for greater storage. In the case of rotating magnetic memories, the strength of the signal depends on the strength of the medium's remnant magnetization. Recent advances in laser technology have resulted in digital optical disks becoming the last word in data storage and retrieval. Here, the laser beam itself provides the energy, hence the head is not in contact with the medium and it is protected, resulting in reduced errors and minimum medium wear. The advantages offered by optical disk technology are increased storage capacity, long data life, low cost per bit, noncontact read/write and easy physical mass replication. The optical disks initially developed could be written to only once. Read/write optical technology is being developed. Applications for optical disks are many and varied. On the interface level it is no different from Winchester drives and SCSI seems to be one of the most suitable of several possible choices. Another magnetic disk technology, ''vertical'' recording, is done with north-south magnetic poles perpendicular to the disk surface instead of end-to-end along the track.

FM & MFM Codes



TL/F/8663–4

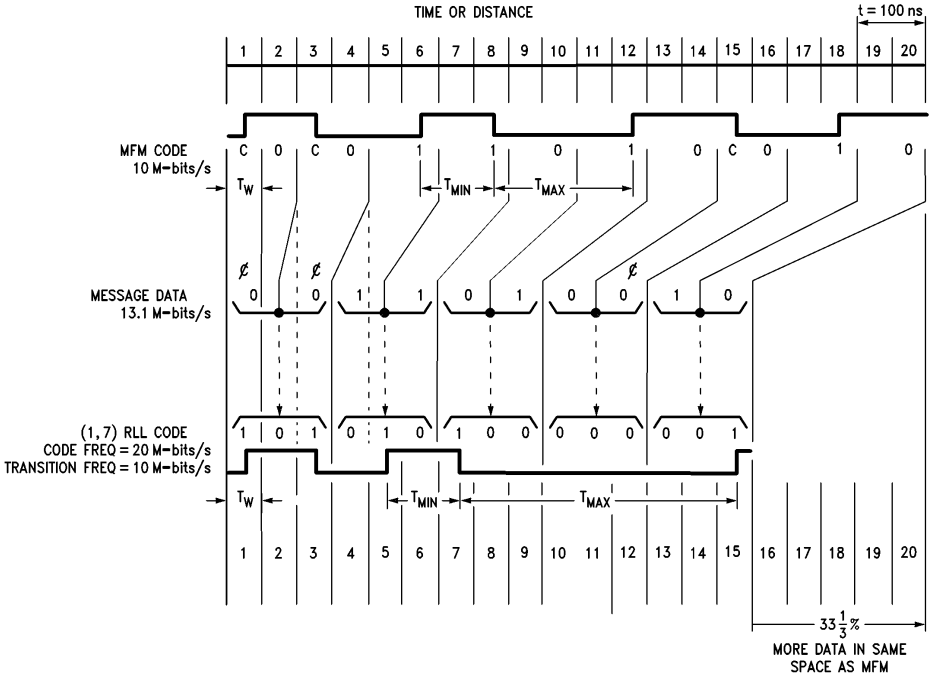
$\frac{4}{5}$ (0,2) GCR Code



TL/F/8663–5

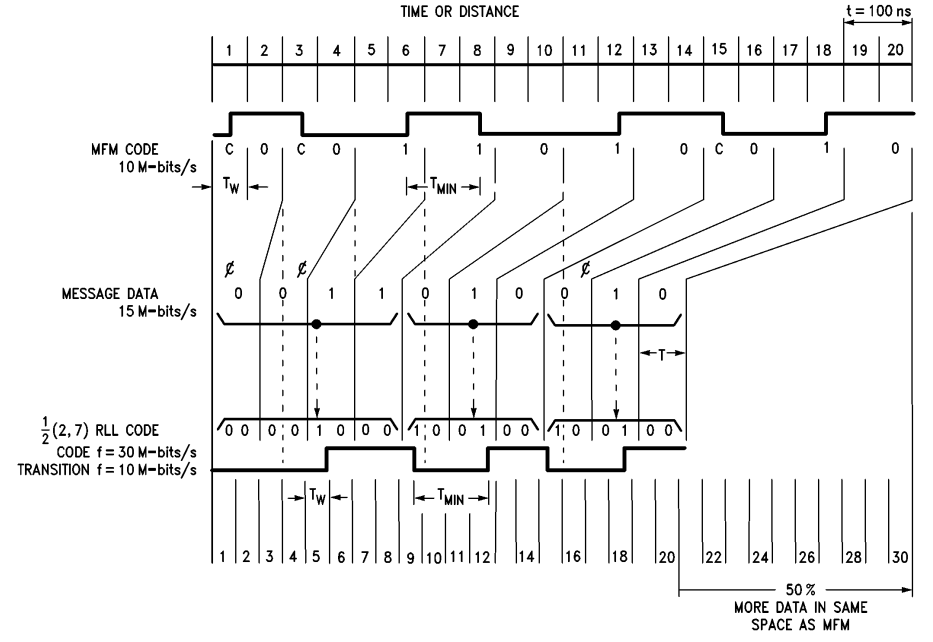
FIGURE 1.2 (a). FM, MFM and GCR Encoding

$\frac{2}{3}$ (1,7) RLL Code



TL/F/8663–6

$\frac{1}{2}$ (2,7) RLL Code



TL/F/8663–7

FIGURE 1.2 (b). RLL Encoding Schemes

1.2 DATA ENCODING/DECODING

Disk Data Encoding is the specific technique by which data is written to the disk, whereas decoding of data is necessary while reading from the disk. Data encoding removes the need of having clock information added to the track. Encoding also assists the controller in resynchronizing the data to the correct byte alignment, by allowing code violations for special data and address marks on the track. Considering the demand for ever-increasing data densities, it is understandable that the selection of a particular code is based largely on the efficiency with which flux reversals are converted into binary information, ZEROs and ONEs. In the ideal case, there should be the fewest flux reversals relative to the number of data bits they represent. Ideally, too, the code itself should provide its own ''clock'' for identifying the bit-cell intervals. Lacking this feature, a separate clock track may be required—or an extremely accurate oscillator must be provided to maintain the bit-cell divisions during intervals without flux reversals. The two requirements tend to be contradictory. An efficient code in terms of flux reversals will not be self-clocking. A self-clocking code will be wasteful of flux reversals. Nearly all of the widely used codes represent a compromise between these two extremes. *Figure 1.2(a)* shows details of FM, MFM and GCR encoding schemes, while *Figure 1.2(b)* shows details of some RLL encoding schemes. The commonly used encoding methods are discussed below in brief.

NRZ (NON-RETURN TO ZERO)

This is a telecommunication code and by far the most efficient. ''Zero'' refers to the transmission signal level. Instead of discrete pulses for each data bit, the signal rises or falls only when a ZERO bit is followed by a ONE bit or a ONE by a ZERO. NRZ coding reduces signal bandwidth by at least half. It also requires precise synchronization between source and destination in order to maintain bit cell divisions during the transmission of long strings of ZEROs or ONEs. NRZ could be used to transmit serial data to or from a magnetic recording device, disk or tape. But the extended intervals which can occur between flux reversals limit its usefulness as a recording technique.

NRZI (NRZ CHANGE ON ONEs)

This is the next most efficient code. It is widely used for tape recording and, to an increasing degree, disk recording. All ONEs are clocked, but special steps must be taken to compensate for the absence of flux reversals during strings of ZEROs. In the case of parallel-bit recording (tape), parity-bit ONEs serve as clock when all other bits in the byte are ZERO. In the case of serial-bit recording, data can be converted to RLL code (discussed below) which restricts the number of successive, unclocked ZEROs.

PE (PHASE ENCODED)

This is the least efficient of the coding methods but is completely self-clocking. The direction of a flux reversal at the middle of each cell indicates whether the bit is a ZERO or a ONE. Either one or two flux reversals occur, therefore, during each bit cell interval. The effect is to shift the ''phase'' of the signal by 180 degrees each time there is an NRZ type transition between ZEROs and ONEs.

FM (FREQUENCY MODULATION)

The FM method of encoding is equivalent to the PE technique and was the first choice for early disk-recording systems. It is generally only used for older floppy drives. Every bit cell interval is clocked by a flux reversal at the start of the cell. ONEs are marked by an additional flux reversal at the middle of the cell, doubling (modulating) the frequency of flux reversals for a series of ONEs compared to a series of ZEROs. A constant bit cell reference, provided by the clock bit, simplifies encoding and decoding with this scheme.

MFM (MODIFIED FREQUENCY MODULATION ENCODING)

With available head and media technology, MFM encoding is the most easily implemented encoding scheme and by far the most popular for floppy drives. It is used in the IBM System/34 and in available double-density LSI controller chips. MFM encoding doubles the data capacity over FM by eliminating the clock transitions (used in FM encoding) with data bits, refer to *Figure 1.2(b)*. Clock bits are still used, but are written only when data bits are not present in both the preceding and the current bit cell. As a result there is a maximum of one flux change per bit cell. Clock bits are written at the beginning of the bit cell, while data bits are written in the middle of the bit cell.

To decode data bits in MFM encoding, a data separator must generate a data window and a data window complement for a clock window. Because not every bit cell has a clock pulse, the data/clock windows cannot be timed from the clock pulse. Instead, the data separator must continuously analyze the bit position inside the windows so that the data/clock windows remain synchronous with the data/clock bits. Ideally, the clock transitions should appear at the center of the window. However, clock edges data bits can shift due to bit-shift effects. Present LSI controller chips can handle the drive interface, double density encoding function, and bit-shift pattern detection and compensation. National's DP8466 takes care of all these functions and needs only the data separator DP8465. Despite these constraints, disc controller design for MFM is simpler than that for either of the following encoding schemes.

M^{2}FM RECORDING SCHEME (MODIFIED-MODIFIED FREQUENCY MODULATION ENCODING SCHEME)

Until recently, M^{2}FM has been used as a double density encoding scheme, because the resolution of the medium and the read/write head was not adequate for the sizes of data window used in MFM. In M^{2}FM, a clock is written only if no data or clock bit is present in the preceding bit cell, and no data bit occurs in the current cell. Because clock pulses are relatively isolated on the medium, the effect of bit shift on clock pulses is minimal. Therefore, a narrower clock window can be used to decode the clock pulse. The width of the data window can thus be increased by 20%, which allows more margin for shifted data bits. Today's ceramic-based read/write heads have much better resolution than those used in the past. This head design reduces the effects of bit shift, and makes the window margin provided by M^{2}FM unnecessary. Additionally, M^{2}FM is subject to a droop problem, which occurs in the read amplifier circuit when a low frequency pattern is read.

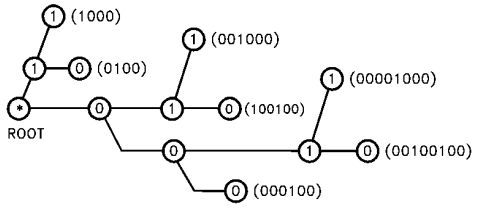
GCR (GROUP CODED RECORDING) ENCODING SCHEME

GCR encoding evolved from methods used in magnetic tape recorders. This method translates four data bits into a 5-bit code during a write. During a read, the 5-bit code is retranslated to four data bits; no clock bits are generated. Using data rates specified by drive manufacturers, this scheme is less dense than MFM. This method requires more circuits to code and decode, requiring necessary look-up tables, and costs more than either of the other two encoding schemes. For example; 1101 is encoded into a serial bit stream 01101 according to GCR encoding rules. To de-

code, a data window is generated around the expected position of each bit. The result is serial read data of 01101, which must be decoded to 1101 by lookup tables.

RLL (RUN LENGTH LIMITED) ENCODING SCHEMES

These recently popular encoding schemes are used in big 14″ drives from IBM, CDC and DEC, and are starting to make an appearance in the small 5¼″ drive market. The RLL encoding schemes have an excellent encoding efficiency, up to 50% higher than MFM. It is, however, considerably more complex to generate, requires a much better data separation unit to recover recorded data and is more suceptible to wider error bursts. The encoding rules for RLL depend on the RLL scheme chosen. The most common one is the 2,7 RLL code which refers to the maximum number of consecutive 0s, refer to *Figure 1.2(b)*. A standard encoding tree is defined and the data is encoded on the basis of those rules, as shown in *Figure 1.3*. The data bit stream is taken and the encoding tree is traversed, starting at the root, where the nodes traversed are the data bit stream in the sequence they arrive. On reaching the leaf of the tree, the code there is then the 2,7 RLL code for that data stream, e.g. if there is a data stream 100011010, then on traversing the tree, a data bit stream of 10 has a code of 0100, while the next bits 0011 are encoded as 00001000.



TL/F/8663–8

FIGURE 1.3. Encoding Tree—2,7 RLL Code

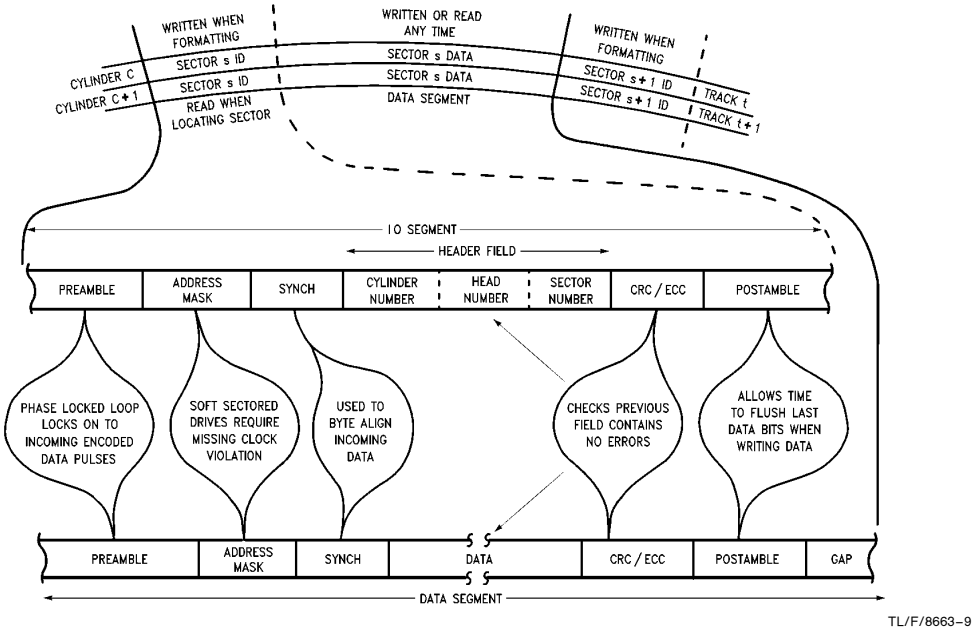
1.3 MEDIA FORMATTING

Media formatting provides the user with a reliable means of data retrieval using the magnetic recording surface of the track. There are many different formats but most of them are variations of the same basic structure. The formatting process is different for hard sectored and soft sectored disks. In hard sectored disks the sectors are defined by the manufacturers because the start of each sector is identified by the sector pulse generated by the drive. In soft sectored disks the drive issues only an Index pulse at the beginning of the track and the user can define all details of how information will be stored on the track, allowing more flexibility. *Figure 1.4* shows the basic format used. It consists of two segments—the ID segment and the data segment. The ID segment contains unique header information for the sector and the data segment contains the actual data. When the system requests a particular sector on a disk, the head must be positioned over the selected track, and the desired sector on that track must be found. This requires electronics to lock on to the data stream and then decode it. The beginning of a track is indicated by the Index pulse while the beginning of the sector is indicated by the sector pulse. This is followed by gap before the start of the sector on the track, which is referred to as the **Post Index/Sector Gap**. The explanation of the various fields are given below:

1.3.1 ID Segment

PREAMBLE OR PLL SYNCH FIELD

This is a field of repetitive clocked data bits usually 10 to 13 bytes long. The preamble normally will be all zeroes of NRZ data (encoded as 1010. . . in MFM). During the ID preamble, the signal Read Gate will go active, indicating that the incoming data pattern has to be locked on to.



TL/F/8663–9

FIGURE 1.4. Typical Sector Format Showing the Various Fields Within the ID and Data Segments

ADDRESS MARK FIELD

Address Mark (AM) is required on soft sectored drives to indicate the beginning of a sector, because this type of drive does not have a sector pulse at the start of each sector. This address mark byte contains a missing clock code violation, typically in MFM. The violation is detected by circuitry to indicate the start of a sector. The first decoded byte that does not contain all 0s after the preamble will be the address mark. The first 1 to be received is then used to byte align after the all zeroes preamble. Some formats have one ID address mark byte, while others have several.

ID SYNCH FIELD

For a hard sectored disk, byte alignment begins with the synch field that follows the preamble. The Synch bytes constitute a bit pattern that enables control circuitry to determine the byte boundaries of the incoming data, bit synchronization. Synch field usually follows the address mark on soft sectored drives and the AM is used for byte alignment also. Some formats use two synch fields: synch #1 and synch #2.

HEADER FIELD

The Header Field format varies between drive types, but typically has two cyclinder number bytes, a sector number byte, and a head number byte. It is generally 3 to 6 bytes long and one of the bytes may contain bits for bad sector or bad track recognition.

HEADER CRC/ECC FIELD

CRC (Cyclic Redundancy Checking) code or ECC (Error Checking and Correcting) code is appended to the header field. If CRC is used it consists of two bytes of the standard CRC-CCITT polynomial. The code detects errors in the header field. If ECC code is used, it is normally the same ECC polynomial that is used for the data field. This appendage is basically a protection field to make sure that the ID field contains valid information.

POSTAMBLE

This field may be used to give the disk controller time to interpret the data found in the ID field and to act upon it. It provides slack for write splicing that occurs between the ID and Data segment. A Write splice occurs when the read/write head starts writing the data field. A splice is created each time a sector's data segment is written to. The slight variations in the rotational speeds cause the first flux change to occur in different positions for each write operation. It also allows time in a write disk operation for the read/write circuitry to be switched from read to write mode. Finally it allows time for the PLL circuit to re-lock on to a fixed reference clock before it returns to synchronize to the preamble of the data field.

1.3.2 Data Segment

PREAMBLE FIELD

The Data Preamble field is necessary when reading a sector's data. It ensures that the PLL circuit locks on to the Data segment data rate. Initially, the ID segment and the data segment of every sector will be written when formatting the disk, but the Data segment will be written over later. Due to drive motor speed variations within the tolerance specified, the ID and Data segments will have slightly different data rates because they are written at different times. This implies that the PLL must adjust its frequency and phase in order to lock on to the data rate of the Data segment before the incoming preamble field has finished. Hence the need for a second preamble field in the sector.

DATA ADDRESS MARK FIELD AND DATA SYNCH FIELD

Following the Data Preamble will be the Data Address Mark for soft sectored drives, and Data synch, both similar to the ID segment equivalents.

DATA FIELD

The Data field is transferred to or from external memory. It is usually from 128 bytes to 64 kbytes per sector.

DATA CRC/ECC

A CRC/ECC appendage usually follows the Data field. CRC/ECC generating (when writing to the disk) and checking (when reading from the disk) are performed on the Data field. Errors may therefore be detected, and, depending on the type of error and if an ECC polynomial is used, they may also be corrected.

DATA POSTAMBLE FIELD

This has the same function as the ID Postamble field.

GAP FIELD

This is sometimes referred to as Gap 3, and is the final field of the sector. It allows slack between neighboring sectors. Without this gap, whenever a data segment is written to a sector, any reduction in drive motor speed at the instance of writing to the disk would cause an overlap of the data segment and the succeeding ID segment of the next sector. This field is only written when formatting the disk.

A final gap field is added from the end of the last sector until the INDEX pulse occurs and this gap is often termed Gap 4. It takes up the slack from the end of the last sector to the Index pulse.

1.4 THE DISK SYSTEM—DRIVE AND CONTROLLER

The Disk system essentially consists of two main paths: 1)the Disk Data Path, 2) the Disk Control Path, refer to *Figure 1.5*. The disk control path is responsible for controlling the disk drive with respect to positioning the head at the desired track and control the associated control signals (these are a function of the disk interface). The various disk interfaces are discussed later. The other component of the Disk System is the Disk Data Path which is responsible for data transfer from and to the disk.

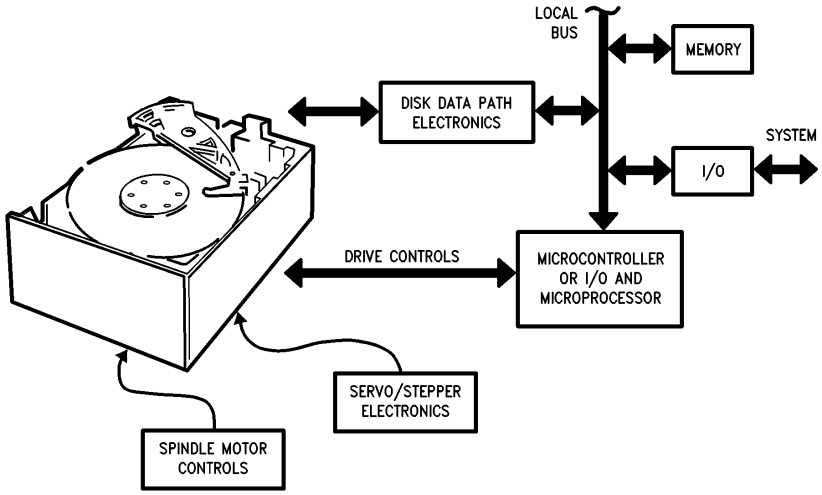
1.4.1 Reading Data from the Disk

Reading data from the disk to the system memory is a complex process and involves a number of operations enroute, as shown in *Figure 1.6(a)*. To initiate a read operation—a command is sent to the disk drive indicating the track and sector from which data is to be read. The seek operation moves the head to the desired track on the disk. Eventually the desired sector is identified by the header ID segment and the various fields are checked depending on the formatting rules used. The flux reversals are recorded by the head and are of the order of 500 microvolts. These pulses are then amplified by the read/write amplifier to about 10 mV. The signal from the read/write amplifier when reading a disk is therefore a series of pulses with alternating polarity. These pulses are passed through a Pulse Detector, like the DP8464. Electrically, these peaks correspond to flux reversals on the magnetic medium. The Disk Pulse Detector accurately replicates the time position of these peaks. The Disk Pulse Detector utilizes analog and digital circuitry to detect amplitude peaks of the signal received from the read/write amplifier associated with the heads of disk drives. A TTL compatible output is produced which on the positive leading edge indicates a signal peak.

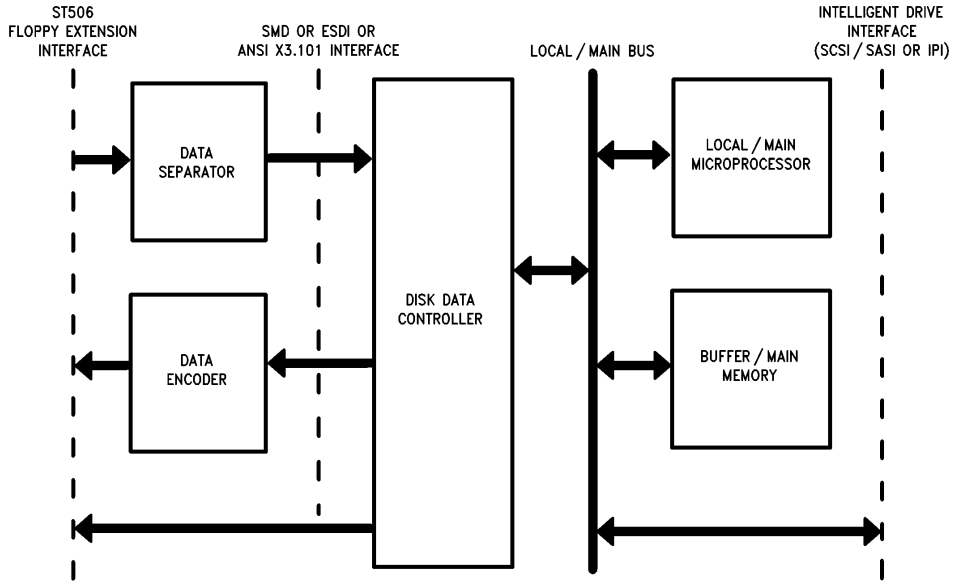
The raw data coming from the pulse detector consists of composite clock and data bits depending on the encoding scheme used. This encoded data has to be synchronized and decoded. These functions are performed by the Data Separator, like the DP8465. Due to bit shifting and distortion of the read pulses, the Pulse Detector issues non-synchronous pulses. For reliable decoding this jittery bit stream must be synchronized. The data separator has a Phase Locked Loop which attempts to lock on to the bit stream and synchronize it.

In hard sectored drives, the sector pulse indicates the beginning of the sector. Normally the preamble pattern does not begin immediately, because gap bytes from the preceding sector usually extend just beyond the sector pulse. Allowing two bytes to pass after the sector pulse helps ensure that the PLL will begin locking on to the preamble and will not be chasing non-symmetrical gap bits. For soft sectored drives, the controller normally will not wait for the Index pulse before it attempts lock-on. Chances are the head will not be over a preamble field and therefore there is no need to wait two bytes before attempting lock-on.

Disk Data Controller in a Disk System

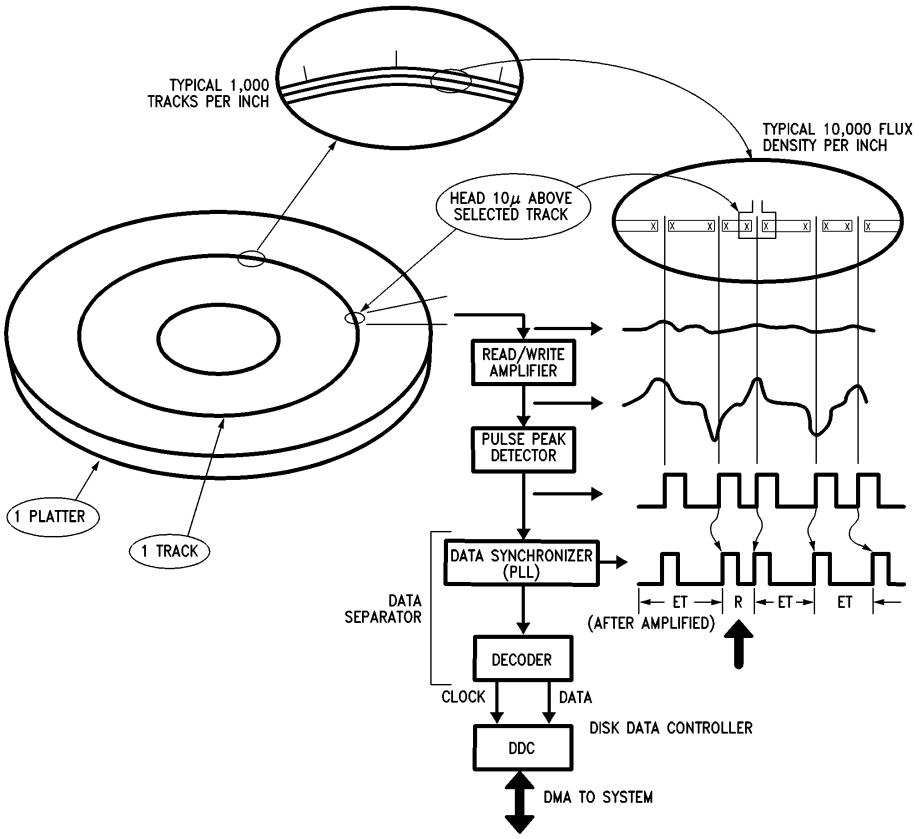


TL/F/8663–10

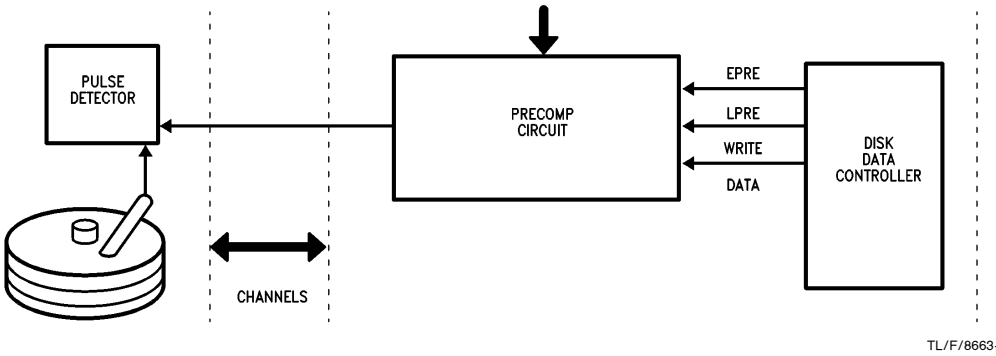


TL/F/8663–11

FIGURE 1.5. Disk System—Data and Control Paths



(a) Reading From Disk



(b) Writing to Disk (MFM Encoding)

FIGURE 1.6

Having locked-on to the bit stream the data synchronizer circuit must first determine the nominal position of clock and data bits and then generate an appropriate clock and data window that is centered around the bit positions. However there are many causes for bits to shift from the position where they are written. Erroneous data could be issued if bit jitter is beyond the tolerance computed. Therefore, special design considerations must be given to the type and resolution of the Data Separator used in reading data bits from the disk. The more accurately the bit position can be determined and the tighter the resolution of the data window, the lower is the soft error rate of the disk. Essentially the Data Separator's Phase Locked Loop locks on to the basic frequency of data bits read from the disk, and determines nominal bit positions for data and clock bits by sampling every bit (clock and data). It uses the phase relationship between a bit and its window to vary the position of the window. By sampling each bit, the phase-lock loop determines the phase error between a bit and the frequency being generated. To determine the nominal bit position around which to center the window, the data separator must track data bit frequency changes, yet ignore jitter. In this manner, even if an unpredictable bit shift occurs, the data separator can adjust the window's position to compensate for the change. Otherwise the shifted bit could be positioned outside the window. To remain within the typical error rate specified by the system, not more than 1 in 10^{10} bits can appear outside the window. With the present media technology, only a data separator based on an analog phase-lock loop technique can provide the necessary reliability.

Once the bit stream read from the disk has been synchronized and decoded to NRZ data, it is directly sent to the Disk Data Controller block, DDC, like the DP8466. In the DDC, the serial data is converted to parallel data (in terms of bytes), by the deserializer block. The main task is to recognize the byte boundaries accurately. In soft sectored drives this can be done by detecting a ''missing clock'' signal, which provides a fixed reference in the bit stream to set the byte boundary. Upon receipt of this signal the divide-by-eight circuit is set, to allow subsequent stages of the controller to acquire the bytes correctly. Hard sectored drives use a preset bit pattern in the synch field to determine byte alignment. Once the data is in parallel form it is stored in a temporary register in the controller. Transfer of data from this register to the system memory is achieved by DMA (Direct Memory Access) transfer. In this fashion data are read from the disk and transferred to the system.

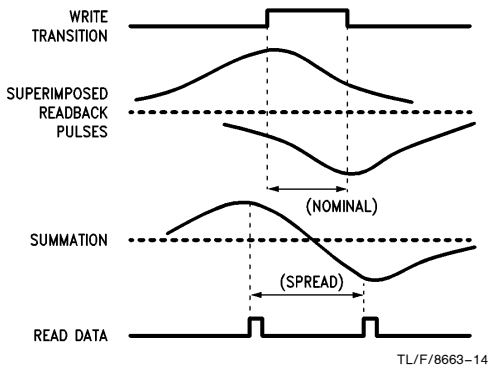
1.4.2 Writing Data to the Disk

The process of writing data to the disk is similar to the read operation in the reverse direction, with some changes. The write operation is initiated after the appropriate Seek command has been issued to the drive and the head is positioned over the desired track/sector. *Figure 1.6(b)* shows the basic write path blocks. Data is transferred from the system to the Controller using the DMA. The parallel data is converted to serial data by the serializer in the controller. This operation is conceptually easier to do, as the controller already has the right byte boundaries in the data and knows exactly where to insert the address mark. Most disk Controllers, like National's DP8466, provide either NRZ encoded data or MFM encoded data.

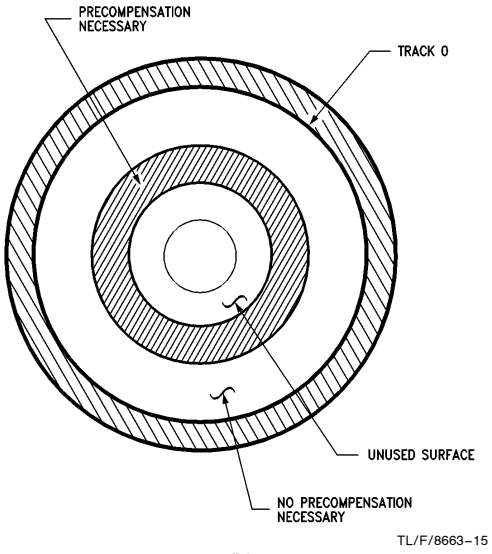
As mentioned in the previous section, predictable bit shift effects result from normal read/write head operation. Data are written when the read/write head generates a flux change in the media. In reading, a current is induced into the read/write head when a flux transition on the medium is encountered. The current change is not instantaneous, since it takes a finite time to build up to the peak and then to return to zero, refer to *Figure 1.7(a)*. If flux transitions are close together, the signal buildup after one flux transition declines, but it does not reach zero before a second transition begins. So when the flux changes are detected by the read/write head the peaks are shifted. A negative flux change, for example, may appear late because it has been added to the remnant of a positive transition. Narrower spacing between bits results in greater bit shift on the inner tracks. Hence compensation is needed on the inner tracks to minimize bit shift while no compensation is required on the outer tracks as bit shift is negligible, *Figure 1.7(b)*. Two methods currently being used are *precompensation* and *postcompensation*.

With precompensation, bits are deliberately shifted in the direction opposite to that of the expected shift. As data are being written, the controller detects bit patterns. From these bit patterns, the controller calculates which bit will shift in which direction. For example, a 4-bit pattern of 0110 on an inner track would cause the third bit to appear a few nanoseconds later than its nominal position. The controller chip, after detecting this late bit shift pattern, would generate an early signal, indicating that the third bit should be written earlier to make it appear closer to its nominal position when read. Conversely, if the third bit were going to appear early, a late signal would be generated so that the bit could be written later. How early or late the bit should be written is a function of its position in the data pattern, track position, and media, among other factors. Most Controllers provide signals to indicate what type of compensation is necessary. External circuitry is used to provide the actual delay as shown in *Figure 1.7(b)*.

The encoded precompensated data is then sent to the read/write amplifier where the stream of pulses is recorded on the disk as magnetic flux reversals. Postcompensation can be used when reading, usually as filter components around the pulse detector.



(a)



(b)

FIGURE 1.7. Bit Shifting

1.4.3 DMA (Direct Memory Access) Transfer/Data Buffering

The DMA block is responsible for the transfer of data between the host system and the disk controller. This is done because it is inefficient to dedicate a special communication channel to the task of transferring data between the disk controller and the system. The DMA system takes control of the control lines associated with a system's address and data buses, and exercises them in such a way as to transfer data in an appropriate direction from one device to another. It is also generally optimally efficient in using the available bus bandwidth whenever it is on the bus. The DMA capability is built-in for some disk controllers while in some an external one is required. National's DP8466 supports a single or dual channel DMA with capability of using an external DMA instead, if desired.

Data buffering is the temporary storage of some or all of the data to be transferred between the disk and the system memory. Any centrally intelligent system benefits from minimizing the bus occupancy. This is because the system has a lot of other tasks to perform, and if the bus is too heavily used, the system will miss performing some timely tasks. Therefore to prevent this, the data from the disk is transferred to a FIFO (First In First Out buffer). In a dual-DMA system, the local channel transfers the data from the FIFO to a local buffer memory while the remote DMA channel optimally transfers data from the local to the remote main memory over the system bus. This minimizes bus bandwidth use by the Disk I/O channel. The size of the FIFO is a function of different factors like: 1) the rate at which the system picks up the blocks of data, 2) the data rate from the disk and, 3) the burst transfer rate of the DMA. The way this is incorporated may differ in disk controllers. The buffer memory optimizes bus bandwidth. A system utilizing a single channel DMA would transfer data from the FIFO directly to the Host.

1.4.4 Error Detection/Error Correction

There are a number of factors which contribute to disk errors, viz. electrical noise, crosstalk, inadequately erased signals from previous recording, offtrack error in positioner, pin holes, inclusions, media thinning, and pattern induced errors. Of these, media defects are permanent errors. In general, ECC (error checking and correcting code), ensures reliable data storage and recovery. Generation of the ECC polynomial involves a detailed understanding of the mathematics of coding theory, and a cookbook approach to designing ECC logic. The basic idea behind ECC is the concept of irreducible polynomials. Take an irreducible polynomial (prime) and multiply it by the data pattern. Store the resulting remainder on the disk after all the data has been sent through the polynomial, *Figure 1.8(a)*. When reading the data back, divide the data coming off the disk into the polynomial. The reciprocal of the result should be equal to the check bytes on the disk, *Figure 1.8(b)*. If not, there is an error. The way error correction works is shown in *Figure 1.8(c)*. If there were no errors, then the sequence follows the straight path and the shift register contains all zeroes. If an error occurred, then at the point of occurrence the sequence vectors off, and at the end the shift register contains the pattern which caused the error. This helps in tracing back to the point of occurrence. The correction span is the number of contiguous bits in error which could be corrected. The probability of miscorrection is given by:

$$P_{mc} = (2^{(C-1)}) \times S/2^{A}$$

where C = correction span in bits

S = no. of bits in the sector

A = no. of bits in ECC appendage

Some codes have a higher miscorrection probability due to pattern sensitivities. A 48-bit ECC with an 11-bit correction span is recommended for 1,7 or 2,7 Run Length Limited encoded disks, while for MFM a 32-bit ECC with a 5-bit correction span results in low miscorrection probabilities. There are different types of codes:

FIRE CODES

Used in older systems and some current chips on the market. Fire codes have a high miscorrection probability related to double-burst errors (errors at two locations separated by more than the detection span) and are not recommended by National. Some examples of fire codes are:

32-bit FIRE CODE $(x^{21}+1)*(x^{11}+x^2+1)$

48-bit FIRE CODE $(x^{13}+1)*(x^{35}+x^{23}+x^8+x^2+1)$

56-bit FIRE CODE $(x^{22}+1)*(x^{11}+x^7+x^6+x+1)*(x^{12}+x^{11}+\ldots+x^2+x+1)*(x^{11}+x^9+x^7+x^6+x^5+x+1)$

COMPUTER GENERATED CODES

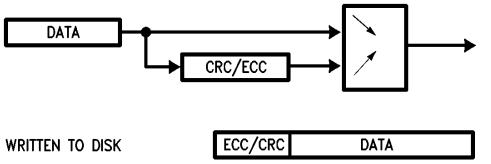
These have a very good reliability and are specifically chosen to guarantee not miscorrecting a specified worst case double burst error. The reliability can be calculated by the equation for miscorrection probability. National recommends the use of these codes for disk systems.

DOUBLE BURST REED SOLOMON CODES

Reed Solomon codes can handle longer bursts, multiple burst error (two burst error within a sector) correction capability and would be necessary for use with some optical media because of the high error rates. Typically RS codes for optical media are as long as a quarter of the sector. An example of the Reed Solomon code is given below.

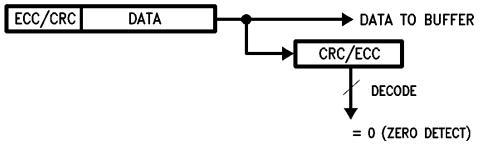
$$(x + a^5)*(x + a^6)*(x + a^7)*(x + a^8)*(x + a^9)$$

How Error Detection Works



TL/F/8663–16

(a) Writing to Disk



TL/F/8663–17

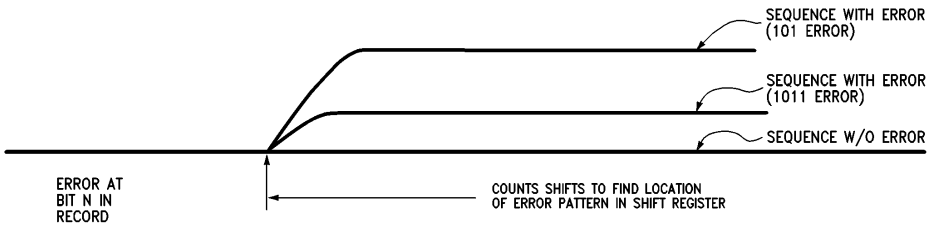
(b) Reading From Disk

$$\frac{\text{Data}}{\text{Polynomial}} = \text{Quotient (Discard)} + \frac{\text{Remainder}}{\text{(append as ECC)}}$$

$$\frac{\text{Data} + \text{Remainder}}{\text{Polynomial}} = \frac{\text{Data}}{\text{Polynomial}} + \frac{\text{Remainder}}{\text{Polynomial}}$$

$$= \frac{\text{Remainder}}{\text{Polynomial}} = \phi$$

If CRC Read = CRC Written



TL/F/8663–18

(c) How Correction Works

FIGURE 1.8. ECC/CRC

CYCLIC REDUNDANCY CODE

These can only detect errors and will not correct. They are generally used for header appendage and in floppy drives. The most widely used code is the CRC-CCITT code given below.

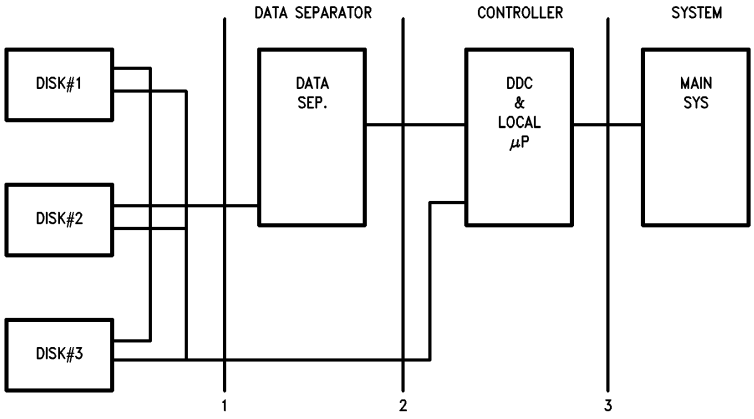
CRC-CCITT 16 bit $x^{16} + x^{12} + x^5 + 1$

Selecting the correct CRC/ECC is a function of the parameters being evaluated.

Detection Ability: The ability of the CRC/ECC to detect errors in the data transferred, measured as the number of bits affected and number of distinct bursts, the important measure being the guaranteed value.

Correction Ability: The ability of the ECC to restore erroneous data to its original state. Again, like the detection ability, this is measured as number of bits and number of bursts, the important measure being the guaranteed value.

Operating Environment: This involves factors like encoding scheme, data rate, data block size, technology on disk, product environment, and compatibility.



TL/F/8663–19

(a) CONCEPTUAL REPRESENTATION OF THE DISK SYSTEM WITH POTENTIAL INTERFACE POINTS.
1 = ST506/ST412; 2 = ESDI, SMD; 3 = SCSI, IPI

Interface	Year	Data Rate	Connectors	Drives	Status
SMD*	1975	≤15 Mb/s	60-Pin, 26-Pin	Hi Perf 8″, 14″	Upgrading Now
SA1000	1978	4.3 Mb/s	34-Pin, 20-Pin	Low Cost 8″	Limited Future
ST506	1980	5 Mb/s	34-Pin, 20-Pin	Most 5¼″, 3½″	Still Popular
ST412HP	1983	10 Mb/s	34-Pin, 20-Pin	Low Cost 8″	
ESDI*	1983	10–15 Mb/s	34-Pin, 20-Pin	Mid-Hi Perf 5¼″	New Standard
Future	1985–6	24 Mb/s	34-Pin, 20-Pin	Hi-Perf 5¼″, 8″	

*Data separator on the drive.

(b) POPULAR HARD DISK DRIVE INTERFACES

Interface	Year	Data Rate	Data Bus	CTL Bus	System	Status
SASI	1981	≤1–2 Mb/s	8-Bit + P	9-Bit	Low-End	Superseded by SCSI
SCSI Asynchronous	1982	≤1–2 Mb/s	8-Bit + P	9-Bit	Low/Mid-End	ANSI Standard
SCSI Synchronous	1984	≤4 Mb/s	8-Bit + P	9-Bit	Mid-End	Recent ANSI Standard
IPI-3	1984	≤10 Mb/s	8-Bit + P 16-Bit + 2P	6-Bit	High-End	Almost ANSI Standard

(c) POPULAR INTELLIGENT DISK SYSTEM INTERFACES

FIGURE 1.9. Drive Interface Standards

1.5 THE DISK DRIVE CONTROL PATH

The disk drive control path essentially consists of the various control signals defined by the drive interface for drive control and data path control. The control path in a disk system has a number of potential interface points, *Figure 1.9(a)*. The popular hard disk drive interfaces which define the physical connections of the controller with the drive are given in *Figure 1.9(b)*. These are at the interface points 1 and 2. At these points data is still in the serial format. With the advent of sophisticated controllers many Intelligent disk system interfaces have come into being, *Figure 1.9(c)*. These essentially incorporate the complete controller on the drive and interface to the outside world, through an 8- or 16-bit standard bus, interface specific. The physical interface with the disk is usually one of the standard hard disk drive interfaces mentioned in *Figure 1.9(b)*.

INTERFACE STANDARDS

Interface standards are the definition of the connection between parts of the disk unit, controller, and system. Interfaces can be defined on several levels, viz.

- Electrical specification of signal levels.
- Timing relationships between signal lines.
- Physical specification of cabling, connectors etc.
- Functional specifications of tasks the standard performs.
- Command descriptor specification of the standard.

Some interface standards require only a subset of the above definition categories. For example, there is no necessity for a command descriptor segment to the ST506 defacto standard, as no provisions are made for command communication other than the simple control lines described in the functional specifications.

Standards are important as they allow numerous manufacturers to cater to the same market segment, thus creating healthy competition. For example, the ST506 interface is an industry standard simply because it is being used by a lot of drive manufacturers. The factors which affect the choice of the standard are: data rate, flexibility, popularity, performance, and cost.

1.5.1 Popular Hard Disk Drive Interfaces

There are a number of disk interface standards. It is important to realize the implications of the various contenders for the interface point. For example, if the data separator is placed on the drive, the cost of the drive increases, the cost of the controller board decreases, the speed of the interface can increase if desired, but the system has to cope with this increase. Some of the most commonly used standards (disk interfaces) are discussed briefly in the following sections.

FLOPPY DISK INTERFACE

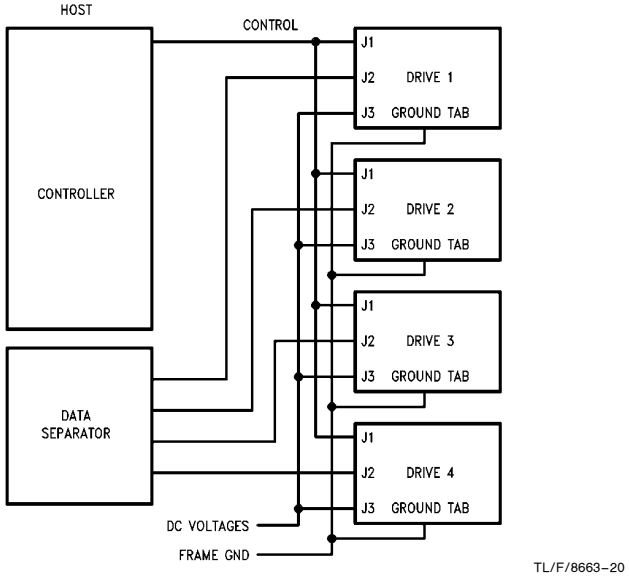
This is a relatively ''dumb'', single connector, serial data and control interface. There are two lines which carry the read/write data, and several control signals. This interface is positioned at point 1 in *Figure 1.9(a)*. The data rate for such interfaces is comparatively slow, around 100 to 500 kBits per second and the data capacity of floppy disks is not very large. The head is positioned by issuing step pulses to the drive. Read and write operations are initiated by asserting signals called Read Gate or Write Gate.

INTERFACE SIGNALS

Head load, Index, Sector (hard sectored drives only), Ready, Drive Select (usually 4), Step, Direction, Write Gate, Track 0, Write protect.

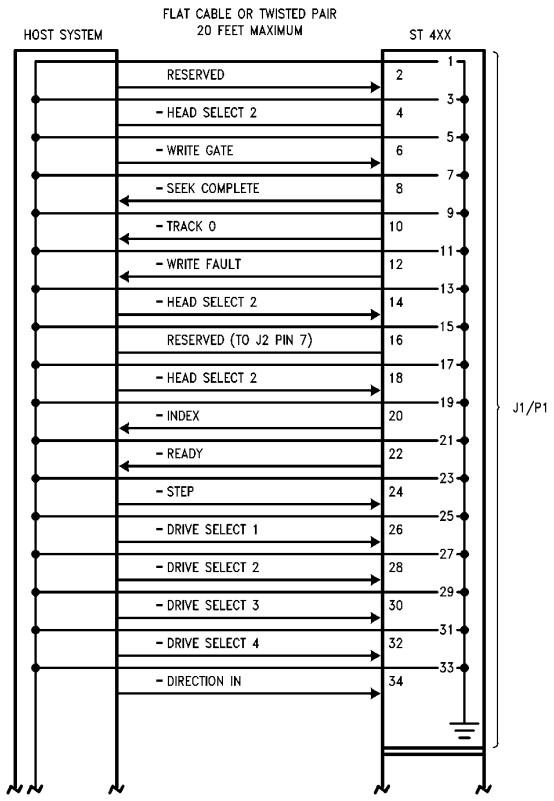
ST506/ST412 DISK INTERFACE STANDARD

This is also sometimes referred to as the floppy extension interface and is one of the most commonly used interface standards. The data rate is defined to be 5 Mbits per second, and the code is MFM. The interface is divided into two cables—a 34-pin control cable and a 20-pin data cable. The control cable allows for a daisy chain connection of up to four drives with only the last drive being terminated, *Figure 1.10*. The data cable must be attached in a radial configuration. This interface is at point 1 and, hence, the data separator is a part of the controller.

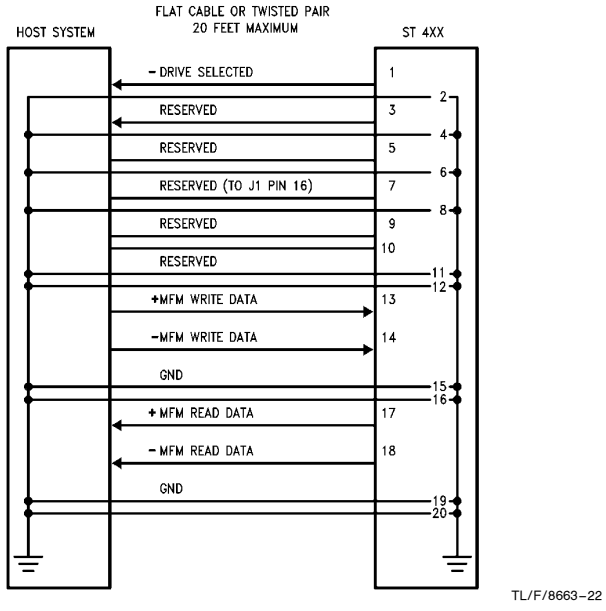


(a) Typical Connection, 4 Drive System

FIGURE 1.10. ST506/412 Configurations



(b) Control Signals Cable



(c) Data Signals Cable

FIGURE 1.10. ST506/412 Configurations (Continued)

Functional Operations

DRIVE SELECTION

Drive selection occurs when one of the DRIVE SELECT lines is activated. Only the selected drive will respond to the input signals, and only that drive's output signals are then gated to the controller interface.

TRACK ACCESSING

Read/Write head positioning is accomplished by:

a) Deactivating WRITE GATE line.

b) Activating the appropriate DRIVE SELECT line.

c) Being in the READY condition with SEEK COMPLETE true.

d) Selecting the appropriate direction.

e) Pulsing the STEP line.

Each step pulse will cause the head to move either one track in or one track out depending on the level of the direction line. A low level on the DIRECTION line will cause a seek inward toward the spindle, a high, outward toward track 0. Some drives have buffered seeks where the drive stores the pulses until the last one is received, then executes the seek as one continuous movement.

HEAD SELECTION

Any of the heads can be selected by placing the head's binary address on the Head Select lines.

READ OPERATION

Reading data from the disk is accomplished by:

a) Deactivating the WRITE GATE line.

b) Activating the appropriate DRIVE SELECT line.

c) Assuring the drive is READY.

d) Selecting the appropriate head.

WRITE OPERATION

Writing data onto the disk is accomplished by:

a) Activating the appropriate DRIVE SELECT line.

b) Assuring the drive is READY.

c) Selecting the proper head.

d) Insuring no WRITE FAULT conditions exist.

e) Activating WRITE GATE and placing data on WRITE DATA line.

Electrical Interface

The interface to the ST506/ST412 family can be separated into three categories, each of which is physically separated.

1. Control Signals.

2. Data Signals.

3. DC Power.

All control lines are single ended and digital in nature (open collector TTL) and either provide signals to the drive (input) or signals to the controller (output) via interface connection J1/P1. The data transfer signals are differential in nature and provide data either to (write) or from (read) the drive via J2/P2. *Figure 1.10* shows the connector pin assignments for this interface.

Since the data separator is on the controller, the ST506/ST412 drive must have a transfer rate of 5 M bit/sec. The bit density cannot be increased as the data rate and disc rotational speed are fixed. The only way to increase drive capacity is to increase the number of tracks, which does not allow large increases of capacity. Despite this limitation, it has a strong future as it moves into lower cost systems and smaller 3½″ drives.

ST412 HP INTERFACE

This standard was designed to provide an upgrade path from the ST506 and is very similar. This interface is also at point 1. The main differences from the ST506 family are:

- One additional control line in the daisy chain . . . Recovery mode.
- Reduced write current is not part of the interface.
- The data rate is 10 Mbits/sec.
- The encoding scheme is not tightly specified, but suggested to be MFM.
- The maximum repetition rate of step pulses has been increased.

The major benefit of this interface is the higher data rate compared to ST506 drives, however, a much more careful design is needed to keep the bit error rate the same and for this reason may not be popular. Since the data separator is located on the controller, the data transfer rate must be exactly the 10 Mbits/sec rate and still be MFM encoded.

Recovery Mode

Recovery mode has been added in response to higher track density. It is asserted by the controller in response to bad data. In this mode the controller issues up to eight step pulses, and the drive steps through its own micropositioning algorithm. After each pulse, the controller tries to reread data and, if it fails again, after the eighth try it abandons the procedure. This drive interface emerged as higher data rate embellishment to the ST412.

ESDI (ENHANCED SMALL DEVICE INTERFACE)

This interface is at point 2 on *Figure 1.9*. This standard was a proposal by Maxtor Corporation, subsequently modified by an experienced working committee, and is finding growing acceptance largely because it is a sensible proposal. It has control and data cables like the ST506/412 interface but adds a driver and receiver on the data cable for the clock information, as shown in *Figure 1.11*. The implication is that the data separator resides on the drive, which means fewer design problems for drive users, and that certain status and command information is transmitted in serial, which means more control circuitry on both sides of the interface. The data rate is allowed to be several frequencies, dependent upon options, with the maximum rate probably reaching 24 Mbits/sec.

Features

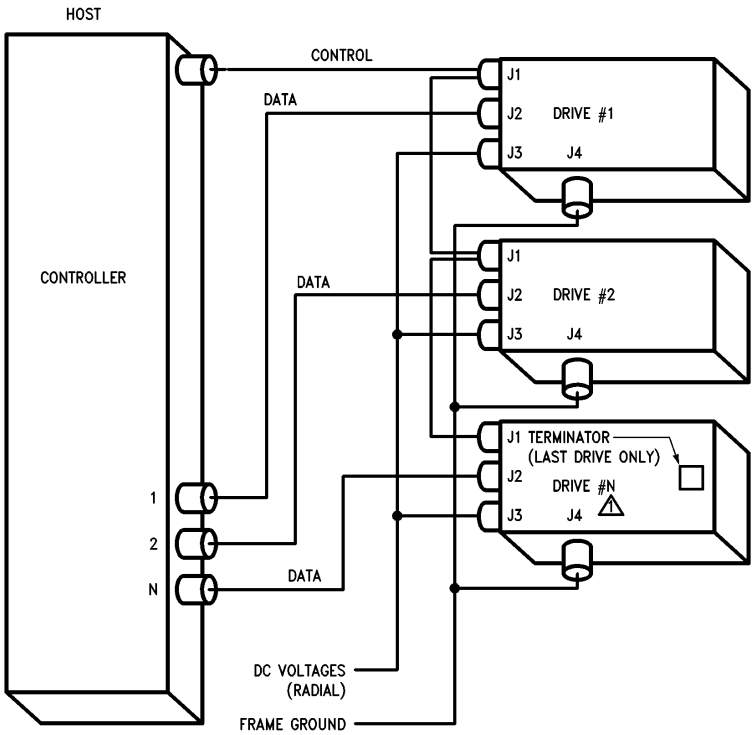
- Low cost, high performance interface suitable for smaller, high performance drives.
- Two protocols supported: serial and step mode.
- Supports up to 7 drives in the serial mode and 3 drives in the step mode.
- Maximum data rate of 24 Mbits/sec.
- Supports cable lengths of up to 3 meters.
- Serial mode of operation utilizes NRZ data transfer along with serial commands and serial configuration and status reporting across the command cable.

- Step mode implementation utilizes the same NRZ data transfer; however, the step and direction lines are used to cause actuator motion. Hence, with this mode configuration and status reporting are unavailable over the interface.

The ESDI interface puts the data separator on the drive and its output is NRZ data with a synchronous clock. This results in the data rate, and therefore the bit density, not being rigidly defined. The controller speed is governed by the synchronous clock coming from the drive, not from a data separator as in the ST412/ST506 interfaces. The drive is code independent as the data across the interface is always NRZ (or decoded) format. This enables the use of codes like RLL which put more data on the disk for the same bit density (flux reversals per inch). Moreover the use of NRZ encoding results in decreased errors due to electrical transients on the interface cable. This lowers practical bit error rates and allows the use of higher speeds.

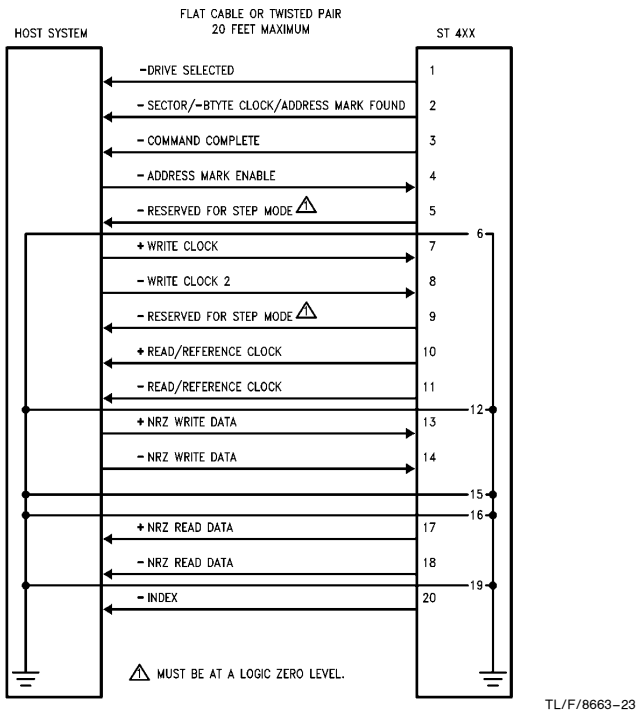
Step Mode

The ESDI step mode is essentially similar to the step mode in the ST506/412 family of drives, except for the NRZ data transfer. Only two of the seventeen signals change function in the control cable between ESDI step and ST412HP. READ GATE being added is the important change which enables the data separator on the drive to the controller. The data cable is considerably different. Differential drivers and receivers are used for signals like Write Clock and Read Clock and a few single ended lines are added like Cartridge Changed (for tapes) and other lines like Seek Complete, Index etc. The step mode pulse timings are comparable to that of the ST412HP. This enables switching between the two interfaces under software control, in the controller design.

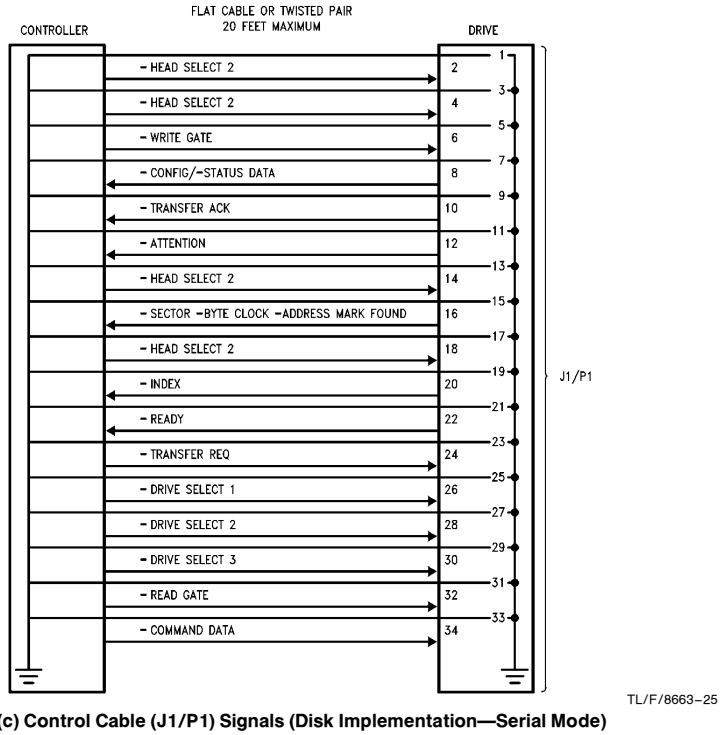


(a) Typical Connection, Multiple Drive System

FIGURE 1.11. ESDI (Enhanced Small Device Interface)



(b) Data Cable (J2/P2) Signals (Disk Implementation—Serial Mode)



(c) Control Cable (J1/P1) Signals (Disk Implementation—Serial Mode)

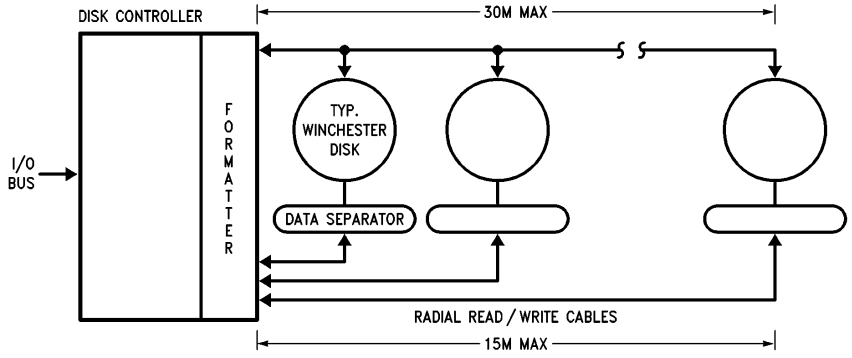
FIGURE 1.11. ESDI (Enhanced Small Device Interface) (Continued)

Serial Mode

Serial mode ESDI is a definite improvement over the interfaces discussed. As the name implies, communication from the controller to the drive takes place on the COMMAND DATA line of J1/P1 in conjunction with the handshake signals TRANSFER REQUEST and TRANSFER ACKNOWLEDGE. Communication from the drive to the controller takes place on the CONFIG-STATUS line of J1/P1 in conjunction with the handshake signals. Each bit of the 16-bit command or status word is handshaked across the interface. The hardware changes between EDSI serial and step modes, have several control lines redefined. The disk drive's microprocessor interprets commands like SEEK (seek to a cylinder), RECALIBRATE (seek to track 0), REQUEST STATUS and REQUEST CONFIGURATION, which provide the controller with standard status and configuration information of the drive like the number of heads, number of tracks, sectors per track, bytes per track, command data parity fault, write fault etc. Hence the controller can configure itself to the drive connected to it and can send the data to the host if desired. Thus ESDI serial mode offers big benefits and is rapidly gaining popularity in higher performance hard disk drives.

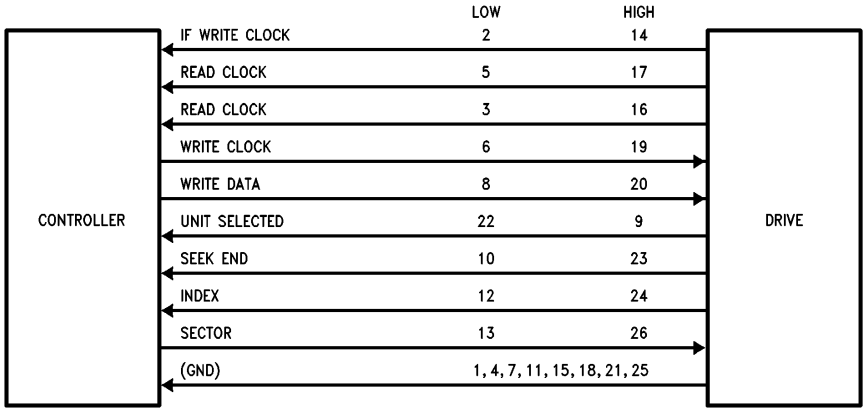
STORAGE MODULE (SMD) INTERFACE (ANS X3.91M 1982)

The Storage Module Interface was originated by Control Data Corporation around 1972. It has been extremely popular with 8″–14″ drives. However, as it is expensive and hardware intensive and because of competition due to ESDI and SCSI, it is not very popular with 5¼″ drives. *Figure 1.12* gives the data and control cable assignment.



TL/F/8663–26

(a) Typical Connection

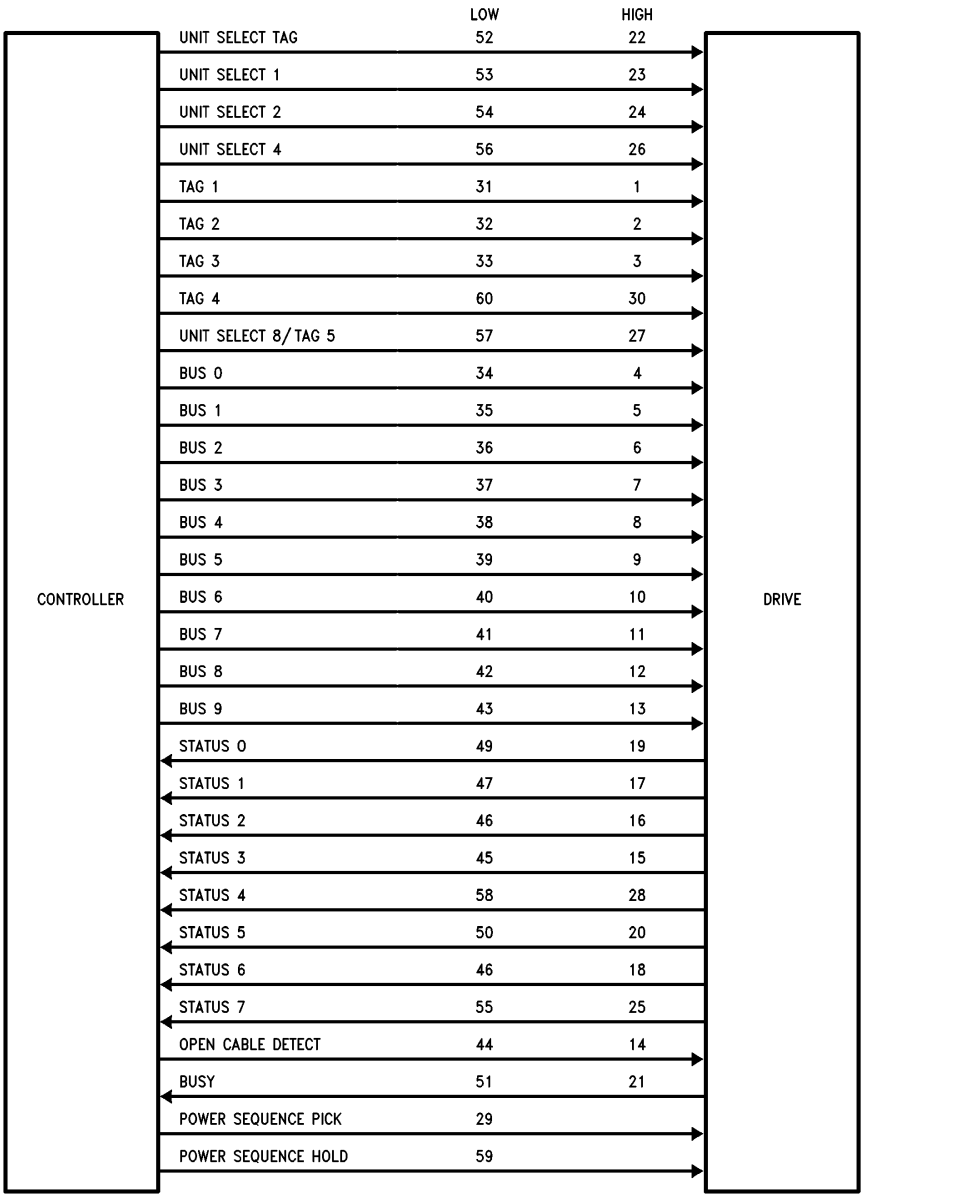


TL/F/8663–28

B = Cable

(b) Data Cable

FIGURE 1.12. Storage Module (SMD) Interface (ANS X3.91M 1982)



A = Cable

(c) Control Cable

Storage Module (SMD) Interface (ANS X3.91M 1982) (Continued)

Features

- Bit serial digital data transfer (Data Separator in drive)
- Relatively high transfer rate (9.67 Mbits/s is common in older 14″ drives and newer 8″ drives, new 10.5″ and 14″ drives are typically about 15 Mbits/s)
- Dominant de-facto standard for 14″ OEM disk drives; virtual basis of OEM disk controller industry. Widely used by minicomputer system manufacturers.
- Differential signals
- 23 required plus 8 optional control bus signals, 7 required plus 2 optional read/write cables
- Parallel control bus, but radial read/write cables, one per drive
- Incorporates error recovery facilities
- Includes power sequencing for multiple units
- Approved ANS X3.91 1982

1.5.2 Intelligent Disk System Interfaces

These are high level interfaces which result in the complete disk controller being situated on the drive and the interface to the host is through a special bus. Their chief advantage is nearly complete device transparency to system hardware and software, also lower system overhead for disk control and higher speeds. A well defined protocol is used for communication with the host system. Some of the popular Intelligent disk system interfaces are discussed below in brief.

SHUGART ASSOCIATES SYSTEM INTERFACE (SASI)

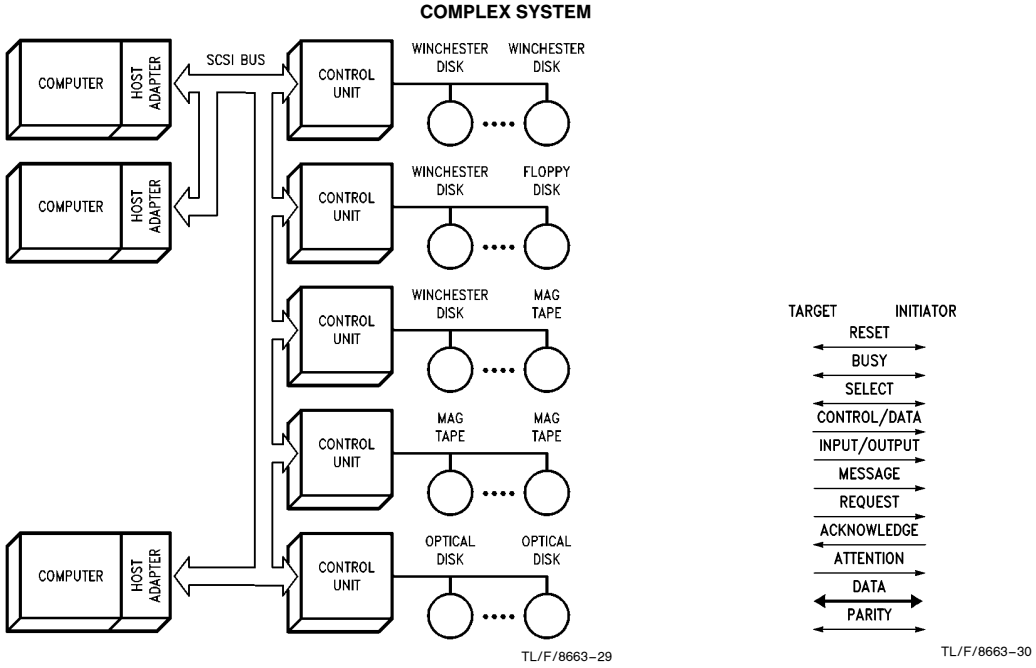
SASI was introduced by Shugart around 1980. The overall objective was to make it easier for computer systems to talk to disk drives. SASI defines the logical level and all lower interface levels, down to the 50-pin connectors and ribbon cable. Eighteen lines are used for signals, nine for data and nine for control. The data lines consist of a single octet with an odd parity bit. The control lines include a two-wire handshake and various lines to put the bus in different transfer modes or phases. The interface is always in one of the five phases:

Bus Free, Arbitration, Selection, Reselection, Data

Eight devices are allowed but only one host or ''initiator'' is allowed. So a maximum SASI system will consist of an Initiator and seven target devices. All signals in the interface are open collector driven. The ANSI standard version of this interface is the SCSI (Small Computer System Interface).

SMALL COMPUTER SYSTEM INTERFACE (SCSI)

The Small Computer System Interface (SCSI) was formed from the SASI framework and ANSI has standardized it under X3T9.2. The interface consists of a single cable that is daisy-chained to other SCSI units. It will accommodate not only disk drives but also tapes, printers and other devices and is potentially a universal peripheral port for small systems. The SCSI system could potentially be a single initiator - single target system or a single initiator - multiple target system or a multiple initiator - multiple target system as shown in *Figure 1.13(a)*. The SCSI bus signals are shown in *Figure 1.13(b)*. The cable consists of transfer handshaking and status signals in addition to an 8-bit data bus. Information is exchanged on the bus via a set of higher level commands sent by the host.



Up to 8 SCSI DEVICES can be supported by the SCSI bus. They can be any combination of host CPUs and intelligent controllers.

(a) Multiple Target-Multiple Initiator **(b) SCSI Cable**

FIGURE 1.13. Small-Computer System Interface

Features

- Connects up to 8 computers and peripheral controllers
- Maximum rate up to 1.2 Mbyte/sec asynchronous, 4 Mbytes/sec synchronous: suitable for floppy disks, all 5.25″ and 8″ Winchester disks, medium performance 8″ and 14″ disks, and tape drives
- Relatively high level peripheral command set
- Single ended version: 50-conductor flat ribbon cable, up to 6 meters, 48 mA drivers
- Differential version: 50-conductor flat or twisted pair cable, up to 25 meters, EIA RS-485
- Distributed bus arbitration
- Includes command sets for common peripherals
- Products now widely available include: disk drives with integral controllers, SCSI to ST506, SMD, Floppy, SCSI to S-100, Multibus®, IBM PC™, VME™, Unibus™, TRS-80™, and Q-BUS™ Adapters, and VLSI bus protocol and disk controller chips

SCSI makes no hardware changes compared to the SASI but adds several features which are discussed below.

Arbitration

This allows multiple Initiators to talk to multiple targets in any order, to a maximum of eight nodes only.

Reselection

This allows a target to disconnect from the Bus while it is getting data and reconnect to the proper Initiator when it has found it. This results in efficient utilization of the bus because other nodes can use it during the relatively long seek time of the disk drive or search time of a tape drive.

Synchronous Mode Transfer

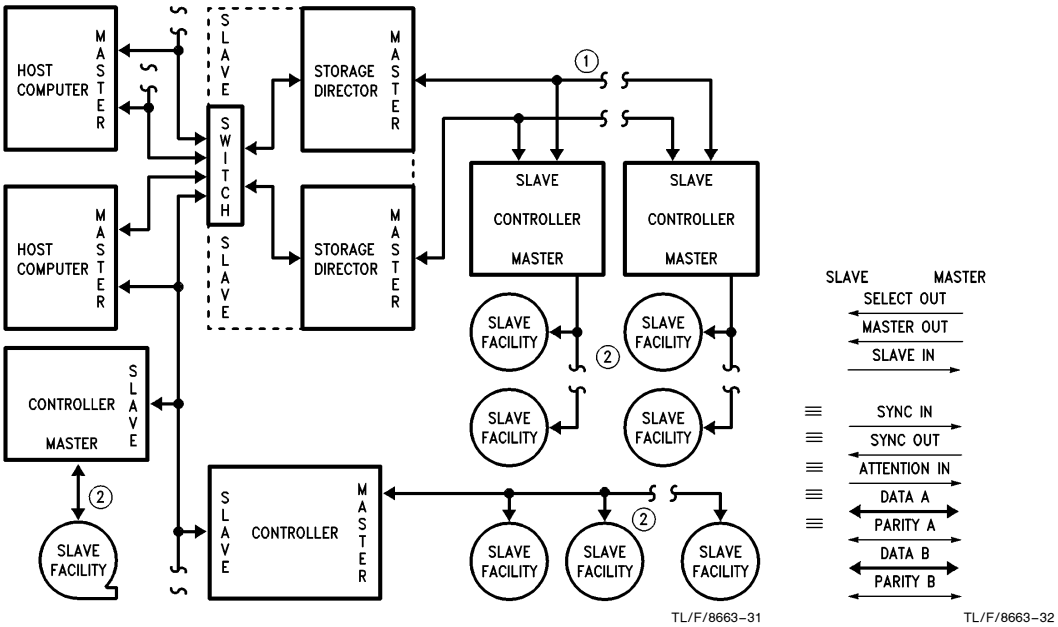
This speeds data transfer to a maximum of 4 Mbytes/sec (from an asynchronous maximum of approximately 1.2 Mbytes/sec).

Differential Transceivers

These boost the maximum length of the interface from 6m to 25m.

Extended Command Set

This set includes expanded large block addressing (from 2^{21} to 2^{32} blocks) and ''Inquiry'' type commands for self configuring controllers. It also handles tape drives, printers, processors, optical disks and read-only optical disks. There is also room for a ''vendor unique'' command set, i.e. commands unique to the device.



(a) IPI Configuration

(b) IPI Cable Signals

FIGURE 1.14. Intelligent Peripheral Interface

INTELLIGENT PERIPHERAL INTERFACE(IPI)

This is the ANSI standard X3T9.3 and is an additional peripheral bus, having higher performance than SCSI. *Figure 1.14(a)* shows the orientation of the IPI system and the IPI port signals are shown in *Figure 1.14(b)*.

Features

- Connects master to a maximum of 8 slaves
- Can be used at various levels in the system as shown
- Two 8-bit buses (in and out) for commands and status speed protocol and status presentation for fast path switching
- 8- or 16-bit parallel transfers
- 24 signals
- Several electrical options; fastest allows 10 Mbytes/sec through a 75 meter cable
- Offers both ''intelligent'' and ''device level'' command definitions, command and data handshaking
- 50-pin cable ground increases noise immunity

The IPI interface comprises four levels. Level 0 consists of cables, connectors and drivers/receivers. Level 1 consists of state machine and bus protocol. Features of Level 2 are device specific commands, timing critical, physical addressing, physical volumes, command parameters and bus control commands. Features of Level 3 are device generic commands, timing independent, buffered, command stacking, queing, limited specific commands, logical addressing and physical volumes. Messages are transmitted in packets.

IPI derives its higher performance from a faster handshake and a wider data bus. Two octets, each with a parity line, make up the data interface. Six control lines fill out the interface of 24 signals. There is one master allowed and up to eight slaves on a daisy-chained cable. This master to slave interface is a parallel one and hence IPI 3 could be used, (point 1) in *Figure 1.14(a)*. Each Slave can address up to 16 Facilities, like disk drives. The Slave-to-Facility interface may be IPI 2, (point 2) in *Figure 1.13(a)*, or a lower level interface such as ESDI. Data can be moved at 5 Mbytes/sec in asynchronous mode, 10 Mbytes/sec in synchronous mode. The interface supports various driver options with maximum cable lengths ranging from 5 meters to 125 meters.

1.5.3 Other Disk Interfaces

There are many other ANSI standardized interfaces which were the outcome of the interface standards discussed above. Some of these are disk level while some are intelligent interfaces. A brief discussion follows, also refer to *Figure 1.15*. Detailed descriptions can be found in the appropriate ANSI document.

FLEXIBLE DISK INTERFACE (ANS X3.80)

Features

- American National Standard Interface between flexible disk cartridge drives and their host controllers
- 50-wire flat ribbon cable (8″ disk) or 34-wire (5¼″ disk)
- Bit serial encoded FM or MFM data transfer from/to Read/Write electronics (data separator in controller)
- Modest data transfer rate (100 kbyte/s)
- Very widely used by the industry; based on a de facto standard. Supported by most 8″ and 5¼″ drives; also used by some, but not all micros (less than 4″) floppy drives.
- Seeks track by track, one step per pulse
- Single ended signals
- Change has been submitted to identify high density 5¼″ drives
- Approved ANS X3.80 - 1981

RIGID DISK INTERFACE (ANS X3.101—1983)

With the emergence of the 8″ rigid-disk drive, there was a strong industry push for a new interface standard with broader applicability than the SMD, which would allow for self-applicability than the SMD, which would allow for self-reconfiguring controllers, as different devices were attached. As a result the ANSI Rigid Disk Interface came into existence.

Features

- Optimized for relatively high performance small winchester disks
- Bit serial digital data transfer (data separator in drive)
- Relatively high transfer rate possible (up to 10 Mbits/s with low cost option, up to 16 Mbits/s with high performance option)
- 50 conductor ribbon cable
- Class A: 24 and 40 mA single-ended control bus signals, 20 mA differential serial data and clock lines
- Class B: 100 mA single-ended control bus signals, 40 mA differential serial data and clock lines
- 22 single-ended plus 4 differential signals
- Byte parallel command bus
- Relatively high level command set
- Approved ANS X3.101—1983

Peripheral Bus Interface

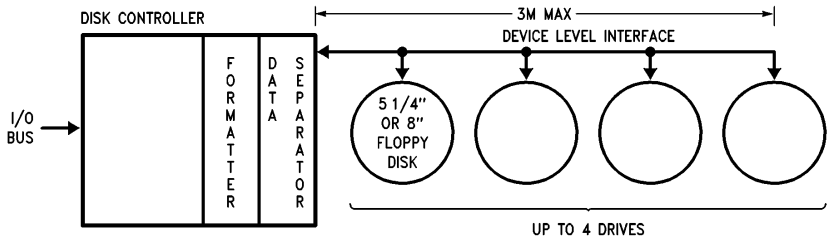
- Connects computer to peripheral different controllers
- Block transfer rather than word transfer orientation
- No provision for memory address on bus
- Longer distances than backplane
- Interface hides many device characteristics from software
- Peer to peer multi-master protocol may be called a ''system bus''

Device Interface

- Specific to particular device type
- Between controller and device
- Often serial data transfer
- User device interchangeability not always certain
- Very widely used by industry

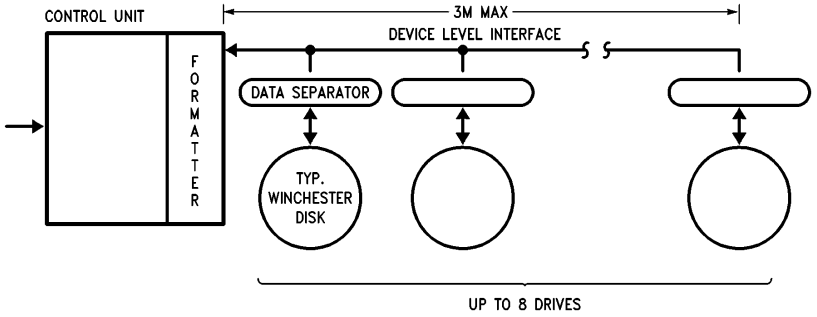
This interface has not gained acceptance and has been superseded by ESDI

Flexible Disk Interface (ANS X3.80—1981)



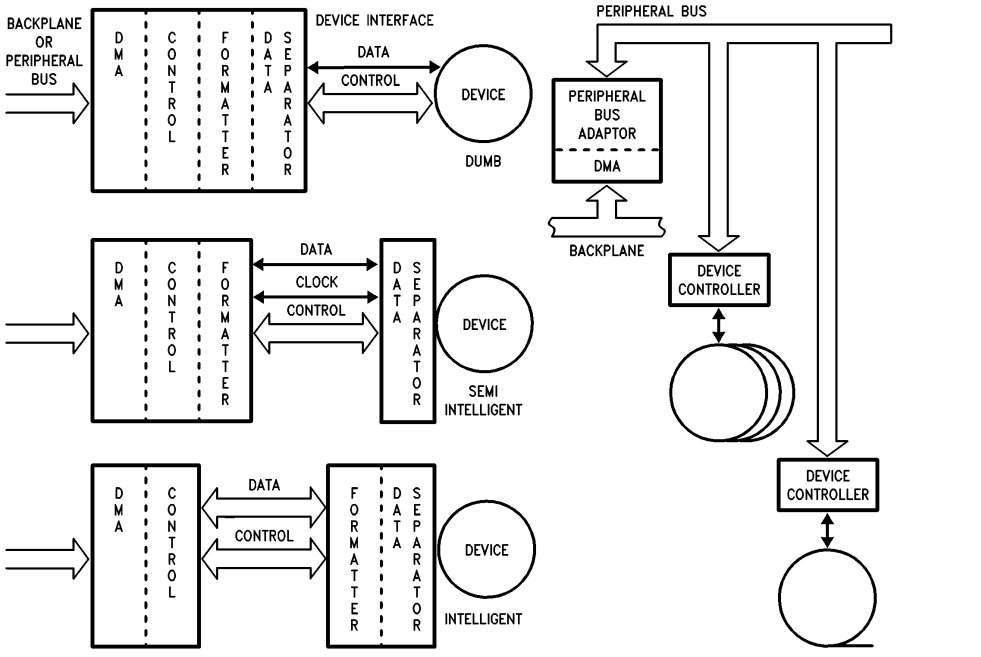
TL/F/8663–33

Rigid Disk Interface (ANS X3.101—1983)



TL/F/8663–34

DEVICE INTERFACE **PERIPHERAL BUS**



TL/F/8663–35

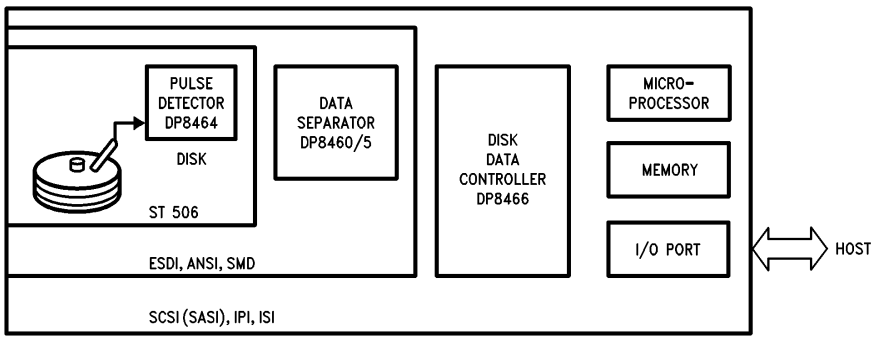
FIGURE 1.15. Other Drive Interfaces—Typical Connections

1.5.4 Universal Plug

Although the concept of a standard interface is appealing, the desire to gain industry recognition and lock in customers, coupled with valid technical improvements, will continue to spawn new interfaces at regular intervals. In systems where a maximum of one or two drives are required or where configuration flexibility justifies the higher cost of standard attachments, high-level interfaces will attach to a ''universal'' system plug. Here controllers will be integrated into the devices themselves, while in cost effective multiple drive systems, the connection of device level interfaces to system specific controllers is expected to continue.

1.6 ELEMENTS OF DISK CONTROLLING ELECTRONICS

Disk controller chips in the market today are complex VLSI chips and perform a multitude of functions. In fact they take care of most of the tasks besides the task of data separation. National's Disk Data Controller DP8466 is one such chip. It takes care of serialization, deserialization, data encoding, DMA transfer, error detection and correction, and pattern recognition to determine type of compensation required. The Disk Data Controller DP8466 is discussed in detail in the following sections. National's Data Separator chip DP8465, together with the Disk Pulse Detector DP8464, comprise the chip set for the disk controlling electronics. If RLL encoding is used, then National's DP8463 2,7 ENDEC could be integrated into the system. *Figure 1.16* shows the place of these chips in the disk data and control path. It also shows the separation lines of the components on the drive and controller for the various interface standards.



TL/F/8663–36

FIGURE 1.16. Typical Disk Controller System Configuration Showing the Interface Points with Respect to the Various Standards

CHAPTER 2 DP8464B HARD DISK PULSE DETECTOR

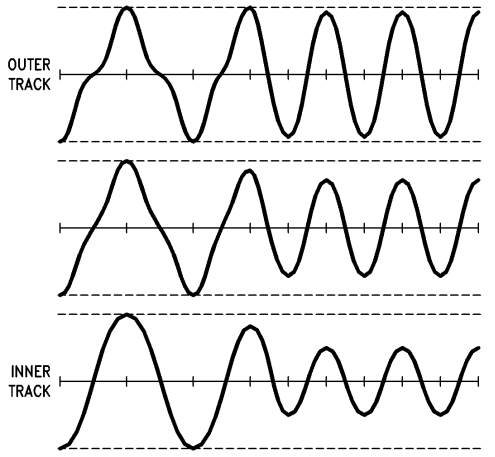
2.0 INTRODUCTION

The standard Winchester disk drive available today uses a magnetic film platter, a ferrite head and MFM coding. This combination produces relatively wide pulses off the disk which can be detected using a simple time-domain filter technique. However, as disk manufacturers strive for higher density and data rates, they must turn to new technologies such as plated media, thin film heads and run length limited codes (such as the 2,7 code). Unfortunately, these technologies produce more complex pulses off the disk which require more sophisticated pulse detection techniques. The DP8464B utilizes a separate time and gate channel which can detect the peaks in these complex waveforms.

2.1 BACKGROUND OF PULSE WAVEFORM DETECTION

Data on the disk is stored as a series of magnetic domains recorded on concentric circular tracks. To read the data, the head arm assembly brings the head directly above the track on the rotating disk. As previously recorded flux reversals pass under the head, a small signal will be induced. The signal from the disk is therefore a series of pulses, each of which are caused by flux reversals on the magnetic medium. The pulse detector must accurately replicate the time position of the peaks of these pulses. This task is complicated by variable pulse amplitudes depending on the media type, head position, head type and the gain of the Read/Write amplifier. Pulse amplitudes may vary on any one track if the distance between the head and the media varies as the disk rotates. Additionally, as the bit density on the disk increases, significant bit interaction occurs resulting in decreased amplitude, pulse distortion and peak shift.

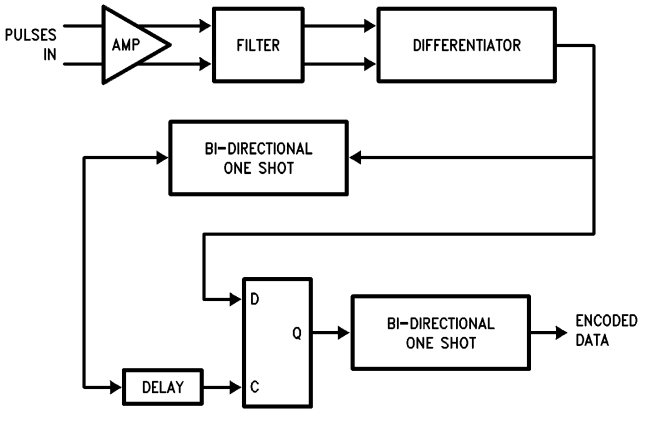
Traditionally MFM code has been used to encode digital information on the disk surface. MFM code uses the limited frequency range of F to 2F as illustrated in *Figure 2.1.* Such a system can use a special self gating circuit for the pulse detector as shown in *Figure 2.2.* Pulses from the inner track (the bottom waveform in *Figure 2.1*) are almost sinusoidal so the peak detector can simply differentiate the waveform to determine the peak positions. On the outer track however, the pulses have a small amount of shouldering as shown in the top waveform in *Figure 2.1.* The problem is that any noise occurring during these very narrow shoulders can be incorrectly interpreted as signal peaks.



TL/F/8663–37

Typical MFM waveforms on oxide media with ferrite heads. Since there is only slight shouldering on the outer track, the traditional self gating (de-snaker) pulse detector can be used.

FIGURE 2.1



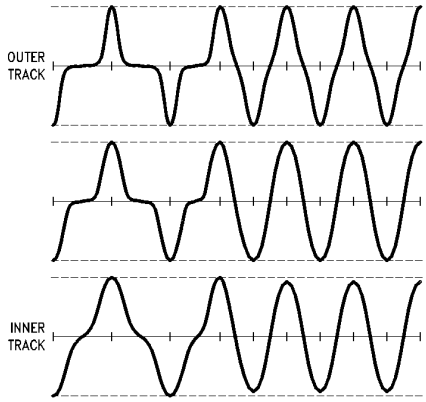
TL/F/8663–38

Self gating circuit (de-snaker) traditionally used for pulse detection of MFM code on oxide media with ferrite heads. The amount of delay in the clock line must exceed the maximum amount of shouldering.

FIGURE 2.2

The self gating circuit places a fixed delay and bi-directional one shot between the output of the differentiator and the clock input. The amount of this delay is selected to be longer than the worst case shouldering. When shoulder-induced noise occurs, the D input to the flip-flop will change states. However, by the time the clock occurs, the noise will no longer be present and the D input will have returned to its previous state. The flip-flop will therefore ''clock'' in the previous data. The output Q will not change states due to these narrow shoulder-induced noise pulses. This fixed delay in the clock line is called a time domain filter. This circuit is also known as a ''de-snaker'', named after the snake appearance of the waveform which exhibits slight shouldering. This ''de-snaker'' circuit works very well with waveforms which exhibit only slight shouldering. Unfortunately, the new methods used to increase the disk capacity do not produce such simple pulse patterns.

There are several methods of increasing the disk capacity. This includes plated media, thin film heads, and run length limited codes. All of these techniques result in narrow pulses with increased shouldering. An example of these slimmer pulses is shown in *Figure 2.3*. Instead of the slight shouldering present in the MFM example, the signal returns to the baseline between pulses. Since the shouldering is so extensive, the ''de-snaker'' technique simply will not work here. If a long delay were used to correct the shouldering present in the top waveform, it would not capture the pulses at the highest frequency.



TL/F/8663–39

Pulse waveforms for high resolution technologies. The large shouldering on the outer tracks precludes the use of the de-snaker pulse detector.

FIGURE 2.3

Detecting pulse peaks of waveforms of such variable characteristics requires a means of separating both noise and shouldering-induced errors from the true peaks. The old self gating circuits (such as the ''de-snaker'') will not work with the new techniques to increase the disk capacity. Hence the need for a circuit that includes a peak sensing circuit with an amplitude sensitive gating channel in parallel. Such a circuit is a key feature of the DP8464B Pulse Detector.

2.2 DP8464B FEATURES

Certainly a key feature of the DP8464B is the combination of a peak sensing circuit with an amplitude sensitive gating channel in parallel which allows the DP8464B to accurately detect the peak of waveforms that preclude the use of the traditional ''de-snaker'' circuit. The DP8464B, however, has many other features that make it ideal for the disk drive read channel.

Another key feature of the DP8464B is a wide bandwidth automatic gain controlled (AGC) amplifier. The automatic gain control removes the signal level variations of the read signal. The amplifier's wide bandwidth (20 MHz) insures that timing errors will not be introduced by the amplifier's pole.

The DP8464B offers considerable flexibility to the user, allowing him to tailor various operating characteristics to his specific needs. In particular, the user can set the frequency response of the differentiator, the width of the pulses on the encoded data output, the amount of hysteresis in the gating channel, the signal amplitude at the output of the gain controlled amplifier and the overall frequency response of the system and AGC. This kind of flexibility is provided by strategically placing pinouts at key points throughout the circuit. Differential signal paths were utilized whenever possible to minimize effects of power supply noise and external noise pickup. The IC can be effectively disabled when the disk drive is in a write mode, thus preventing saturation of the input amplifier and preventing disturbance of the AGC level.

The IC is powered from a single +12V supply (which is standard in most drives) and has an internal regulator and separate analog and digital grounds in order to properly isolate the sensitive analog circuitry. It is presently offered in a 24-pin DIP but will soon be made available in a 28-pin PCC surface mount package.

The DP8464B is fabricated on an advanced low power Schottky process which allows the part to handle data rates up to 15 Megabits/sec., 2, 7 Code.

2.3 THE DP8464B HARD DISK PULSE DETECTOR OPERATION

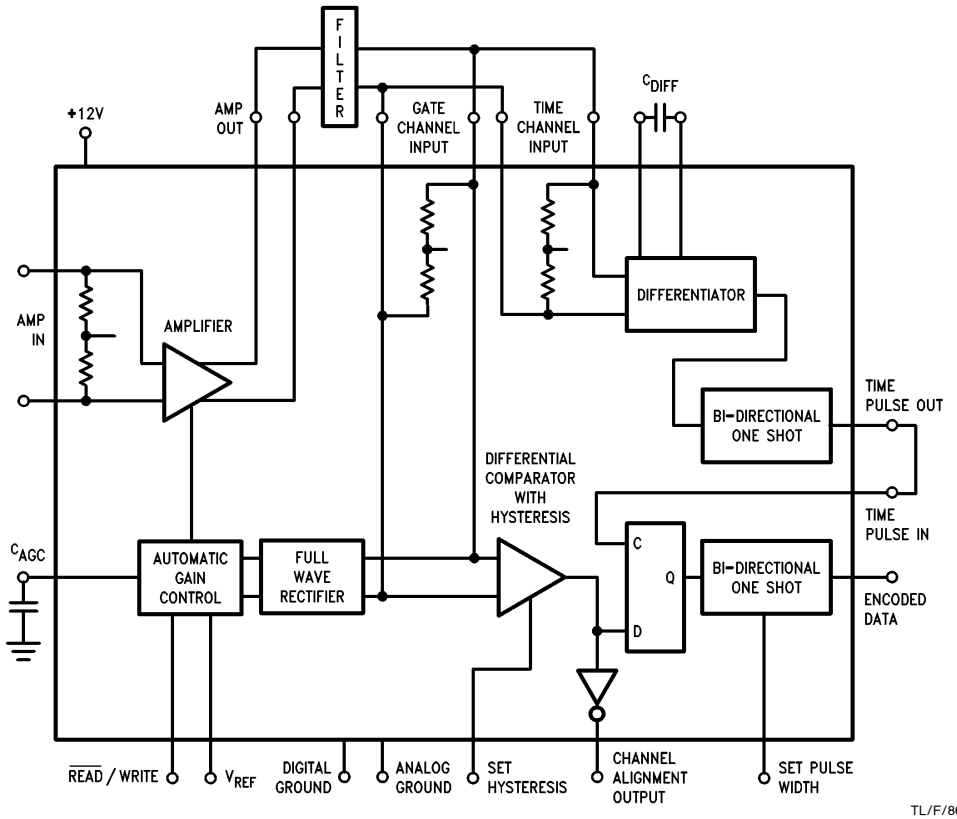
The main circuit blocks of the DP8464B are shown in *Figure 2.4*. The circuit consists of three main sections: the Amplifier, the Time Channel and the Gate Channel. The Amplifier section consists of a wide bandwidth amplifier, a full wave rectifier and the Automatic Gain Control (AGC). The Time Channel is the differentiator and its associated bi-directional one shot, while the Gate Channel is made from the Differential Comparator with Hysteresis, the D flip-flop and its following bi-directional one shot. Also, there is special circuitry for the Write mode. To better understand the circuit operation, let's discuss each section separately.

2.3.1 Gain Controlled Amplifier

The purpose of the Amplifier is to increase the differential input signal to a fixed amplitude while maintaining the exact shape of the input waveform. The Amplifier is designed to accept input signals from 20 mV_{pp} to 660 mV_{pp} differential and amplify the signal to 4 V_{pp} differential. The gain is therefore from 6 to 200 and is controlled by the Automatic Gain Control (AGC) loop.

2.3.2 Time Channel

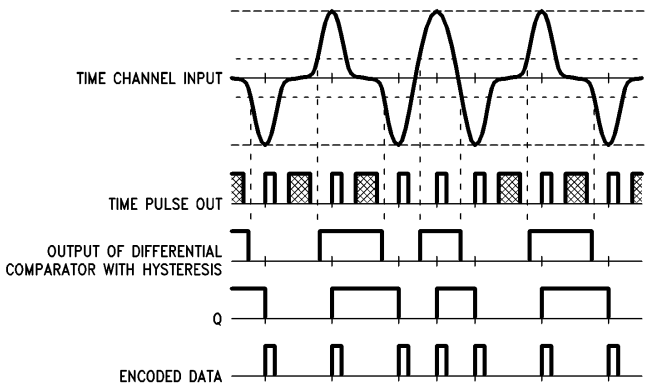
The peak detection is performed by feeding the output of the Amplifier through an external filter to the Differentiator. The Differentiator output changes state when the input pulse changes direction, generally this will be at the peaks. The Differentiator can also respond to noise near the baseline, in which case the Gating Channel will inhibit the output pulse (as discussed in the Gate Channel section). The purpose of the external filter is to bandwidth limit the incoming signal for noise considerations. Care must be used in the design of this filter to ensure the filter delay is not a function of frequency. The output of the Differentiator drives a bi-directional one shot which creates the Time Pulse Out.



TL/F/8663–40

Main circuit blocks of DP8464B. The three main sections are the Amplifier (amplifier and AGC), the Gating Channel (comparator with hysteresis and D flip-flop), and the Time Channel (differentiator and bi-directional one shot).

FIGURE 2.4



TL/F/8663–41

These signal waveforms illustrate the operation of the DP8464B. The noise in the Time Pulse Out (which occurs during the shouldering) simply clocks in old data present at the output of the Differential Comparator with Hysteresis. A bi-directional one shot at the Encoded Data output provides a rising edge representing the relative time position of the peaks at the Time Channel Input.

FIGURE 2.5

2.3.3 Gate Channel

While the actual peak detection is done in the Time Channel with the Differentiator, there is the problem of preventing the output data from being contaminated when the Differentiator responds to noise at the baseline. To prevent this, the signal is also passed through a Gate Channel which prevents any output pulse before the input signal has crossed an established level. This Gate Channel comprises a Differential Comparator with Hysteresis and a D flip-flop. The hysteresis for this comparator is set externally via the Set Hysteresis pin.

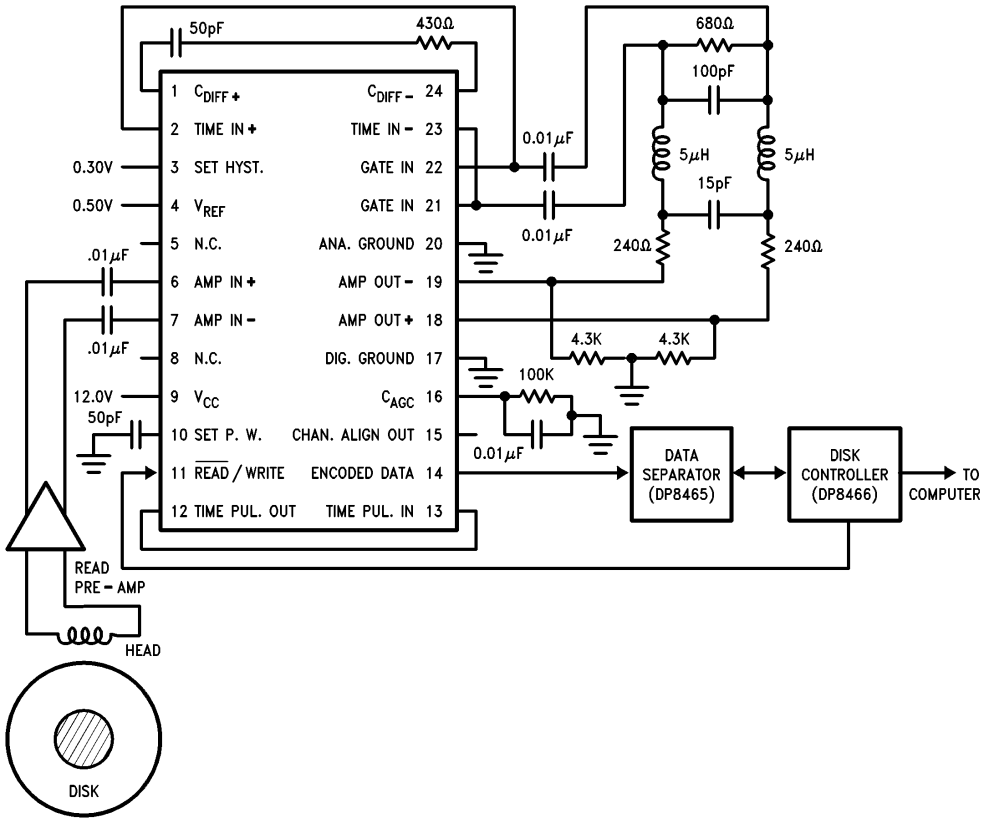
The operation of this Gate Channel is shown in *Figure 2.5*. At the top is a typical waveform which exhibits shouldering at the lowest frequency, and is almost sinusoidal at the highest frequency. This waveform is fed to both the Time and the Gate Channel. The hysteresis level (of abut 30%) has been drawn on this waveform. The second waveform is Time Pulse Out. While there is a positive edge pulse at each peak, there is also noise at the shoulders. Since Time Pulse Out is externally connected to the Time Pulse In, this output is therefore the clock for the D flip-flop.

The third waveform is the output of the Comparator with Hysteresis which goes to the D input of the flip-flop. The true peaks are the first positive edges of the Time Pulse Out which occur after the output of the comparator has changed states. The D flip-flop will ''clock'' in these valid peaks to the output bi-directional one shot. Therefore, the noise pulses due to the Differentiator responding to noise at the baseline just ''clock'' in the old data through the flip-flop so there is no noise pulse on Encoded Data.

The Q output of the flip-flop drives the bi-directional one shot which generates the pulses on Encoded Data. The positive edges on Encoded Data correspond to the signal peaks. The width of the data pulses can be controlled by an external capacitor from the Set Pulse Width pin to ground. This pulse width can be adjusted from 20 ns to 1/2 the period of the highest frequency.

Design Example

A typical system implementation of the DP8464B is shown in *Figure 2.6*. The DP8464B is driven from the Read/Write amplifier which is generally located very close to the actual Read/Write head. This amplifier is fixed gain and pre-amplifies the weak readback signal picked up by the head from the surface of the disk.



TL/F/8663–42

Shown here is a typical 10 Mbit/sec MFM disk drive application. Signals are picked off the disk by the Read/Write head and amplified by the pre-amplifier. The DP8464B further amplifies the read signal and accurately represents the time position of the peaks by the rising edge of the signal at the Encoded Data output pin. The Data Separator locks up to the signal at the Encoded Data output and provides decoded MFM data to the Disk Data Controller. The Disk Data Controller handles the interface between the disk drive and the computer.

FIGURE 2.6

The output of the DP8464B drives a data separator (such as the DP8465) which extracts the digital data from the encoded data supplied by the DP8464B. The data separator utilizes a phase-locked loop that locks onto the leading edge of the encoded data signal from the DP8464B.

In addition to extracting the digital data from the encoded data, the data separator synchronizes the digital data thereby removing any timing jitter present in the encoded data signal. The data separator implements this last function by opening up timing windows that bracket the encoded data signal. The encoded data signal need only appear within the window to be detected. For 10 Mbit MFM, the window is only 50 ns wide.

In order to guarantee that the drives have error rates on the order of 1 per 10 to the 10th power, bits read (industry standard), the leading edge of the encoded data must fall well within the 50 ns window. Because of this stringent criteria, there is little room for error with regard to the accuracy that the DP8464B can extract the relative time position of the peaks of the read back signal.

One form of timing error is jitter of time position of the leading edge of the encoded data signal. This jitter can be a result of noise output from the differentiator or noise pickup in other portions of the read channel. These timing errors will significantly affect the Encoded Data signal causing an increase in the error rate.

The filter that drives the differentiator is important in reducing the noise input to the differentiator. For this reason, a high order Bessel filter with its constant group delay characteristic can be used in this application. The constant group delay characteristic insures that the filter does not introduce any timing errors by distorting the signal and moving the position of the peaks. Often, this filter must be specifically designed to correct phase errors introduced by the non-ideal characteristics of the input read head. The typical −3 dB point for this filter is around 1.5 times the highest recorded frequency. Reducing the noise input to the differentiator will ultimately reduce the amount of noise jitter on the encoded data output.

Another way to reduce noise jitter is to limit the bandwidth of the differentiator with a series combination of resistor, capacitor and inductor in the external differentiator network and to use as large a differentiator capacitor as possible, thereby maximizing the differentiator gain. In order to prevent saturation of the differentiator, Schottky diode clamps were added to the differentiator output thus allowing the use of a larger differentiator capacitor.

An automatic gain control (AGC) circuit is used to maintain a constant input level to the gating channel (which is typically tied directly to the input of the differentiator). By maintaining a constant signal level at this point, we insure not only a large input level to the differentiator but also a constant level of hysteresis of the signal to the gating channel. Gain control is also necessary because the amplitude of the input signal will vary with track location, variations in the magnetic film, and differences in the actual recording amplitude. The peak-to-peak differential amplitude on the Gate Channel Input is four times the voltage set by the user on the V_{ref} pin.

The actual dynamics of the AGC loop are very important to the system operation. The AGC must be fast enough to respond to the expected variations in the input amplitude, but not so fast as to distort the actual data. A simplified circuit of the AGC block is shown in *Figure 2.7*. When the full wave rectified signal from the Amplifier is greater than V_{ref}, the voltage on the collector of transistor T1 will increase and charge up the external capacitor C_{agc} through T2. The maximum available charging current is 3 mA. Conversely, if this input is less than V_{ref}, transistor T2 will be off, so the capacitor, C_{agc}, will be discharged by the base current going into the Darlington T3 and T4. This discharge current is approximately 1 μA. The voltage on the emitter of T4 controls the gain of the Amplifier.

If the AGC circuit has not received an input signal for a long time, the base current of the Darlington will discharge the external C_{agc}. The Amplifier will now be at its highest gain. If a large signal comes in, the external C_{agc} will be charged by the 3 mA from T2, thereby reducing the gain of the Amplifier. The formula, $I = C^*(dV/dt)$ can be used to calculate the time required for the Amplifier to go from a gain of 200 to a gain of 6. For instance, if $C_{agc} = 0.05\ \mu F$, the charging current I is 3 mA, and the dV required for the Amplifier to go through its gain range is 1V, then

$$dt = (0.05\ \mu F^*\ 1V)/(3\ mA) \text{ or } 17\ \mu s.$$

By using the same argument, the time required to increase the Amplifier gain after the input has been suddenly reduced can be calculated. This time, the discharging current is only 1 μA, so

$$dt = (0.05\ \mu F^*\ 1V)/(1\ \mu A) \text{ or } 50\ ms.$$

This time can be decreased by placing an external resistor across C_{agc}.

2.4 READ/WRITE

In the normal read mode, the signal from the read/write head amplifier is in the range of 20 mV_{pp} to 660 mV_{pp}. However, when data is being written to the disk, the signal coming into the analog input of the pulse detector will be on the order of 600 mV. Such a large signal will disturb the AGC level and would probably saturate the amplifier. In addition, if a different read/write amplifier is selected, there will be a transient introduced because the offset of the preamplifiers are not matched.

A TTL-compatible $\overline{\text{READ}}$/WRITE input pin has been provided to minimize these effects to the pulse detector. When the $\overline{\text{READ}}$/WRITE pin is taken high, three things happen. First, the 1k resistors across the AMP IN pins are shunted by 300Ω resistors. Next, the amplifier is squelched so there is no signal on the Amp Output. Finally, the previous AGC level is held. This AGC hold function is accomplished by not allowing any current to charge up the external C_{agc}. The voltage across this capacitor will slowly decrease due to the bias current into the Darlington (see *Figure 2.7*) or through any resistor placed in parallel with C_{agc}. Therefore, the gain of the amplifier will slowly increase. All of these three events happen simultaneously.

When the $\overline{\text{READ}}$/WRITE input is returned low, the pulse detector will go back to the read mode in a specific sequence. First, the input impedance at the Amp In is returned to 1k. Then, after approximately 1.2 μs, the Amplifier is taken out of the squelch mode, and finally approximately 1.2 μs after that, the AGC circuit is turned back on. This return to the read mode is designed to minimize analog transients in order to provide stable operation after 2.4 μs.

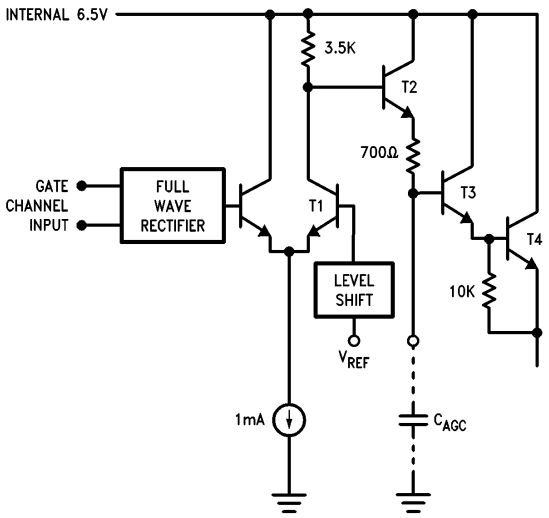
2.5 CONCLUSION

The push to higher disk capacities in increasingly smaller drives has forced drive manufactures to utilize different media, heads and encoding. Each of these changes render the traditional de-snaker pulse detector unusable. The DP8464B pulse detector utilizes a new detection technique that overcomes the limitations of the de-snaker. Furthermore, the DP8464B provides both gain controlled amplification of the pre-amplified readback signal and the ability to disable the circuit during write operations. The DP8464B is easily adapted to a wide variety of applications through selection of external components.

References

1. I. H. Graham, *"Data Detection Methods vs. Head Resolution in Digital Recording,"* IEEE Transactions on Magnetics Vol. MAG-14, No. 4 (July 1978).

2. I. H. Graham, *"Digital Magnetic Recording Circuits*," available through the University of Santa Clara, (California), bookstore (408) 554-4491.



The AGC Circuit senses the level of the signal at the Gate Channel Input and compares it to an externally set reference voltage. The signal that results from this comparison (at the collector of T1) charges C_{AGC}. The voltage across C_{AGC}, when buffered by T3 and T4, provides the gain control voltage to the input amplifier.

FIGURE 2.7

CHAPTER 3 DISK DATA SEPARATOR OVERVIEW (DP8460/61/62/65 AND DP8451/55)

3.0 INTRODUCTION—THE DATA SEPARATOR

As was discussed in the chapter on Disk Drive Technology (overview), the disk information which is recovered during a read operation ordinarily would have no defined phase relationship with respect to the timing within the host system. In order to establish a method of reconstructing a clock waveform with which the disk data may be entered into a shift register for deserialization and decoding, clock information is imbedded into the recorded bit pattern in any of a number of different ways by the various encoding schemes discussed in Chapter 1. The schemes vary in their efficiency of use of disk surface (bit density) and ease of recovery (challenge to the data separator), but they all are employed to achieve a mixture of clock and data within the same serial bit stream. It is then the function of the data separator to accurately extract this clock information from the bit stream and reconstruct a stabilized replica of the data, while at the same time remaining essentially immune to the random displacement of individual bits due to noise, media defects, pulse crowding and anomalies in the data channel.

From a ''black box'' standpoint, the data separator is fed a logic-level digital signal from a pulse detector (DP8464) within the disk head electronics (with positive transitions representing flux reversals on the media) and a read gate signal from the controller, and produces a reconstructed clock waveform along with a re-synchronized data output derived from the incoming disk pulse stream (see *Figure 3.1*). The regenerated clock and data signals have fixed timing relationships with respect to one another for use by subsequent shift register circuitry.

3.0.1 Separators and Synchronizers

The term ''data separator'' actually applies to a device which both regenerates a clock waveform from the bit stream as well as decodes (separates) the original NRZ data from the encoded disk data. This would include National's DP8461 and DP8465, both of which perform data synchronization with MFM-to-NRZ data separation, while including slight functional variations between devices. (The DP8460 initially released device is being replaced by the fully pin-for-pin compatible DP8465. The DP8461 pinout matches the DP8465, but is intended for use with hard and pseudo-hard sectoring only. Further details will be discussed later.) A device which performs clock regeneration and data synchronization without the separation function is simply called a data synchronizer. This would include the DP8451, DP8455, and the DP8462; again, there are functional differences between the devices, which will be discussed later. Additionally, the DP8461 and DP8465 also have outputs available which allow them to serve as data synchronizers, if desired (See Table 3.1).

The complete data separator circuit eliminates the need for external decoding circuitry but is dedicated to only a single code type. The data synchronizer requires an external decoding network but has the capacity to be used with any coding scheme. Since the MFM environment is being addressed in this design guide, the discussion in the remainder of this chapter will deal primarily with the integrated, MFM-type data separator. The PLL fundamentals being presented apply to all of the circuits.

3.0.2 Window

The data separator must establish what is called a ''window'' around the expected position of bits within the disk data stream. Windows are laid end-to-end in time by the data separator at a repetition rate (equal to the separator's VCO frequency) known as the disk code rate. Each window is an allotment of time within which a disk bit, if detected, will be captured and interpreted as if it had occurred exactly at the window center. This allows for a random displacement (jitter) of individual bits within the boundaries of the window with no apparent effect on the accuracy of the data recovery (error rate). Bits are displaced from a nominal position in a fashion which could be represented by a bell-shaped probability curve (see *Figure 3.2*), and it could be easily seen that for optimum performance, the window must be accurately centered about the mean of this curve. This is traditionally a difficult goal to achieve, and is essentially the primary responsibility of the data separator. Although various techniques have been employed to attain this goal, the phase-locked loop (incorporated within all National data separator/synchronizers) has proven to date to yield the most reliable and satisfactory results among all the synchronization methods, and its use is the standard approach taken within the disk industry.

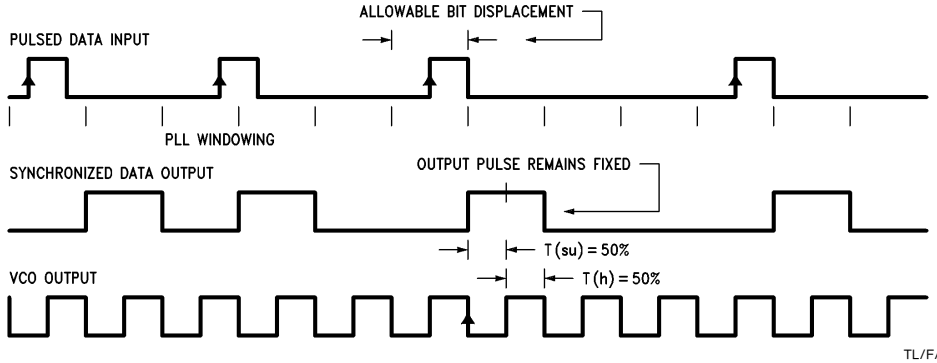
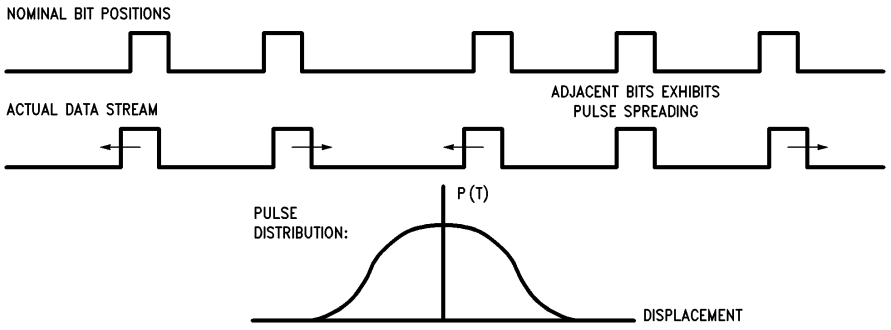


FIGURE 3.1. DP8460-Series Data Synchronizer Timing Relationships

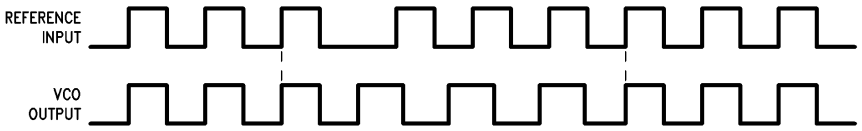
3.1 PHASE-LOCKED LOOP OVERVIEW

A phase-locked loop is a closed-loop control system which forces the phase from the output of a controlled oscillator to track the phase of an external reference signal. It consists of three essential elemental blocks; a phase detector, a loop filter, and a voltage controlled oscillator (VCO) (see *Figure 3.3*). The phase detector compares the phase of the reference input with that of the VCO output and generates an ''error'' signal at its output which is proportional to the sensed phase difference. This error signal is filtered by the loop filter (low-pass) to suppress any unwanted high-frequency components within the error signal, and is then fed to the VCO control voltage input. If the phase of the reference signal leads that of the VCO signal, the phase detector develops a positive error voltage across the loop filter, and the VCO responds by increasing its frequency (advancing phase). This continues until the phase error is eliminated and the control voltage returns to a quiescent value. When the reference signal lags that of the VCO, the adjustment occurs in the opposite direction until equilibrium (phase lock) is again obtained.

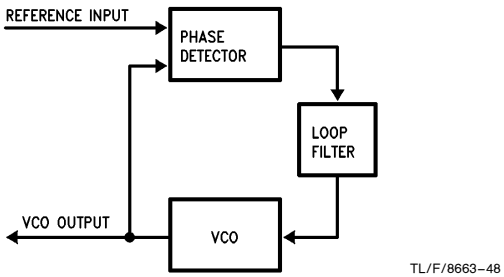


TL/F/8663–46

FIGURE 3.2. Data Jitter



TL/F/8663–47



TL/F/8663–48

A phase-locked loop is a system which forces the phase of the output from a voltage controlled oscillator (VCO) to track the phase of a reference signal.

FIGURE 3.3

3.1.1 PLL Dynamics

Much of the performance of the PLL, given adequate design in the major functional blocks, is determined by the loop filter. This includes (1) the ability of the PLL to rapidly (or slowly) track phase or frequency changes in the reference signal, (2) the ability of the loop to re-aquire lock after encountering a large frequency step at the reference input, and (3) the ability of the loop to exhibit stable behavior during operation.

TRACKING

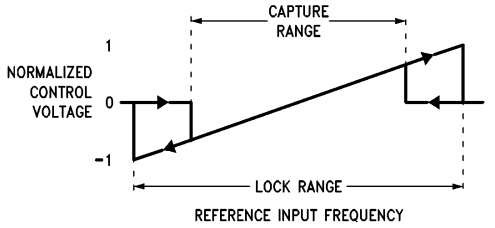
In many PLL applications, such as FM demodulation and frequency shift keying demodulation, rapid PLL tracking (high bandwidth) is a necessity. However, in the disk drive application, where (1) frequency changes (disk rotational speed variations) are gradual with respect to the data rate and (2) it is desirable to suppress response to instantaneous bit shift (jitter), a very slow tracking rate (low bandwidth) is necessary.

CAPTURE RANGE

The classical PLL is well able to maintain relative phase lock to a reference signal, but is unable to pass true difference-frequency information through its phase detector (this is true for the standard analog four-quadrant multiplier technique as well as for the gated-VCO technique employed in disk drive data separation applications). With this being the case, the PLL has a limited ability to re-establish lock when an instantaneous input frequency change occurs. The new frequency must lie inside a relatively narrow band on either side of the current VCO frequency, or re-lock will not occur (see *Figure 3.4*). This band is known as the capture range, and is a direct function of the passband of the loop filter, or more accurately, of the bandwidth of the PLL as a whole. A digital frequency discrimination technique, however, is employed in National Semiconductor's disk drive PLL's which provides an extended capture range and guarantees successful lock to the reference clock input and, as a chip dependent option, to the data as well. Consequences of having a limited capture range are discussed in National Semiconductor Application Notes AN414, AN415, AN416 (Also see ''Frequency Lock'' in section 3.2 of this chapter).

STABILITY

Mathematical analysis of the functional blocks of the PLL show a 1/s factor in the VCO; i.e., it behaves as a integrator, adding a pole to the transfer function. The loop filter adds at least one additional pole, resulting in a system which is, at minimum, second order in nature. Since the PLL is then a closed loop, minimum second order system, it has the potential for instability if improperly implemented. All of the dynamic characteristics of the PLL, however, can be controlled by loop filter selection, including bandwidth and capture range along with stability; thus great care must be taken in the selection of the loop filter in order to achieve the desired performance within the specific application.



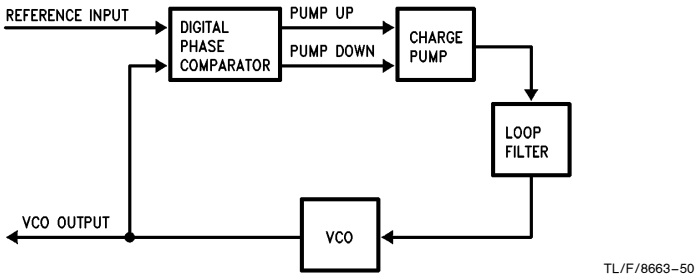
TL/F/8663–49

Capture range in a PLL is determined by the loop filter bandwidth.

FIGURE 3.4. Capture and Lock Range

3.2 THE PLL WITHIN A DISK DRIVE SYSTEM

Many methods have been employed in implementing each of the individual blocks within the phase locked loop (PLL), with techniques being customer-tailored to specific applications. Design methods include analog and digital configurations or a combination of both. For the application of a PLL within a disk drive data separator (and specifically regarding National's family of data separator/synchronizers), a combination of digital and analog block design has been found to provide the most efficient and reliable solution (see *Figure 3.5*). Here, since the waveforms to be compared are digital signals and the phase relationships are indicated by logical transitions (positive edges), the phase detector is comprised of a simple set of cross-coupled latches which produce ''pump-up'' (reference leads VCO) and ''pump-down'' (reference lags VCO) digital outputs. Since the filtering, however, is most easily and flexibly performed with passive analog components, the pump-up and pump-down signals are converted into gated sourcing and sinking currents, respectively, via an analog ''charge pump'' circuit, which is used to develop an error voltage across a capacitive loop filter (see *Figure 3.6*). This error voltage is used as a control potential for a variable rate relaxation oscillator (emitter coupled multivibrator VCO), whose oscillation is converted to digital signal levels again for use both at the phase detector input and as the regenerated clock waveform.



TL/F/8663–50

Goals:

1. Edge sensitive detector which eliminates dependence on waveform shape.
2. Unlimited capture range to ensure phase and frequency lock.
3. Zero phase difference when in lock to improve lock range.

FIGURE 3.5. Basic Disk System PLL

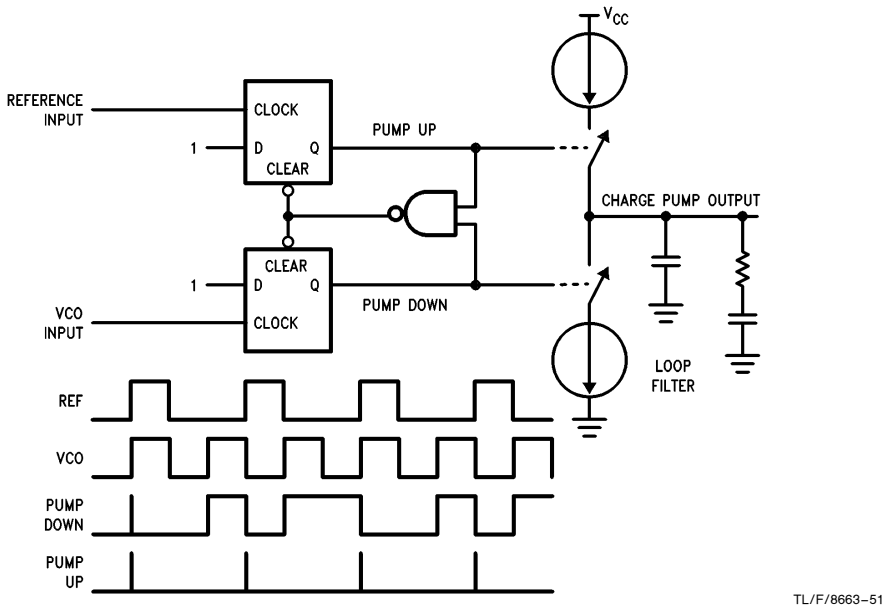


FIGURE 3.6. Digital Phase Detector

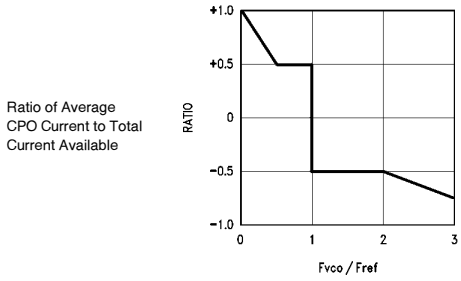
PHASE DETECTOR

The digital phase detector employed within National's data separator/synchronizer circuits is actually a true phase/frequency discriminator block capable of allowing a theoretically infinite capture range for the PLL (see *Figures 3.7–3.10*). Essentially, the capture range is limited only by the design constraints placed on the VCO's frequency excursion. In all of National's current disk PLL circuits (excluding the DP8460/50), this extended lock capability is employed while the circuit is in the non-read mode and the PLL is locked to a constant reference signal, guaranteeing proper lock recovery from any given mislock which may occur during a read operation.

PULSE GATE

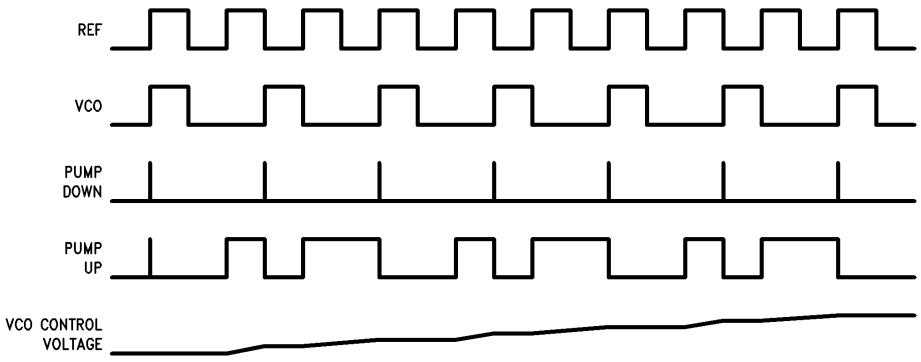
The data returning from the disk is not a periodic waveform, but instead has the possibility of bits either appearing or not appearing within assigned positions (windows) in the data stream (see *Figure 3.1*). The PLL is required to achieve and maintain lock to this pseudo-random pattern, despite the missing bits. This is analogous to the placement of teeth in a gear (data separator) which are ready to mesh with another gear (data stream), regardless of whether or not some teeth on the second gear (data) are occasionally missing (random data patterns).

In order to allow for the missing bits, the PLL employs a Pulse Gate circuit, which functions as follows: a data bit arriving at the PLL is sent to its corresponding input on the phase detector, and at the same time trips a gate which allows the next occurring VCO edge into the phase detector; the gate then closes following transmission of the VCO edge. If no data bit arrives, no comparison occurs and the VCO holds its frequency. Essentially, the PLL is attempting to align each data bit with the nearest occurring VCO edge, thus maintaining phase lock while frequency discrimination is suppressed (see *Figures 3.11* and *3.12*).



Ensures unlimited capture range. The digital phase detector ensures frequency acquisition by forcing the charge pump to always pump in the direction needed to make F_{VCO} equal to F_{REF}.

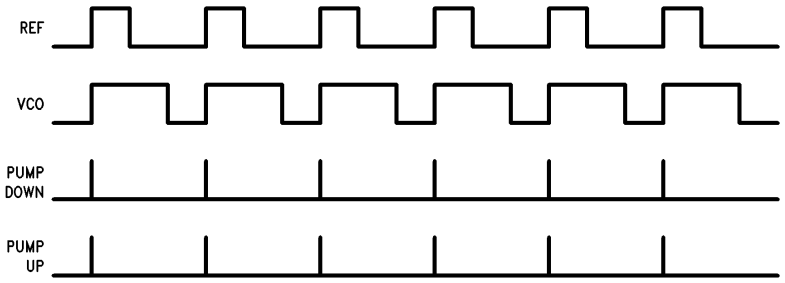
FIGURE 3.7. Digital Phase Detector



TL/F/8663–53

When the digital phase-lock loop is out of lock, the output of the phase comparator has a duty cycle which varies between 0 and 100%. The charge pump is active more than 50% of the average time but it only pumps current in the direction necessary to lock the VCO phase.

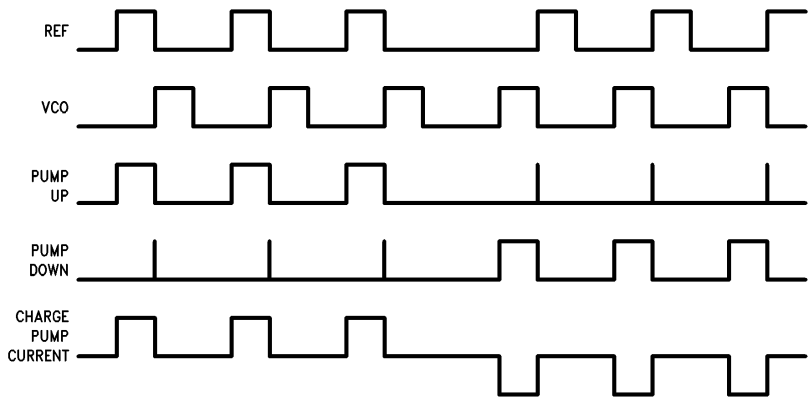
FIGURE 3.8. Digital Phase Detector



TL/F/8663–54

Once the VCO is locked to the reference signal, the only phase difference which occurs will be that required to pump enough charge to compensate for any leakage current in the charge pump, loop filter or bias current in the VCO control input.

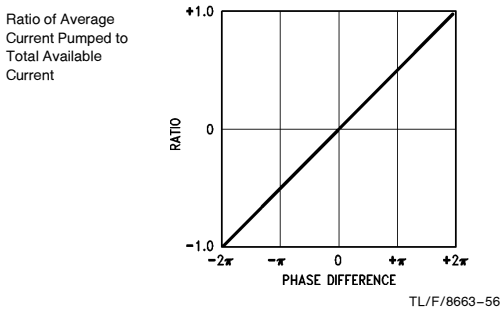
FIGURE 3.9. Digital Phase Detector



TL/F/8663–55

Effect of phase difference on digital phase detector when the VCO and reference frequencies are equal. The net charge pumped during each period is equal to the product of the charge pump current and the time difference between the phase comparator inputs.

FIGURE 3.10. Digital Phase Detector

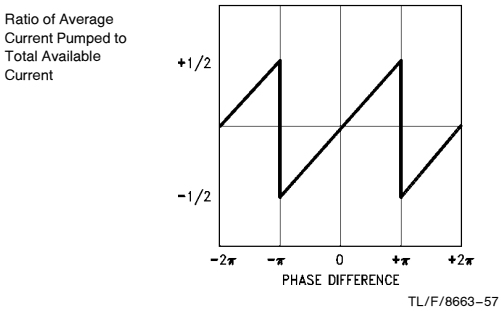


Phase detector gain when in lock

When the loop is in lock the net charge (Q) pumped during each period is equal to the product of the charge pump current and the time difference between the phase comparator inputs.

$$Q = I_{cpo} (T_{up} - T_{down})$$

FIGURE 3.11. Digital Phase Detector Gain without Pulse Gate



Each vertical transition represents a window boundary.

FIGURE 3.12. Digital Phase Detector Gain with Pulse Gate

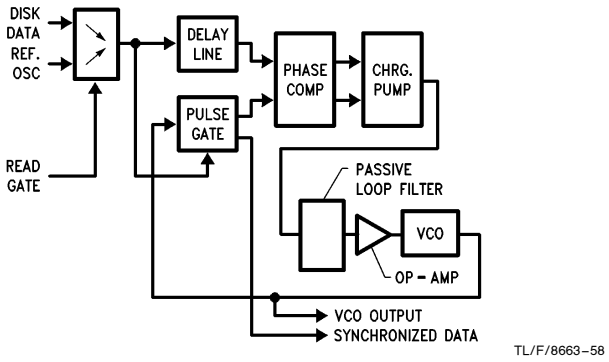


FIGURE 3.13. Disk Data PLL

DELAY LINE

In the generation of the symmetrical bit-detection window mentioned previously, a delay line is employed as shown in *Figure 3.13*. The incoming data is allowed to trip the VCO pulse gate immediately upon arrival, but is allowed into the phase detector only after it traverses the delay line. To understand the purpose for the delay line, first consider the case where the delay line is not present; given proper PLL lock, the VCO would align its edge to occur exactly at the same time as the arrival of each data bit. Any bits shifted early would cause a small VCO phase-advancing correction within the loop, which would be desirable. However, any bits shifted even a very slight amount late would arrive after the current VCO edge had passed and been suppressed by the pulse gate. The bit would then have to be compared to the subsequent VCO edge instead of to the current edge, producing an erroneous phase correction. With the delay line in place and set to delay the data bit by one-half of the VCO period, the data bit would first trip the VCO gate and then spend one-half of the VCO period traversing the delay line before it reached the phase detector. Given the loop is in lock, both the delayed bit and the VCO edge arrive at the phase detector at exactly the same time. If the bit were early up to one-half VCO cycle, it would still gate the appropriate VCO edge through, and produce an appropriate phase correction at the phase detector. Also, if the bit were late up to one-half of the VCO cycle, it would again still gate the appropriate VCO edge through to the phase detector, as well as produce the appropriate phase correction. Thus, the net effect of the delay line is to allow the incoming data bit to shift either one-half cycle early or one-half cycle late while yet maintaining a proper comparison to the appropriate VCO edge.

WINDOW ACCURACY

As mentioned previously, the integrity of the window alignment is crucial in maintaining an acceptable system error rate. It can be easily seen that accuracy in the delay line is critical in achieving this alignment. This has traditionally made the implementation of a delay line a costly design challenge. Within National's data separator/synchronizer

circuits, a proprietary technique has been employed which extracts precise timing information from the 2F CLOCK input waveform, and uses this information to regulate the timing within an on-chip silicon delay line. This in itself has unique advantages since the 2F source is either (1) a highly accurate crystal oscillator source, or (2) derived from the disk servo clock which tracks the data rate and consequently allows the delay line to adjust its delay accordingly. Completely dependent on this 2F signal for timing information, the delay remains independent of variations in its associated external components, power supply, temperature, and silicon processing. It requires no adjustment (although fine tuning is optional on the DP8462), and its accuracy is guaranteed within the window tolerance specification for the device.

FREQUENCY LOCK

The frequency discriminating capability of the phase detector within National's data PLL circuits can be employed to great advantage if used appropriately. It is brought into play simply by the internal bypassing of the pulse gate circuit. The advantages achieved are (1) the avoidance of mis-lock to the reference clock input, (2) rapid and guaranteed lock recovery from an aborted read operation, and (3) avoidance of mis-lock within the disk PLL synchronization field (preamble).

Items #1 and #2 above are easily attained by pulse gate bypassing when the PLL is locked to the reference signal in the non-read mode. Pulse gate bypassing allows the digital phase detector to perform unrestricted frequency comparison and thus guarantees lock. Both items are employed within all of National's currently released data separator/synchronizer circuits (see Table 3.1).

Incorporation of item #3 (employed within the DP8461 and DP8451, and optional within the DP8462) is highly dependent on preamble type and places specific requirements on the controller's sector search algorithm. First, there are several common preamble types currently in use on disk drives; (1) the MFM and 1,N type, (2) the 2,7 high frequency preamble, the (3) the 2,7 low frequency preamble:

Code Type	VCO Cycles (Code Positions or Windows) Per Recorded Preamble Bit
GCR*	1
MFM	2
1,7	2
1,8	2
2,7	3
2,7	4

***Note:** GCR (Group Code Recording) is used almost exclusively in tape drive systems; it is mentioned here for comparative purposes only.

Since each preamble is recorded at a different frequency with respect to the VCO operating frequency, the VCO must be internally divided down to equal the preamble frequency for the particular code in use before being fed into the phase detector along with the data pattern. (This function is performed internally within specific National disk PLLs listed in Table 3.1.) It is then the responsibility of the PLL to detect the occurrence of frequency lock and revert back to the pulse gate mode prior to leaving the preamble and encountering random data patterns. Second, while in the frequency acquisition mode, the controller must allow a read operation to begin (initial PLL lock to the data) only during the presence of the appropriate field, i.e., the system must employ a hard-sectored or pseudo hard-sectored PLL control algorithm which will guarantee the PLL read gate will only be asserted at the start of the preamble on the disk, otherwise serious PLL mislock problems will result.

DP846X EXPOSITION

Because of the varied requirements and applications which exist for the data separator/synchronizer, National provides an assortment of disk PLL circuits, including versions which provide frequency lock for specific preamble types, as mentioned above.

TABLE 3.1. Data Separator/Synchronizer Reference List

Device	Synchronized Codes	Separated Codes	Frequency Lock	Delay Trim
DP8461*	MFM; 1,N	MFM	Reference & Data	None
DP8462*	2,7 1,N MFM	None	Reference (Optional for Data)	Optional
DP8465*	All	MFM	Reference	None
DP8451	MFM; 1,N	None	Reference & Data	None
DP8455	All	None	Reference	None

Note 1: ''All'' code synchronization does not include GCR.

Note 2: DP846X devices are in the 24-pin, 300 mil. package; DP845X devices are in the 20-pin, 300 mil. package.

Note 3: *Also available in 28-lead plastic chip carrier.

Note 4: DP8461 and DP8451 pinouts match the DP8465 and DP8455, respectively; for use with hard and pseudo-hard sectoring only.

Note 5: DP8451 and DP8455 are also available in 20-pin plastic chip carrier.

3.3 SYSTEM DYNAMICS—LOOP FILTER DESIGN

The key element contained within the PLL system for governing the loop dynamics and overall performance is the loop filter. It is at this point that the user has the greatest flexibility and control regarding the behavior of the PLL. As previously mentioned, there are several requirements placed on the dynamics of the loop, some of which tend to conflict with others. Table 3.2 lists some of the issues at hand, and where they lie with respect to one another.

TABLE 3.2

	High Bandwidth	Low Bandwidth	High Damping Factor	Low Damping Factor
Lock Time	Good	Poor	Good	Poor
Jitter Rejection	Poor	Good	Poor	Good
Capture Range	Good	Poor	Good	Poor
Noise Immunity	Poor	Good	Poor	Good
Stability	(No Relationship)		Good	Poor

Although the designations of ''good'' and ''poor'' are very general in nature, they apply fairly well here for comparative purposes. An ideal PLL would be able to lock to any frequency and/or phase step in a very short time with no possibility of missed or false lock, settle quickly to a highly stable state, and track any frequency variations encountered in the data stream while at the same time rejecting all bit jitter and extraneous noise. While all this is not possible with any single loop filter, acceptable performance can be achieved via careful compromise, with the design biased to accommodate the more critical parameters.

3.4 OVERVIEW OF FILTER DESIGN OBJECTIVES

The first design objective to be discussed is the minimization of acquisition time. This includes both acquisition of phase lock to the data stream as well as acquisition of phase lock to the crystal frequency (or servo track). Both of these acquisition times impact the length of the preamble field which precedes the address mark. Since longer preambles result in more overhead per data sector, a decrease in formatted disk capacity may result from excessively long acquisition times. Acquisition times are directly controlled by the phase locked loop filter.

The second design objective is the maximization of data margin. Data margin measures the ability of the data separator to allow the data bit to move from its expected time position without a resulting data error. This movement of the data bit away from its expected time position is caused by noise, read channel asymmetry, magnetic domain interference, and other factors in the head, media, and channel portion of the drive system. The data separator generates a window around the expected data position; however, the window accuracy is affected by some factors in the data separator. These factors include delay inaccuracy, VCO jitter, phase detector inaccuracy, and phase locked loop response to bit movement which occurred in preceding windows. The last of these factors, loop response, is controlled by the phase locked loop filter.

The final objective to be checked is the tracking of disk data. The rate of change of phase between the VCO and the read data is modulated by various mechanical phenomena in the drive. Instantaneous variations in disk speed as well as head vibration contribute to this modulation. The maximum frequency of these mechanical resonances tends to be in the 10 kHz or 64 Krads/sec range. Phase-locked loop bandwidth must be wide enough to allow this modulation to be tracked. This objective tends to be encompassed by the acquisition time objective. However, it is conceivable that a system which allows relatively long acquisition times may come up against this barrier.

The loop filter design process may start with any one of these objectives. If the disk format has been established, or a certain disk capacity is desired, the acquisition performance may dictate the loop filter design. If data reliability and error rates are of primary importance, the design may start with margin loss considerations. In any case, all aspects of the loop performance must be checked and the final design is usually a tradeoff between the desired performance and the achievable performance.

3.5 ACQUISITION PERFORMANCE

The read acquisition time is the time between the assertion of READ GATE and the reading of the address mark. Also of concern is the time required for the loop to acquire lock to the crystal frequency. Many application-specific system parameters impact this portion of the loop design. Some of these parameters which will be discussed include sector search algorithms in soft sectored systems and frequency differences between the crystal and the data in removable media systems.

Before the READ GATE is asserted, the VCO is locked to the crystal. When READ GATE is asserted, the phase difference between the VCO and the read data is random. The first portion of the acquisition is where the loop captures phase alignment. In the worst case, the initial phase alignment is such that the data bit is positioned at the edge of the window which gives the proper polarity of error signal, however, the loop cannot keep the bit in the window since it started so close to the edge. One of the results is that the incoming data will appear to be different than what is actually being read. Any system which desires to immediately monitor the read data must wait for this initial cycle slip to occur before reading. The second result is that a series of error signals of the wrong polarity occurs after this initial cycle slip while the phase aligns to the window center. The duration of this slip and phase acquisition is approximately $1/\omega n$ for damping factors between 0.7 and 1.0. See Section 3.7 for acquisition plots. Note that ωn is the loop bandwidth or the natural frequency of the loop and that the phase error is zero at $\omega nt = 1$.

The next period of the acquisition is where the loop begins to capture frequency. There is also some overshoot from the phase acquisition during this period. This analysis assumes that the frequency acquisition begins where $\omega nt = 1$ and is superimposed upon the phase acquisition for $t > 1/\omega n$.

In a fixed media system the difference between the crystal frequency and the read data frequency can be about 1%. In a removable media system the data may be written in one drive and read in another. The total difference between the read data frequency and the crystal frequency can be twice the rotational speed difference of the drive. For example, if the rotational speed variation of the drive is $+1\%$ when the disk is written, and then -1% when the disk is read, the read back data frequency will be 2% slower than the crystal frequency. Since a frequency difference becomes a phase ramp, the phase error will initially grow during read acquisition while the loop attempts to hold the phase error to zero (see Section 3.7). If the peak phase error which results exceeds the window tolerance, the loop will not capture within the desired acquisition time. The data will slip out of its proper window into the neighboring window and a series of error signals of the wrong polarity will result.

When the data rate is 5 Mbit/sec, the window period is 100 ns and the peak phase error allowed is ideally 50 ns. For a damping factor of 0.7, the peak phase error is given by:

$$\text{peak error} = (0.45)\,(\Delta\omega/\omega n)\,(200\ \text{ns}/2\pi) + 0.21(\Delta\theta)$$

where $\Delta\theta$ is the worst case initial phase (50 ns). The $200\ \text{ns}/2\pi$ term converts radians to seconds. $\Delta\omega$ is the

worst case initial frequency difference which is determined by the rotational speed variation of the drive. For a removable media drive with 1% variation:

$$\Delta\omega = (2\pi/200\text{ ns})\ (0.02) = 628\text{ Krads/sec}$$

The results are as follows

$\omega n = 600$ Krads/sec $\rightarrow$ peak phase error $= 25.5$ ns

$\omega n = 500$ Krads/sec $\rightarrow$ peak phase error $= 28.5$ ns

$\omega n = 400$ Krads/sec $\rightarrow$ peak phase error $= 33.0$ ns

$\omega n = 300$ Krads/sec $\rightarrow$ peak phase error $= 40.5$ ns

For $\omega n = 300$ Krads/sec, with a damping factor of 0.7, the loop will just barely capture. If a removable media system were to use a 300 Krads/sec loop bandwidth, the capture range analysis should be performed very carefully since the numbers given here are very approximate. Capture performance improves for larger damping factors, however, total acquisition time increases.

There is no precise definition of phase lock. At the end of the preamble, there will be some residual phase error. The loop is locked if this phase error contributes an acceptable amount of margin loss during the reading of the address mark. It is recommended that the residual error be about 2 ns in a 5 Mbit/sec data rate system:

$$2.0\text{ ns}\ (2\pi/200\text{ ns}) = 0.062\text{ radians}$$

If the damping factor is 0.7, the following results

$\omega n = 600$ Krads/sec $\rightarrow$ error $= 0.062$ at t $= 7\ \mu$s

$\omega n = 500$ Krads/sec $\rightarrow$ error $= 0.062$ at t $= 9\ \mu$s

$\omega n = 400$ Krads/sec $\rightarrow$ error $= 0.062$ at t $= 12\ \mu$s

$\omega n = 300$ Krads/sec $\rightarrow$ error $= 0.062$ at t $= 17\ \mu$s

The total read acquisition time includes the initial phase acquisition as follows:

$\omega n = 600$ Krads/sec $\rightarrow 7\ \mu s + 1.7\ \mu s = 9\ \mu s = 5.5$ bytes

$\omega n = 500$ Krads/sec $\rightarrow 9\ \mu s + 2.0\ \mu s = 11\ \mu s = 6.5$ bytes

$\omega n = 400$ Krads/sec $\rightarrow 12\ \mu s + 2.5\ \mu s = 15\ \mu s = 9.0$ bytes

$\omega n = 300$ Krads/sec $\rightarrow 17\ \mu s + 3.3\ \mu s = 20\ \mu s = 12.5$ bytes

3.5.1. Crystal Acquisition

Analysis of the crystal acquisition time is similar to the read acquisition time. In the case of the National DP8465 data separator, however, a high bandwidth mode is provided to decrease the acquisition time. The high bandwidth is activated when $\overline{\text{SET PLL LOCK}}$ is deasserted. This increases the phase detector gain (increases the charge pump current). When the phase detector gain increases, both the loop bandwidth and the damping factor are increased. Loop performance is poor if the damping factor gets much larger than 1.0 and therefore the increase in loop bandwidth should be limited to the point where the damping factor is 1.0. This means that:

$$\omega n(\text{high track}) = \omega n(\text{low track})\ (1.0/0.7)$$

Repeating the acquisition analysis for the four bandwidths used before:

$\omega n = 600$ Krads/s $(1/0.7) = 857$ Krads/s $\rightarrow$ acq $= 5$ bytes

$\omega n = 500$ Krads/s $(1/0.7) = 714$ Krads/s $\rightarrow$ acq $= 6$ bytes

$\omega n = 400$ Krads/s $(1/0.7) = 571$ Krads/s $\rightarrow$ acq $= 7$ bytes

$\omega n = 300$ Krads/s $(1/0.7) = 429$ Krads/s $\rightarrow$ acq $= 9$ bytes

3.5.2. Margin Loss Due to PLL Response

Fast acquisition is desirable to minimize preamble lengths. However, the wider the loop bandwidth, the larger is the loop response to shifted data. Loop response to shifted data results in margin loss. The data is shifted by noise and other factors which contain no information about data frequency changes. When the loop responds to this bit shift it moves the windows for subsequent data bits thereby reducing the amount of shift allowed for these bits.

For a 5 Mbit/sec system with a maximum shifted early bit, the following formula gives the loop response:

$$\text{loss} = 40[1 - (\cos\sqrt{1-\zeta^2}\,\omega nt - \zeta/\sqrt{1-\zeta^2}\,\sin\sqrt{1-\zeta^2}\,\omega nt)\exp(-\zeta\omega nt)] \quad \text{Note 1.}$$

where 40 is the phase step (in ns) due to the early bit, ωn is the loop bandwidth, and ζ is the damping factor. The phase detector output is active for 40 ns and the amount of loss is determined by setting t $= 240$ ns which is the time to the far edge of the next window where a bit may appear.

The ωn and ζ for this calculation will be the same as used in the data acquisition analysis as long as t does not exceed 2 VCO cycles. If t is much greater than 2 cycles, the effective phase detector gain is reduced and the ωn is reduced also (see design example of ωn formula). This calculation also assumes that the second pole in the filter is well outside the loop bandwidth.

ωn(max freq data) $= 600$ Krads/s $\rightarrow$ loss $= 7.7$ ns

ωn(max freq data) $= 500$ Krads/s $\rightarrow$ loss $= 6.4$ ns

ωn(max freq data) $= 400$ Krads/s $\rightarrow$ loss $= 5.2$ ns

ωn(max freq data) $= 300$ Krads/s $\rightarrow$ loss $= 3.9$ ns

There are techniques for reducing this margin loss without heavily impacting the acquisition performance. See Section 3.7 for details.

DESIGN EXAMPLE FOR 5 Mbit/s DATA RATE

Although there is no real standard, most of the track formats for small Winchesters are using about 12 bytes of preamble. There is a formated gap after the data ECC field but it cannot be assumed that any of this gap is available for PLL acquisition. Some of this gap will be lost when the sector is updated due to rotational speed variation and the remainder is required for write-to-read recovery of the read channel.

Note 1: Phaselock Techniques; Floyd M. Gardner, Second Edition, John Wiley & Sons; pg. 48.

As discussed, in a soft sectored disk drive, acquisition to the crystal, as well as acquisition to the read data, must occur within the preamble. (In a hard sectored drive some of the preamble may be lost to uncertainty in sector pulse detection but usually most of the preamble is available for data acquisition.) For this reason, the loop bandwidth during acquisition must be in the 600 Krads/sec range for a soft sectored drive. The 600 Krads/sec bandwidth gives 5 byte crystal acquisition and 5.5 byte data acquisition. The margin loss can be held to about 6.0 ns + 7.7 ns = 13.7 ns with the DP8465-3. (This margin loss can be reduced if a longer preamble/lower bandwidth were used or by using some of the techniques discussed in the 10 Mbit/sec design example.)

For the DP8465 with $\overline{\text{SET PLL LOCK}}$ asserted:

$$\omega n = \sqrt{(2.5)\,(F_{VCO})/(N)\,(C1)\,(R_{rate})}$$

Rrate should be set at 820Ω since the current in this resistor does have a small effect on the VCO stability and 820Ω has been determined to be the optimum value. F_{VCO} is the center frequency of the VCO in hertz. F_{VCO} is 10 MHz for 5 Mbit/sec data rates. N is the number of VCO cycles per data bit. MFM preamble data has 2 cycles per data bit. This gives:

$$C1 = (2.5)\,(F_{vco})/(N)\,(Rrate)\,(\omega n^2)$$
$$= (2.5)\,(10E6)/(2)\,(820)\,(3.6E11)$$
$$= 0.042E\text{-}6 \text{ (use } 0.039\ \mu F)$$

C1 should be an ultra-stable monolithic ceramic capacitor or equivalent timing quality capacitor.

In the data field, the MFM data frequency can be half the preamble frequency. This means that N = 4 in the bandwidth equation. This reduces the bandwidth by $1/\sqrt{2}$:

$$\omega n \text{ (min)} = (1/\sqrt{2})\,(6.25.1 \text{ Krads/s}) = 442.1 \text{ Krads/s}$$

where 625.2 Krads/sec is the computed bandwidth in the preamble with C1 = 0.039 μF. Since there should be no mechanical resonances anywhere near this frequency, the loop will be able to track the data.

The damping factor should be about 0.5 when ωn is minimum. Response to bit shift is minimized when the damping factor is small; however, if the damping factor drops much below 0.5 the system tends to be oscillatory (underdamped):

$$\zeta = (\omega n)\,(R1)\,(C1)/2 \longrightarrow R1 = (2)\,(\zeta)/(\omega n)\,(C1)$$
$$R1 = 1/(442.1E3)\,(0.039E\text{-}6) = 58 \text{ (use } 56\Omega)$$

The actual damping during acquisition is then:

$$\zeta(acq) = (625.2)\,(56)\,(0.039E\text{-}6)/2 = 0.68$$

The linear approximation used to predict loop performance assumes that the phase detector output is constantly proportional to the input phase difference. In reality, the phase detector output is a pulse applied for a period of time equal to the phase difference. The function of C2 is to smooth the phase detector output over the cycle. C2 adds a second pole to the filter transfer function as discussed in section 3.7. This pole should be far enough outside the loop bandwidth that its phase and amplitude contribution is negligible to the loop bandwidth. If:

$$C2 = C1/50 = 789 \text{ pF (use 820 pF)}$$

the acquisition performance and the margin loss are not significantly changed from the predictions. If a larger C2 is used, the margin loss can be reduced at the expense of the acquisition. This may be desirable for some systems. See section 3.7 for discussion of the function of C2.

Note 1: Ibid.

The final loop component is R_{boost}. When $\overline{\text{SET PLL LOCK}}$ is deasserted, R_{boost} is in parallel with R_{rate} to set the charge pump current. If the parallel combination of R_{rate} and R_{boost} is called R_p:

$$\omega n \text{ (high)} = \sqrt{(2.5)\,(F_{VCO})/(N)\,(C1)\,(R_p)}$$

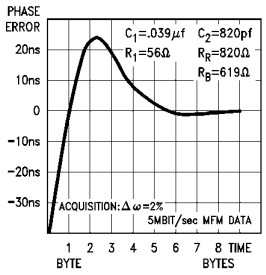
Since the total set current, I_{set}, into the charge pump may not exceed 2 mA, the parallel resistance R_{boost} and R_{rate} (R_p) should also be restricted to:

$$R_p \geq V_{be}/I_{set} = 0.7V/2 \text{ mA} = 350\Omega$$

Solving for R_{boost}:

$$R_{boost} = (R_p)\,(R_{rate})/(R_{rate} - R_p)$$
$$= (350)\,(820)/(820 - 350) = 610.9 \text{ (use } 619\Omega)$$

Remember, R_{boost} is switched-in whenever $\overline{\text{SET PLL LOCK}}$ is deasserted. This design example assumes that $\overline{\text{SET PLL LOCK}}$ is deasserted whenever READ GATE is deasserted.



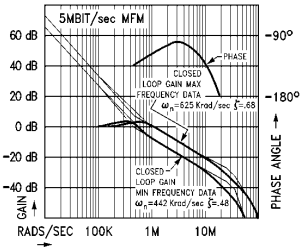
TL/F/8663–59

ωn (acquisition) = 625 Krads/s ζ = 0.68
Lock time ≈ 5 bytes.

ωn (write mode) = 957 Krads/s ζ = 1.04
Lock time ≈ 4 bytes.

ωn (min data) = 442 Krads/s ζ = 0.48

FIGURE 3.14a. 5 Mbit/sec Design Example



TL/F/8663–60

FIGURE 3.14b. 5 Mbit/sec Design Example

DESIGN EXAMPLE FOR 10 MBIT/SEC MFM DATA RATE

At 10 Mbit/sec the window period is reduced to 50 ns and margin loss becomes a much more important design parameter than at 5 Mbit/sec. The total bit shift allowed is half the window (25 ns). If the National DP8465-3 is used, there will be a maximum static loss (independent of the loop filter components) of 6 ns. This part could be used as the starting point for a 10 Mbit/sec design.

The margin loss due to loop response should be kept as low as possible. The maximum allowed bit shift with the DP8465-3 is (25 ns − 6 ns) or about 19 ns. The loss approximation formula is then:

$$\text{loss} = 19[1 - (\cos\sqrt{1-\zeta^2}\,\omega nt - \zeta/\sqrt{1-\zeta^2}\,\sin\sqrt{1-\zeta^2}\,\omega nt)\exp(-\zeta\omega nt)]$$ Note 1.

where 19 is the phase step (in ns) due to an early bit. Evaluation should be done with t = 120 ns:

ωn = 600 Krads/s and ζ = 0.66 ⟶ loss = 1.9 ns

This 1.9 ns loss should be acceptable for most designs. The total loss would then be 6 ns + 1.9 ns = 7.9 ns for a total window of 34.2 ns with the DP8465-3. As discussed in Section 3.7, the actual margin loss due to loop response will be less than 1.9 ns due to the roll off of the second pole in the loop filter.

The acquisition performance of a 600 Krad/s loop was analyzed in the 5 Mbit/sec design example. With a 2% frequency difference between the read data and the crystal, the peak phase error was about 24 ns. At 10 Mbit/sec $\Delta\omega$ is doubled and the conversion term becomes 100 ns/2π. The initial phase misalignment is 25 ns. The peak phase error is then just over 20 ns. This is pushing the capture range of the DP8465-3 since its allowed shift is only 19 ns. If the system is fixed media with 1% or better rotational speed variation there is no problem since the peak phase error is reduced by a factor of two in this case. If capture is needed at 2%, $\overline{\text{SET PLL LOCK}}$ can be deasserted during read acquisition as discussed later.

Notice that the 5 Mbit/sec lock time was given in bytes (i.e. 5.5 bytes). At 10 Mbit/sec the read acquisition time will remain about the same but that means that the number of bytes is doubled to 11 bytes. This read acquisition time can be reduced without increasing the margin loss if the disk controller will hold $\overline{\text{SET PLL LOCK}}$ deasserted (high) during read acquisition. This increases the damping factor to 1.04 and the bandwidth to 940 Krads/sec. The result is a read acquisition time of 8 bytes. The crystal acquisition time is also 8 bytes so the total preamble length is 16 bytes in a soft sectored disk drive. (This assumes zero monitoring overhead. If x bytes are required to determine that the read data is not preamble, the total preamble length would be 16 + x bytes.) Of course the price paid for this reduction in acquisition time is that the controller must now control the $\overline{\text{SET PLL LOCK}}$ input during acquisition.

The calculation of the loop components is similar to the 5 Mbit/sec data rate example:

$C1 = (2.5)\,(F_{VCO})/(N)\,(R_{rate})\,(\omega n^2)$

$= (2.5)\,(20E6)/(2)\,(820)\,(3.6E11)$

$= 0.085E\text{-}6$ (use 0.082 μF (5%))

C1 should be an ultra-stable monolithic ceramic capacitor or equivalent timing quality capacitor. Evaluation of the bandwidth formula with C1 = 0.082 μF gives:

$\omega n = \sqrt{(2.5)\,(20E6)/(2)\,(820)\,(0.082E\text{-}6)} = 609.8$ Krads/s

The target damping factor at this bandwidth is 0.66:

$R1 = (2)\,(\zeta)/(\omega n)\,(C1)$

$= (2)\,(0.66)/(609.8E3)\,(0.082E\text{-}6) = 26.4$ (use 27Ω)

The total set current into the charge pump through the parallel combination of R_{boost} and R_{rate}, (R_p), must not exceed 2 mA. For $I_{set} \leq Vbe/R_p$, then:

$$R_p > Vbe/I_{set} = 0.7V/\,2\text{ mA} = 350\Omega$$

Solving for R_{boost}:

$R_{boost} = (RP)\,(R_{rate})/(R_{rate} - R_p)$

$= (350)\,(820)/(820 - 350) = 610.6$ (use 619Ω)

This design example assumes that R_{boost} is switched in during crystal acquisition and data acquisition.

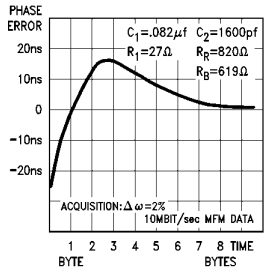
If the margin loss due to loop response of 1.9 ns is acceptable and acquisition performance is to remain unaffected:

$C2 \ll C1/50 = 0.082\ \mu F/50 = 1640$ pF (use 1600 pF)

A smaller value of C2 is chosen for this example since the 1.9 ns of margin loss seems very acceptable whereas it is undesirable to impact the capture range or the acquisition time. Larger values of C2 may be used, however, the additional pole in the filter may begin to affect frequencies inside the loop bandwidth. The major impact of large C2 is on the capture range although some degradation in read acquisition time can be seen for larger values of C2. See section 3.7 for further discussion.

3.6 COMMENTS ON OTHER CODES

MFM is a 1,3 RLL code. This means that a minimum of one empty window will occur between two windows which each contain a disk data bit, and that a maximum of three empty windows will occur between windows which each contain a disk data bit. The most popular of the newer RLL codes are the 2,7 codes, in which there are a minimum of 2 and a maximum of 7 windows between windows containing a disk data bit. Although the ratio of encoded disk data bit positions (windows) on the disk to non-encoded data (NRZ) bits for both MFM and 2,7 code is 2:1, the two codes differ in the actual number of recorded pulses required to store a given number of NRZ bits (their NRZ-bit versus disk-bit ratio, or efficiency). The 2,7 code requires fewer recorded bits than MFM on average for disk encoding of the same amount of information and has a 50% larger minimum bit spacing than MFM. These allow, on a given disk, a theoretical increase in data storage of 50% when 2,7 encoding is chosen over MFM while the minimum flux transition spacing is kept constant.



TL/F/8663–61

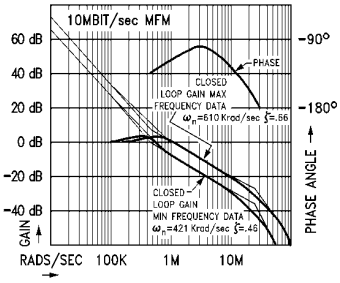
Controller must keep $\overline{\text{SET PLL LOCK}}$ deasserted for 8 bytes after READ GATE is asserted.

ωn (acquisition) = 933 Krads/s ζ = 1.03

ωn (min data) = 431 Krads/s ζ = 0.48

ωn (max data) = 610 Krads/s ζ = 0.66

FIGURE 3.15a. 10 Mbit/sec Design Example

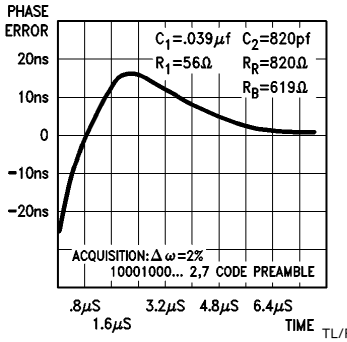


TL/F/8663–62

FIGURE 3.15b. 10 Mbit/sec Design Example

The impact of a 2,7 code on the data separator is significant. Loop bandwidth is dependent upon the sample rate into the phase detector. With MFM it has been seen that the bandwidth in minimum frequency data is $1/\sqrt{2}$ times the bandwidth in maximum frequency data. This is a ratio of 1:0.707. In a 2,7 code the ratio is $\sqrt{3/8}$ or 1:0.612. The damping factor follows the bandwidth so a 2,7 code system must be more carefully designed to avoid underdamped or overdamped response.

Another complexity of 2,7 codes is that most systems are not using the maximum frequency data in the preamble. The 100100 . . . . encoded data does not decode to a data pattern with byte alignment. The 10001000 . . . encoded data pattern decodes to all ones byte aligned data and is being used more often. The lower frequency data increases the read acquisition time and increases the probability of harmonic lock. Channel induced pulse pairing (from channel asymmetry), coupled with an initial phase alignment which puts the data bit at the extreme window edge, may allow the loop to stabilize out of phase. There are two techniques which are used to eliminate this problem. The first is to start the VCO in phase with the read data (zero phase start-up). The second technique is to perform phase and frequency comparison (i.e. do not window the data) during read acquisition. The second technique is used on the National DP8462 data separator which is specifically designed for 2,7 codes.



TL/F/8663–63

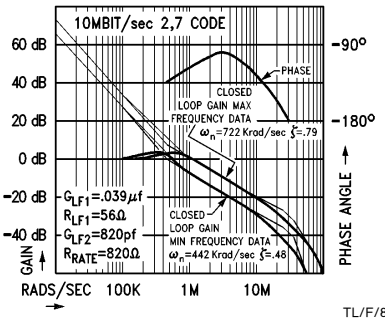
Controller must keep $\overline{\text{SET PLL LOCK}}$ deasserted for 6 μs after READ GATE is asserted.

ωn (acquisition) = 956 Krads/s ζ = 1.04

ωn (min data) = 442 Krads/s ζ = 0.48

ωn (max data) = 722 Krads/s ζ = 0.79

FIGURE 3.16a. 10 Mbit/sec 2,7 Code Example



TL/F/8663–64

FIGURE 3.16b. 10 Mbit/sec 2,7 Code Example

Note 1. Ibid.

In conclusion, loop filter design for 2,7 codes is more difficult than for MFM. In hard sectored drives, or pseudo hard sectored drives with dc erased gaps, the problem is simply the wide range of damping factors and can be easily solved.

3.7 LOOP FILTER DETAILS

Time domain response for a second order feedback control system is well known. The response to a phase step and a frequency step is shown in the graphs in *Figure. 3.17.* The phase locked loop system is normally designed to have a damping factor between 0.7 and 1.0 during acquisition so these curves show the performance boundaries. The equations for the phase step response are:

$$\zeta < 1 \rightarrow \theta(t) = \Delta\theta(\cos\sqrt{1-\zeta^2}\,\omega nt - \zeta/\sqrt{1-\zeta^2}\,\sin\sqrt{1-\zeta^2}\,\omega nt)\exp(-\zeta\,\omega nt)$$

$$\zeta = 1 \rightarrow \theta(t) = \Delta\theta(1-\omega nt)\exp(-\omega nt) \qquad \text{Note 1}$$

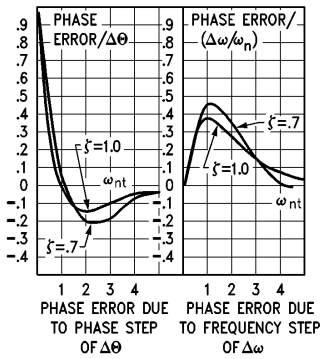
The equations for the frequency step response are:

$$\zeta < 1 \rightarrow \theta(t) = (\Delta\omega/\omega n)\,(1/\sqrt{1-\zeta^2}\,\sin\sqrt{1-\zeta^2}\,\omega nt)\,\exp(-\zeta\,\omega nt)$$

$$\zeta = 1 \rightarrow \theta(t) = (\Delta\omega/\omega n)\,(\omega nt)\,\exp(-\omega nt) \qquad \text{Note 1.}$$

These equations were used to derive the margin loss due to bit jitter (i.e. phase steps) as well as the acquisition performance in the previous design examples. The second order time domain response is the usual starting point for loop filter design.

Disk data separation is complicated by the fact that the input data stream is not a single frequency. Missing data bits must not be allowed to generate error signals to the loop since the loop bandwidth would have to be set unacceptably low to filter out this erroneous information. As a result, most phase detectors for disk data separators do not generate a continuous voltage dependent upon the input phase difference. The discontinuity in the input data stream is dealt with by only generating a phase detector output when a data bit has arrived and only during the period of time corresponding to the input phase difference.

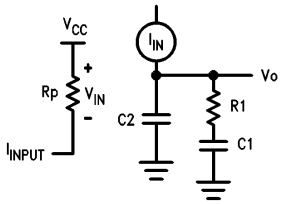


TL/F/8663–65

FIGURE 3.17. Phase Error as a Function of Damping and Phase Step

There are some problems which result from a pulsed phase detector output. First, oscillator response can be affected by the phase relationship between these pulses and its internal charging and discharging cycle. Second, the response of the oscillator occurs in a quantum jump when the error pulse is generated, reducing margin for later bits. These problems are reduced by adding one or more poles to the loop filter. These poles smooth the phase detector output and reduce the loop response to bit jitter. As long as the added poles do not add significant gain loss and phase shift within the loop bandwidth, the system time domain response will not differ appreciably from a pure second order feedback control system.

The voltage output of the National DP8465 phase detector is converted to a current which is sourced into the filter:



TL/F/8663–66

FIGURE 3.18 Charge Pump and Loop Filter Equivalent Circuitry.

If Zf is the input impedance of the loop filter:

$$Zf = (1/sC2) \,//\, (1/sC1 + R1) =$$

$$(1 + sC1R1)/sC1(sC2R1 + C2/C1 + 1)$$

$$Vo = (Iin)\,(Zf) =$$

$$(Vin/R_p)\,(1 + sC1R1)/sC1(sC2R1 + C2/C1 + 1)$$

$$Vo/Vin = (sC1R1 + 1)/sC1R_p(sC2R1 + C2/C1 + 1)$$

The effect of C2 is to introduce a pole into the transfer function of the loop filter. The pole location is where $s = (1 + C2/C1)/C2R1 = 1/C2R1$ if $C2 << C1$.

The open loop gain is the product of the phase detector, the filter, and the oscillator transfer functions. The phase detector transfer function is simply a constant, K_{pd}, in volts/radian. The oscillator transfer function is K_o/s where the 1/s term represents phase as the integral of frequency and K_o is in units of radians/(volt x sec). The open loop transfer function is then:

$$G(s) = (K_oK_{pd})\,[st2 + 1]/[st1(st3 + 1)s]$$

where $t1 = C1R_p$, $t2 = C1R1$, and $t3 = C2R1$. The transfer function has a zero at 1/t2, a pole at 1/t3, and two poles at the origin.

Note: For the National DP8465 the K_oK_{pd} product is (2.5) $(F_{VCO})/N$ where F_{vco} is the oscillator frequency in Hertz and N is the number of oscillator cycles between data bits.

In the frequency domain, the open loop gain falls at 40 dB/decade until equal to $K_oK_{pd}/t1$ at $j\omega = 1$. The 40 dB/decade slope continues until the zero at $j\omega = 1/t2$. At the zero the slope changes to 20 dB/decade. The slope returns to 40 dB/decade when the pole breaks at $j\omega = 1/t3$. The phase shift begins at -180 degrees and asymptotically approaches -90 degrees. The phase is equal to -135 degrees at $j\omega = 1/t2$. The phase plot turns around and starts back toward -180 degrees as $j\omega$ approaches 1/t3 such that at $j\omega = 1/t3$ the phase equals -135 degrees.

If a second pole were in the filter around $j\omega = 1/t3$, the phase would equal -225 degrees at $j\omega = 1/t3$ and stability would require that the gain be below 0 dB before the phase reached -180 degrees. This sometimes limits how closely the poles can be moved to the loop bandwidth. When the gain is 0 dB, the difference between the actual phase shift and 180 degrees is referred to as the phase margin. The open loop phase margin is related to the damping factor of the second order system. Note that there is a pole in the buffer amplifier of the DP8465 between the filter and the VCO. This pole is at 5 MHz or about 31.4 Mrads/s and could affect the phase margin in a very wide band loop.

The additional pole in the loop filter helps to improve read margin because it lowers the loop gain at the frequency of the bit jitter. The fundamental frequency content of the bit jitter is slightly below the bit frequency since the pump up error begins before the end of the window and the pump down error ends after the end of the window. At 5 Mbit/sec, the bit jitter frequency is 1/(200 ns + 80 ns) = 22.4 Mrads/sec for 40 ns bit shifts. The 615 Krads/sec loop designed earlier for 5 Mbit/sec data would have a gain of -28 dB at $j\omega = 22.4$ Mrads/s if C2 were not in the loop. With C2 = 820 pF, the gain is reduced to about -31 dB at the bit jitter frequency. If the additional pole were added at $j\omega = 6.15$ Mrads/s (10 ωn) the gain would be -38 dB at the bit jitter frequency for an extra 10 dB of noise rejection. C2 would be 0.0027 μF in this case.

It is difficult to analytically predict the effect of C2 on the acquisition performance. Since C2 is moving close to the loop bandwidth, the system behavior is not purely second order. There are some computer programs which allow time domain response to be predicted for third order systems but normally it is not necessary to use these tools. It is usually sufficient to start with a second order analysis and experimentally measure the system performance as the third pole (or poles) is brought closer to the loop bandwidth.

CHAPTER 4 DP8466 Disk Data Controller Overview

4.0 INTRODUCTION

National's Disk Data Controller (DDC) chip, DP8466, performs many of the functions in the disk data electronics path of either disk controllers or intelligent disk drives. The primary function of the chip is to correctly identify the selected sector on disk and then to transfer that sector's data to or from memory.

The DDC performs serialization and deserialization of disk data, CRC/ECC generation, checking and correction, data buffering with a 16-word (32-byte) FIFO, and single or dual channel DMA addressing. It can write NRZ or MFM encoded data to the disk. The data separation required in the disk data path electronics can be obtained by using one of National's Data Separator chips, DP8460 or DP8461/5. If 2,7 is used instead of MFM, 2,7 ENDEC chip could be used in conjunction with the DP8462 2,7 Data Synchronizer. The DDC is fabricated using the dual layer metal 2μ microCMOS process, which allows complex functions to be implemented with high operating speeds and modest power consumption. Internal gate delays of less than 2 ns allow the DDC to function with disk data rates up to 25 Megabits/sec. This enables the DDC to be used not only with 3½-inch, 5¼-inch and 8-inch drives, either Winchester or Floppy (or both), but also with high-end drives such as 14-inch Winchester drives, vertically recorded drives, and optical drives.

The DDC interfaces with drives compatible with the ST506, ST412HP, ESDI, SMD and other interfaces. Also the DDC may instead be part of an intelligent disk drive that has a SCSI (SASI), or an IPI type interface. Refer to chapter one where the block level boundaries of the various disk interface standards are shown.

4.1 THE DDC ARCHITECTURE AND BASIC OPERATION

An architectural block diagram of the DDC is shown in *Figure 4.1*. The 64 internal registers consist of control, command, pattern and count registers. These registers are initially preloaded with information such as header or synch bytes, ECC polynomial bytes, preamble or postamble patterns, or address marks (for soft sectored drives) etc. Some of the registers will be programmed each time the DDC starts an operation, for example the command register.

The DP8466 has a range of commands that enable reading and writing of both data, and header fields, checking for header fields, formatting with either hard or soft sectored formats, and aborting. Each of these operations can be performed in various modes and an abundance of formats. Most operations can be performed as single or multi-sector operations.

In a typical disk read or write operation, the desired sector (where the data information is to be read from or written to) is first located by comparison of the header bytes. To achieve this comparison, the incoming serial data from the external data separator is deserialized into byte-wide data that is fed both to a comparator and FIFO. The comparator checks the address mark (if present) and the synch bytes to align the incoming bytes. Once the incoming data stream has been byte aligned, header comparison for the desired sector then begins. As each header byte is deserialized it is compared with the next preloaded header byte. If any of the header bytes do not match, the desired sector has not been located, but the DP8466 still performs a CRC/ECC check on the header and waits for the ID segment of the next

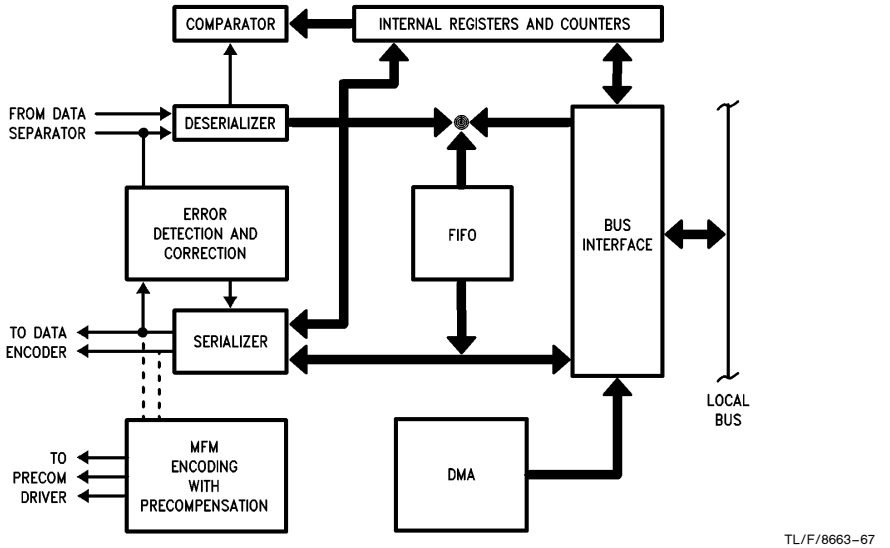


FIGURE 4.1. The DDC Data Path Architecture

sector. If after two disk revolutions the header is not found the DDC will abort the operation. Once a header match is detected, the DDC prepares to transfer data to or from the data segment of the sector.

1. When writing a data segment to the disk, the DDC first inserts the preamble pattern field, address mark (for soft sectored drives) and synch fields, each byte repeated a specified number of times. These fields are followed by the data field bytes that are provided sequentially through the FIFO and external memory. Internal CRC or ECC (or external ECC) check bits are generated from the bits in the data field and subsequently appended to it. The write operation ends with the postamble. As each byte is serially transmitted, the next byte becomes available to be serialized and output. The serial output may be either NRZ data with the associated write clock or MFM encoded data.

2. When reading a data segment from the disk, the DDC deserializes the incoming data and byte aligns with the address mark (if present) and synch fields using the comparator. It then transfers the data field bytes into FIFO. At the same time it checks incoming serial data using an internal CRC code or ECC code.

For both read or write operations, disk data goes to or from the internal FIFO. Once the FIFO has filled or emptied to the selected threshold level, data may be transferred to or from the external buffer memory in the selected burst length by means of a DMA channel. A second DMA channel is available to transfer data to or from buffer memory to the system (for systems that utilize a buffer memory).

If the operation terminates properly an interrupt is issued, and the user may check status. If an error results during the operation the DDC will also interrupt the microprocessor, and the user must determine the appropriate action.

The DDC can be configured in three different modes; peripheral, master and slave. In the peripheral mode, the microprocessor accesses internal register to read or write data. The DDC acts like a peripheral when it is being configured, and when the microprocessor issues a command. During the execution of a command and when the on-chip DMA has been granted access to the bus for local and remote transfers, the DDC goes into its master mode, and becomes bus master. If during the command an external DMA controls data transfer the DDC will go into a slave mode.

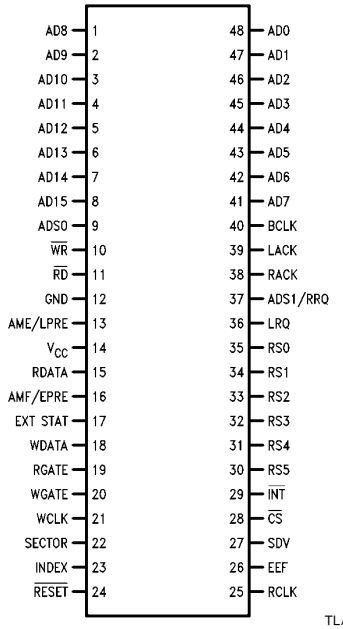
4.2 PIN ASSIGNMENT AND DESCRIPTION

In this section a complete pin description is presented. The pin assignment diagrams are shown in *Figure 4.2(a)* and *4.2(b)*. Specific timing information for these signals can be found in the DP8466's Datasheet.

4.2.1 Bus Interface

Chip Select ($\overline{CS}$): When the DDC is in the peripheral mode the chip select signal must be asserted low to access enable microprocessor access. In the peripheral mode pins RS0–5 are address inputs and pins AD0–7 are set for 8-bit transfer of data between the DDC and microprocessor. $\overline{CS}$ has no effect if on-chip or external DMA is performing a transfer. (DDC in slave or master modes.)

Bus Clock (BCLK): The DDC uses BUS CLOCK input as the reference clock when the DDC is bus master. It is used only during RESET and DMA operations and is independent of the disk data rate. BCLK may be the microprocessor clock and must be at least ¼ the rate of READ CLOCK, RCLK.



TL/F/8663–68

FIGURE 4.2(a). The DDC (DP8466) Connection Diagram

Address/Data (AD0–7): This 8-bit data/address bus port has one microprocessor associated function and four memory transfer associated functions. When the DDC is in the peripheral mode and $\overline{CS}$ is set low, this port transfers data between the internal sections of the DDC and the microprocessor. When external DMA is active (i.e. the DDC is in slave mode) with LACK (local acknowledge) set low, data bits D0–7 are transferred between the FIFO and memory.

When the DDC is controlling the bus (i.e. when on-chip DMA is active), the AD0–7 bus is multiplexed between DMA address and FIFO data bits. Using the single DMA mode, A0–7 are issued on this port as are A16–23. When using dual DMA, these lines are used to transfer both the local and remote DMA address bits, A0–7. In either dual or single channel mode, the data bits D0–7 are transferred between FIFO and external memory through this port.

Address/Data (AD8–15): This 8-bit I/O port has four memory transfer functions (in the peripheral mode with $\overline{CS}$ low, these pins remain indeterminant low impedance). In the slave (external DMA active) mode with LACK set low, data bits D8–15 are transferred between the FIFO and memory when 16-bit transfers are enabled. When the DDC is controlling the bus, (master mode) it issues address bits A8–15 on this port and can also issue address A24–31 if it is in single channel DMA mode.

Register Select (RS0–5): In the peripheral mode, these 6 inputs are used to select the internal registers to be accessed by the microprocessor. These inputs feed ''fall through'' latches that are controlled by the ADS0 input. The RS0–5 inputs fall through and are decoded by the DDC when the input level on ADS0 pin is high. The RS0–5 inputs are stored on the falling edge of ADS0. This enables easy connection to either multiplexed or non-multiplexed buses.

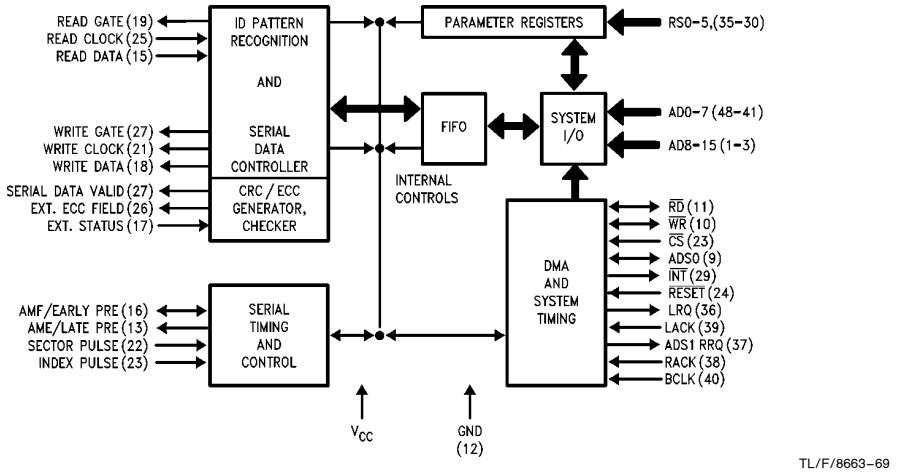


FIGURE 4.2(b). The DDC Block Diagram with Pin Assignment

Address/Data Strobe 0 (ADS0): This I/O strobe has two functions. In the peripheral mode, ADS0 becomes an input and may be used as a microprocessor address strobe input. In this mode when ADS0 is high, the address bits on RS0–5 enter the Register Select Latch and are latched on low going transition.

With the bus under DDC control, ADS0 becomes an output that issues the address strobe to external memory at the start of the DMA transfer cycle. The low going transition of ADS0 coincides with the DMA address bits A0–15 being valid on pins AD0-15. (Note: ADS0 when an output will still latch ''data'' into the RS0–5 latches. Normally this is random data and is of no consequence since $\overline{CS}$ is high. However when the system wants to access the DDC after a DMA, the proper address must be loaded into the latches or ADS0 must be high prior to $\overline{CS}$ going low.)

Address/Data Strobe 1 / Remote Request (ADS1/RRQ): This output pin can be configured to have one of two functions. If the DDC has been configured for 32-bit address single channel DMA, the pin becomes ADS1. This address strobe is issued either at the start of the very first memory transfer cycle of a new disk operation or when the lower 16 address bits have just rolled over. In either case the address on pins AD0–15 is A15–31 at the instance of low going transition of ADS1.

If the DDC is configured to perform remote DMA transfers in its dual channel mode, the pin becomes RRQ, or Remote DMA Request. The DDC will assert the RRQ high whenever it is ready to transfer data between buffer memory and the system I/O port. The RRQ will be reset at the end of the selected burst transfer length.

Remote DMA Acknowledge (RACK): This input pin must be asserted high after RRQ has been set high, when the external I/O device is ready to transfer data. Data will be transferred between external buffer memory and the remote I/O device until the DDC sets RRQ low, or until RACK is set low externally. If RACK is removed during a transfer any cycle in progress at this time will complete.

Local DMA Request (LRQ): This output pin is low when no data transfer is required between FIFO and the external buffer memory. The DDC asserts LRQ high when FIFO requires data to be transferred to or from the external buffer memory for both dual and single DMA mode. LRQ will return low during the last cycle of the burst transfer or on emptying or filling up the FIFO.

Local DMA Acknowledge (LACK): Once the DDC has set LRQ high, and the external circuitry subsequently sets the LACK input high the DDC becomes bus master. Data transfers between the on-chip FIFO and external memory will now proceed until the DDC sets LRQ low, or until LACK is set low externally, in which case any cycle in progress at this time will complete.

$\overline{\text{Read}}$ ($\overline{\text{RD}}$): The $\overline{RD}$ strobe I/O pin has two functions. In the peripheral mode $\overline{RD}$ is an input that when low causes data from a DDC register (selected through pins RS0–5) to be output on pins AD0–7 when CS is low. Pins AD0–7 will be high impedance before and after the READ strobe.

When the DDC has bus control, with either LACK or RACK active, the $\overline{RD}$ strobe is an active low output from the DDC to be used by external memory. It enables data to be transferred from the selected memory location to either the on-chip FIFO for local transfers or to an external I/O device for remote transfers.

$\overline{\text{Write}}$ ($\overline{\text{WR}}$): The $\overline{WR}$ strobe I/O pin has two functions. In the peripheral mode WR is an input that when set low will cause data present on pins AD0–7 to be loaded into a DDC register (selected through pins RS0–5, and when $\overline{CS}$ is low).

When the DDC has bus control, with either LACK or RACK active, the $\overline{WR}$ strobe is an active low output used to write data to the selected memory location from either the on-chip FIFO for local transfers or from an external device for remote transfers.

Interrupt (INT): The INT output is set active low whenever the DDC wishes the controlling microprocessor to check status. It is set high when the microprocessor services the interrupt by setting the $\overline{CS}$ input low and reading the Status Register.

Reset: The RESET pin is an input that is normally set high. When RESET is set low, the DDC goes into the internal reset mode. It clears the FIFO contents, the Status and Error registers and also deactivates LRQ, RRQ, WRITE GATE and READ GATE.

4.2.2 Disk Interface

Write Gate (WGATE): When the DDC writes data to the disk during either a Write Data, Write Header or Format operation, it asserts the WRITE GATE output pin active high at the start of the operation. The transition coincides with the first WRITE DATA bit being issued. WRITE GATE remains high until either the last data bit of the sequence to be written has ended, or when the DDC aborts or resets.

Write Data (WDATA): During a write disk operation, this pin outputs serial disk data. The DDC can be configured to output either NRZ or MFM encoded data on the WRITE DATA pin. If NRZ data is selected its bit rate has the same period as the WRITE CLOCK output. When the DDC has been configured to issue MFM encoded data to the disk, data pulses will be output on WRITE DATA as determined by the MFM encoding rules. In either configuration, when WRITE GATE is inactive low, so is WRITE DATA.

Write Clock (WCLK): When the DDC is configured to write NRZ data, a synchronous clock is provided at the WRITE CLOCK output. The WRITE CLOCK frequency is the same as READ CLOCK. When the DDC is configured to output MFM encoded data, clock information is not needed however this output will still toggle.

Address Mark Found/Early Precompensation (AMF/EPRE): This pin has two modes. When the DDC is configured to read from a soft sectored disk, this pin is ADDRESS MARK FOUND, an active high input. Normally this input will be low but whenever external circuitry detects an Address Mark (such as a missing clock, or blank information) AMF should go high for at least one period of READ CLOCK. This will indicate to the DP8466 that a valid address mark has been located.

When the DDC is configured to write MFM encoded data, this pin becomes the output EARLY PRECOMPENSATION. When it is high, the MFM pulse appearing on the WRITE DATA output requires early precompensation. When low, the MFM pulse does not require early precompensation.

If both functions are being used in the system, WRITE GATE is used to determine the function of this pin. When WRITE GATE is active high, this pin is an LPRE output, otherwise it is an AMF input. External demultiplexing circuitry can be used.

Address Mark Enable/Late Precompensation (AMF/LPRE): This pin has two modes of operation depending on whether NRZ or MFM data is written to the disk. When the DDC is configured to write NRZ data on a soft sectored disk, this output pin is ADDRESS MARK ENABLE. AME is normally low and will remain low when the DDC is configured for a hard sectored disk (bit HSS in disk format register is set). On the other hand, for the soft sector configuration, the DDC will set AME active high during the time any Address Mark byte is serially output on WRITE DATA pin.

When the DDC is configured to write MFM encoded data, this pin becomes LATE PRECOMPENSATION output. In this configuraton if LATE PRECOMPENSATION is high, then late precompensation is required on the MFM pulse being output on WRITE DATA. If LATE PRECOMPENSATION is low, late precompensation is not required. If both EARLY PRECOMPENSATION and LATE PRECOMPENSATION output pins are set low, no precompensation is required.

Read Gate (RGATE): When the DDC is set to read the disk, such as in a Read Header, Compare Header, Ignore Header, Read Data or Ignore Data operation, READ GATE will go active high. This informs external data separator that it can begin locking on to incoming disk data. If the data separator fails to achieve locking, the DDC will set READ GATE inactive for 18 bit times before another locking attempt is made. READ GATE is also set inactive either at the end of the specified operation or if the DDC aborts or resets.

Read Data (RDATA): Once READ GATE has been set active high and external circuitry has locked on to the incoming encoded disk data, the encoded data must be separated into clock and NRZ data. The NRZ data connects to the READ DATA input of the DDC, and is clocked into the DDC on the positive edge of READ CLOCK. When READ GATE is set low, the READ DATA input will be ignored.

Read Clock (RCLK): READ CLOCK is a clock input that may have slightly differing frequencies, depending on the READ GATE control pin. When READ GATE is inactive, this clock should be derived from either a servo clock or a crystal clock to produce a clock with a period close to the bit rate of the disk data. After READ GATE has been set active, and external circuitry has locked on to the incoming encoded disk data, the READ CLOCK input must switch frequency (without any short pulses or glitches) to a period identical to the READ DATA signal.

Sector Pulse (SPULSE): In hard sectored drives the SECTOR PULSE input goes high as the start of each sector passes under the drive head. Once the DDC detects this high signal (for at least one period of the READ CLOCK input), it interprets this to indicate a sector operation can begin. In soft sectored drive there is no sector pulse and the start of each sector must be indicated by an Address Mark byte or bytes, this pin should be tied to ground.

Index Pulse (IPULSE): All drives have an index pulse output that goes active high as the beginning of any track passes under the drive head. Once the DDC detects this high signal (for at least one period of the READ CLOCK input), it assumes an INDEX PULSE has occurred. The DDC uses the INDEX PULSE input to begin various operations.

Serial Data Valid (SDV): This output pin goes high whenever the DDC is issuing or receiving either header field bytes and internal header CRC or ECC bytes, or data field bytes and internal data CRC or ECC bytes. It is set high synchronous with the first header or data bit appearing on the WRITE DATA output pin, or the READ DATA input pin. If the encapsulation mode is set then SCV is set high synchronously with the first sync byte (address mark). If the start with address mark bit is set and encapsulation is enabled SDV will be set high at the first sync #2 byte. (See Chapter 5 and 6.2.) It is set low synchronous with the last bit of internal CRC or ECC field ending on the WRITE DATA output pin or the READ DATA input pin. If internal CRC or ECC is not selected, it will be set low synchronous with the last bit of the header field or data field. The SERIAL DATA VALID pin may be used to select external ECC circuitry or for diagnostics in checking the lengths of the fields.

External ECC Field (EEF): This output pin is normally low, but will go high at specified times if external ECC has been selected. This will be during the time the external ECC field check bits need to be generated (with WRITE GATE high) or checked (with READ GATE high). It will be deasserted (synchronous with SERIAL DATA VALID output going low) after the last bit of the external ECC field has ended.

External Status (EXT STAT): The EXTERNAL STATUS input pin has three possible functions: Enabling wait states for the DMA, or Supplying external synchronization information and/or ECC information to the DDC. The user selects either the first alternative, or the other two. In other words, generating wait states is mutually exclusive with external synch and ECC. If the wait state alternative is selected, the use of this status pin is limited to only supplying wait states to the DMA bus cycle. If the latter two alternatives are selected, input signals on EXTERNAL STATUS may provide synchronization at the start of a header or data field, and external ECC error status at the end of the external ECC field.

4.2.3 Power Supply

V_{CC}, GND: The supply pins require a standard $+5V \pm 10\%$ regulated supply. As with any high speed controller that must connect to high speed buses, output switching transients can cause supply noise glitches which can affect other circuitry within the IC. Thus, a good ceramic decoupling capacitor is recommended to be connected across these pins. This capacitor should be $\geq 0.1\ \mu f$ and should be located in close proximity to the V_{CC} and ground pins. Good GND and VCC planes are also recommended. Both of these precautions are to minimize the effects of current switchings on the chip affecting sensitive sections of the chip. With inadequate decoupling or GND and VCC planes, inexplicable behavior of the chip may result.

4.3 DDC FUNCTIONAL DESCRIPTION

This section is intended to provide a block level functional overview of the DDC. The detailed operational information is given in Chapter 7. A block diagram of the DDC is shown in *Figure 4.1*. The DDC is composed of a bus interface unit which communicates with the microprocessor and memory. It also is composed of a serializer and a deserializer that communicates with the disk. A single/dual channel DMA block provides intelligent on-chip data transfer. This DMA controller transfers data to and from the internal multi-mode FIFO block. A 32/48 bit ECC or 16-bit CRC correction block is included for error generation and checking of disk data. The functional description of each block follows.

4.3.1 Bus Interface

This block of the DDC provides an interface between the DDC and system bus through its two input/output data bus ports (AD0–7, AD8–15) and one input port (RS0–5). In the peripheral mode, the internal registers of the DDC are selected through pins RS0–5 and data is transferred between microprocessor and the internal registers through the I/O port AD0–7.

When the DDC is controlling the bus, two I/O ports (AD0–7, AD8–15) provide 16-bit address both for the local and remote DMA data transfers. In single channel DMA mode, an address up to 32 bits could be obtained to access memory up to 4 Gigabytes (see the DMA block description section). Multiplexed along with the DMA address information on these two ports is data information. In the 8-bit transfer mode only AD0–7 is used, and the interface logic contains a multiplexer to convert 16 internal data bits to 8 bits. For 16-bit transfers, AD0–7 transfers the lower 8 bits and AD8–15 transfers the upper 8 bits between the FIFO and the system bus.

4.3.2 Internal Registers

The DDC has 64 internal registers including parameter, pattern and count registers. Some of these registers are read-only, some write-only and the remainder read/write. These registers can be classified in four categories:

1) Command and Control Registers
2) ECC/CRC Registers
3) Format Registers
4) DMA Registers

Each of the above mentioned classes is described in the following paragraphs. A list of the DDC's internal registers with their hexadecimal addresses is given in Table 4.1.

COMMAND AND CONTROL REGISTERS

The Command and Control registers are the key registers of the DDC. They control basic functions and operations of the chip. The registers which can be included in this category are Drive Command (address 10H), Operation Command (address 11H), Status (address 00H), Error (address 01H), Disk Format (address 35H), Sector Counter (address 12H), Number of Sector Operations (address 13H), Header Byte Count/Interlock (address 0FH), and Header Diagnostic Readback (address 36H). Table 4.2 lists these registers along with a short description.

The Drive Command register basically determines the operations to be performed on the disk data. Also it can be used to set the DDC to format drives and to abort any operation in progress. The operations determined by the drive command register can then be controlled through the Operation Command register. The Operation Command register enables the DDC to issue certain interrupt and acknowledge signals during different operations. The Status register gives the status of the operation while the Error register indicates errors which may occur during these operations.

The DDC is adapted to the selected drive format through the Format register which determines the format of the information to be written to the disk. The Start Sector, Number of Operations, and Header Byte count registers, in conjunction with the Drive Command register, allow the DDC to perform multisector operations. The Header Diagnostic Readback register on the other hand enables the DDC to perform a readback operation on the header bytes present in the FIFO.

TABLE 4.1. The DDC Internal Registers In Numerical Order

Hex Address	Name
00	Status Register
01	Error Register
02	ECC Shift Register Out0/Polynomial Preset Byte0
03	ECC Shift Register Out1/Polynomial Preset Byte1
04	ECC Shift Register Out2/Polynomial Preset Byte2
05	ECC Shift Register Out3/Polynomial Preset Byte3
06	ECC Shift Register Out4/Polynomial Preset Byte4
07	ECC Shift Register Out5/Polynomial Preset Byte5
08	Polynomial Tap Byte0
09	Polynomial Tap Byte1
0A	Polynomial Tap Byte2
0B	Polynomial Tap Byte3
0C	Polynomial Tap Byte4
0D	Polynomial Tap Byte5
0E	ECC Control
0F	Header Byte Count/Interlock
10	Drive Command Register
11	Operation Command Register
12	Start Sector Number
13	Number of Sector Operations
14	Header Byte0 Pattern
15	Header Byte1 Pattern
16	Header Byte2 Pattern
17	Header Byte3 Pattern
18	Header Byte4 Pattern
19	Header Byte5 Pattern
1A	Local Data Byte Count L
1B	Remote Data Byte Count H
1C	DMA Address Byte0
1D	DMA Address Byte1
1E	DMA Address Byte2
1F	DMA Address Byte3

Hex Address	Name
20	Data Postamble Byte Count
21	ID Preamble Byte Count
22	ID Address Mark Byte Count
23	ID Synch Byte Count
24	Header Byte0 Control Register
25	Header Byte1 Control Register
26	Header Byte2 Control Register
27	Header Byte3 Control Register
28	Header Byte4 Control Register
29	Header Byte5 Control Register
2A	Data External ECC Byte Count
2B	ID External ECC Byte Count
2C	ID Postamble Byte Count
2D	Data Preamble Byte Count
2E	Data Address Mark Byte Count
2F	Data Synch Byte Count
30	Data Postamble Pattern
31	ID Preamble Pattern
32	ID Address Mark Pattern
33	ID Synch Pattern
34	Gap Byte Count
35	Disk Format Register
36	Local Transfer Reg/Header Diagnostic Readback
37	Remote Transfer Register
38	Sector Byte Count L
39	Sector Byte Count H
3A	Gap Pattern
3B	Data Format Pattern
3C	ID Postamble Pattern
3D	Data Preamble Pattern
3E	Data Address Mark Pattern
3F	Data Synch Pattern

TABLE 4.2. Summary of Control Registers

Address	Register Name	General Operations
00H 01H 0FH	Status Register Error Register Header Byte Count	Disk Operation, DMA Status (Read) Error Determination of Operation
10H 11H	Drive Command Operation Command	Start Disk Operation Reset, Remote DMA, INTR Operation
12H 13H	Start Sector No. of Sector Operations	Can Contain Sector Number Used in Multi-Sector Operation
35H 36H	Disk Format Header Diagnostic	MFM, Hard Sector, External ECC

THE ECC/CRC REGISTERS

The ECC/CRC registers (addresses 02H to 0EH) are used to set up the DDC for desired error detection and correction configuration. Registers 02H to 07H are read-write and contain the preset pattern for the internal ECC. The preset pattern is the data that the ECC shift register is initialized to prior to an operation. Typically this is all ones. During a correcton cycle, reading these registers provides the syndrome bytes to correct the erroneous data. Registers 08H to 0DH are write-only and are used to load in the internal ECC polynomial required for the drive format selected. The ECC Control register (address 0EH) is used for selecting the internal ECC Correction Span, inversion of input or output check bits to the ECC register and ECC encapsulation (the mode that includes sync bytes (address marks) in the check bit calculation). These registers are listed in Table 4.3.

THE FORMAT REGISTERS

The Format registers, Table 4.4, (addresses 20H to 2FH, 30H to 35H and 38H to 3FH) determine and control the format of the fixed fields for the selected drive type according to the selected format. (*Figure 4.3* shows the Sector Format fields incorporated in the DDC). Registers 20H to 2FH contain the byte count of the fixed fields along with the 6 Header Control registers while Registers 30H to 35H and 38H to 3FH contain the patterns of the fixed fields along with the Inter Sector Gap Count and the Sector Byte Count registers. These Registers are shown in Table 4.4. Since almost every pattern register has an associated count register (which controls the repetition number of its field) or a control register, Table 4.4 is organized to show both together.

THE DMA REGISTERS

The DMA registers consist of the Local Transfer register (address 36H), Remote Transfer register (address 37H), and count and address registers, 1AH to 1FH. The Local Transfer register controls the data transfers between the DDC and buffer memory by controlling data and address bus lengths, byte ordering, memory cycle and the burst length. The Remote Transfer register, on the other hand, controls the data transfers between the buffer memory and the system I/O port in dual bus architectures. In addition to the data and address bus lengths, the memory cycle length, and determining the burst lengths, it also controls the transfers in the dual channel DMA mode.

Registers 1AH and 1BH determine the byte count required in a remote data transfer while registers 1CH to 1FH are DMA Address bytes 0 to 3 for local and remote transfers.

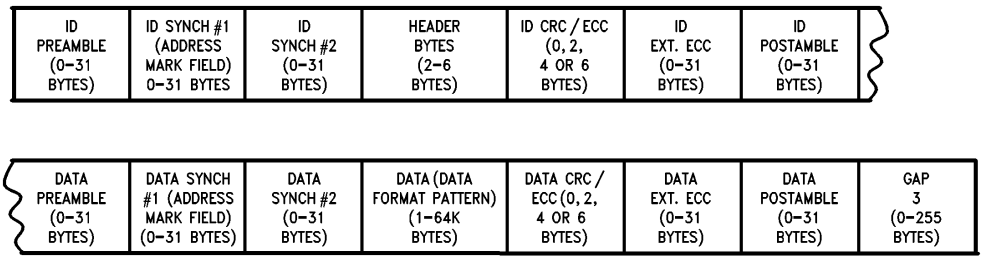
Table 4.5 lists the DMA control and address registers, and their function when the DDC is used in either dual channel or single channel mode.

TABLE 4.3. ECC Control Registers

Shift Reg/ Polynomial Preset	Polynomial Tap	Register Description
02	08	ECC Shift Reg/Poly Preset and Tap 0
03	09	ECC Shift Reg/Poly Preset and Tap 1
04	0A	ECC Shift Reg/Poly Preset and Tap 2
05	0B	ECC Shift Reg/Poly Preset and Tap 3
06	0C	ECC Shift Reg/Poly Preset and Tap 4
07	0D	ECC Shift Reg/Poly Preset and Tap 5
0EH		ECC Control
08H, 09H		Data Byte Count

TABLE 4.4. Format Count, Control and Pattern Registers

Count/ Control Reg	Pattern Reg	Header/ID Field Register Descriptions	Count Reg	Pattern Reg	Data Field/ECC Format Register Description
20	30	Data Postamble	2A	—	Data External ECC
21	31	ID Preamble	2B	—	ID External ECC
22	32	ID Synch Field 1	2C	3C	ID Postamble
23	33	ID Synch Field 2	2D	3D	Data Preamble
			2E	3E	Data Address Mark
24	14	Header Byte 0	2F	3F	Data Synch
25	15	Header Byte 1	34	3A	Post Sector Gap
26	16	Header Byte 2			
27	17	Header Byte 3	—	3B	Data Format
28	18	Header Byte 4	38	—	Sector Byte (LSB)
29	19	Header Byte 5	39	—	Sector Byte (MSB)



TL/F/8663–70

FIGURE 4.3. Sector Format Fields Incorporated in the DDC

TABLE 4.5. DMA Registers and Functions in Dual or Single Channel Mode

Addrs	Register Name	Single Channel	Dual Channel
1AH	Remote Count	NA	LSB Data Xfer Count
1BH	Remote Count	NA	MSB Data Xfer Count
1CH	DMA Addr 0	A0–A7	A0–A7 Local DMA
1DH	DMA Addr 1	A8–A15	A8–A15 Local DMA
1EH	DMA Addr 2	A16–A23	A0–A7 Remote DMA
1FH	DMA Addr 3	A24–A32	A8–A15 Remote DMA
36H	Local Transfer	Configures Local or Single Channel	
37H	Remote Transfer	Configures Remote (Dual Channel Mode)	
37H	DMA Sector Count	DMA Sector Counting (Dual Channel)	

4.3.3 The FIFO

The primary function of the DDC is to transfer data between disk and the system. The DDC has been configured so that during a disk data transfer operation, it does not occupy the bus for the whole disk transfer. Instead, it allows burst transfers so that the bus is free between the bursts for normal system usage. Systems with a main microprocessor and main memory will interface directly to the DDC. In this type of application, burst data transfers will require occupancy of the main system bus, so use of the bus must be granted at the discretion of the system. For example, if the system is performing a higher priority operation, it must not relinquish the bus for disk data transfer.

Once the bus has been relinquished, the system is held from performing other operations and a burst of data is then transferred. For the DDC, these requirements mean first, that some degree of data buffering is necessary to store the continuous arrival or removal of disk data, and second, that when the bus is granted, transfer must be fast. The amount of data buffering will be dependent on the system, but the majority of low-end systems should be able to respond to a data transfer request from the DDC within 50 μs. Most disk drives in this kind of application run at a data rate of 5 Mbits/sec, or one bit every 200 ns. Typically then, the data buffer must be able to store around 250 bits. A 32-byte FIFO has been included on the DDC enabling it to operate with most bus systems.

The data is transferred between the FIFO and the local memory (dual channel mode) or system memory in different burst thresholds, 1, 4, 8 or 12 words for 16-bit wide word transfers or 2, 8, 16 or 24 bytes for byte wide transfers. In a disk read operation when the FIFO fills to the selected threshold level with disk data, the DMA controller issues a data transfer request. The FIFO continues to fill. Whenever the DMA gets access to the system bus, it transfers data in selected bursts. These bursts may be of fixed length or until the FIFO empties. If the DMA request is not acknowledged, and the disk data fills the FIFO before it reaches to its maximum 32-bytes capacity, the FIFO Data Lost error occurs and the operation is aborted.

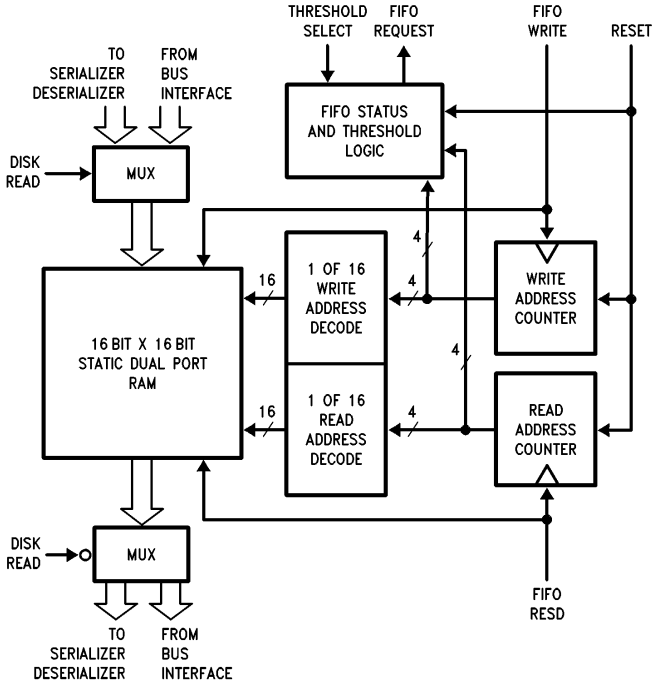
Conversely, in a disk write operation, the FIFO is first filled. It then requests a new data burst when it empties to below the selected threshold level. Depending on burst mode, the DMA will then request a fixed number of bytes or fill the FIFO. If the FIFO completely empties during the operation, a FIFO Data Lost error is again generated.

Figure 4.4 shows the basic blocks that compose the FIFO. It consists of a 16 x 16 bit dual port RAM array, which is addressed by a read counter and a write counter. These counters are decoded to address the array and also to feed the status logic which takes the difference between these pointers to generate the threshold signals. The input and output data ports of the FIFO can be connected to the serializer/deserializer or bus interface block. The direction of the FIFO is determined by the disk operation, read or write. An 8-bit bus interface is supported in 8-bit transfer mode by treating the FIFO as two interleaved banks of 8 bits.

4.3.4 The DMA

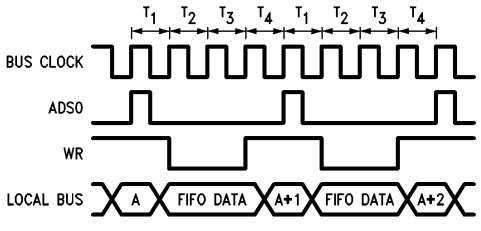
The DDC has an important feature that helps both in saving external ICs and in increasing data throughput, namely powerful DMA capability. With on-chip DMA capability, there is no need to dedicate a channel of a DMA controller for disk transfers. This offers two advantages: first, it may alleviate the need for a DMA controller chip, and second, memory transfer time will be faster because DMA controllers are relatively slow, usually around 2 Mbytes/sec maximum throughput. The DDC can transfer data much faster than this, especially when selected to transfer 16 bits each cycle. This faster transfer rate offers a much lower bus occupancy time, freeing the bus sooner for other usage.

When using the DMA capability, the DDC becomes bus master during the data transfer operation. In bus master mode the DDC issues incrementing address information at the start of each memory cycle. Each read or write memory cycle takes four clock periods, using a similar sequence to a four clock cycle microprocessor with multiplexed address and data bus. In some cases a five clock cycle sequence is used when two address words must be multiplexed to form the DMA address. *Figure 4.5* shows a typical read or write cycle of 4.



TL/F/8663–71

FIGURE 4.4. Simplified Block Diagram of DDC's FIFO



TL/F/8663–72

FIGURE 4.5. DDC-to-Memory Word Transfers (16-Bit Address)

The block diagram on the DMA section is shown in *Figure 4.6*. The heart of the DMA is a sequencer/PLA that uses inputs from the FIFO (FIFO Request) and disk control logic (DC Ready) for local DMA. It also generates signals for the remote channel to determine if a transfer is necessary. It then issues the requests, and once the acknowledge is received, the PLA sequences through the DMA cycle by placing the address onto the data/address bus (via the bus interface block) and manipulating ADS0, ADS1, read, and write strobes. The various counters are then incremented. In addition to the address counters, there are counters to keep track of bytes per sector (local and remote channel), bytes per header (local and remote), number of total sector operations, and remote counter burst length. The DMA sector counter is used in the tracking mode (described later), and enables the destination DMA whenever it is not zero. The remote transfer counter is decremented after each remote transfer and is used to set the total length of the transfer. When it reaches zero the sequencer halts remote DMA operations.

The DDC can be configured into two DMA modes, Single Channel and Dual Channel. The Dual Channel Mode has two sub-modes: Tracking and Non-Tracking. The general operation of these modes is described below.

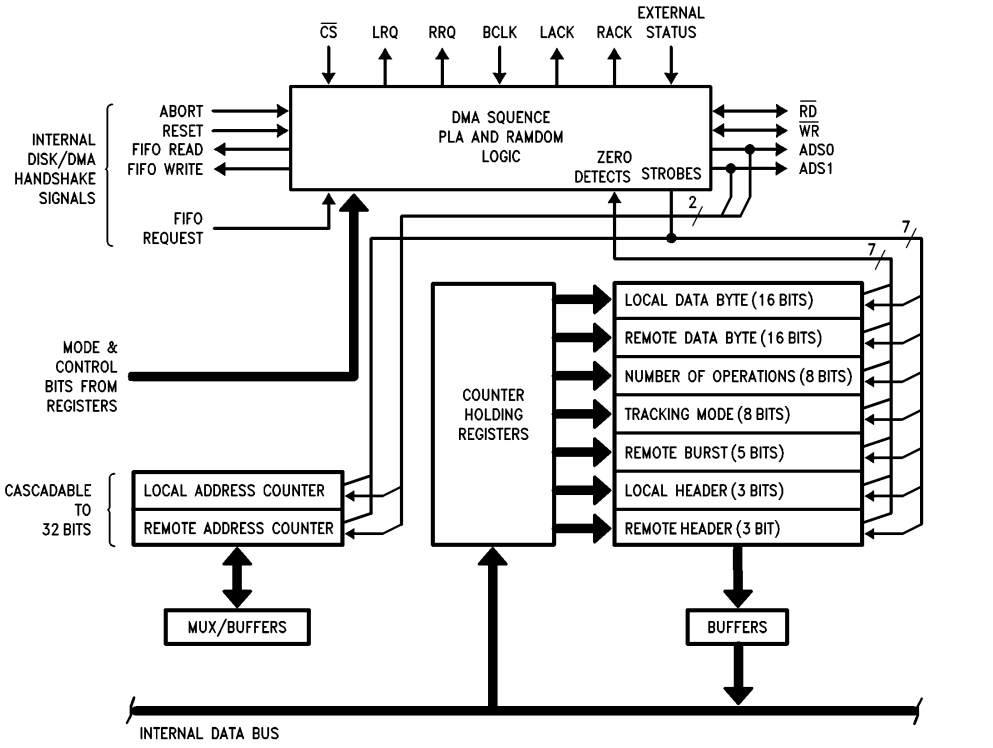
SINGLE CHANNEL

In single channel DMA mode the DDC interfaces directly to main memory having 32 address bits available to access up to 4 Gigabytes of memory. *Figure 4.7(a)* shows the DDC in single channel DMA configuration. The lower 16 address bits are normally issued at the start of each memory cycle so that most memory cycles comprise four clocks. The upper 16 bits are issued at the start of an operation or if the lower 16 bits rollover. In these cases, the memory cycle becomes 5 clock periods, the upper 16 bits are issued during first clock period and the cycle then completes the next four clock periods as in normal read or write operation. The upper two bytes of address information should be latched during the first clock period.

DUAL CHANNEL

Some systems may require the DDC to interface to a local bus with its own dedicated buffer memory before it interfaces to the main system. Such an application would be in intelligent disk drives that comply with the SCSI (SASI) or IPI interfaces. Intelligent drives may receive or transmit data whenever the controlling unit is ready. Another application could be in higher end systems, where the main memory is hooked onto a main bus such as the Q-BUS™, MULTIBUS®, VME or Future Bus. These buses are usually very busy and often impose high latency times while the main processor is performing important tasks. Once the bus is free, it is advantageous to be able to transfer all the disk information in as short a time as possible to minimize bus occupancy. For these types of applications, the dual channel capability of the DDC is ideal.

In the dual channel mode, the DMA generates a 16-bit address for both the local and remote transfers. The local channel controls the data transfers between the FIFO and the local buffer memory. The remote channel, on the other hand, controls data transfers between the addressed local buffer memory and the main system bus through an I/O port. A DMA channel in the system DMA controller could



TL/F/8663–73

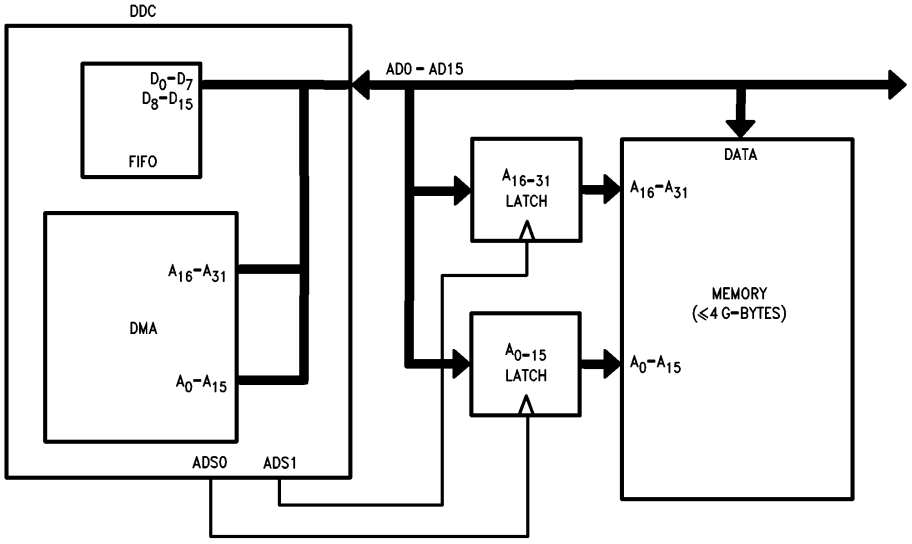
FIGURE 4.6. Block Diagram for Dual/Single Channel DMA Controller

then transfer the data from the port to system memory. Refer to *Figure 4.7(b)*. The local request and remote request are issued when the DDC requires a transfer, the microprocessor and bus arbiter respond by acknowledging the requests. If both channels are requesting, the system should arbitrate the acknowledge, however the local DMA has a higher priority if both requests are acknowledged.

The dual channel mode can be further divided into two modes, Tracking (non-overlapping) and Non-Tracking (overlapping). These dual channel modes are described below.

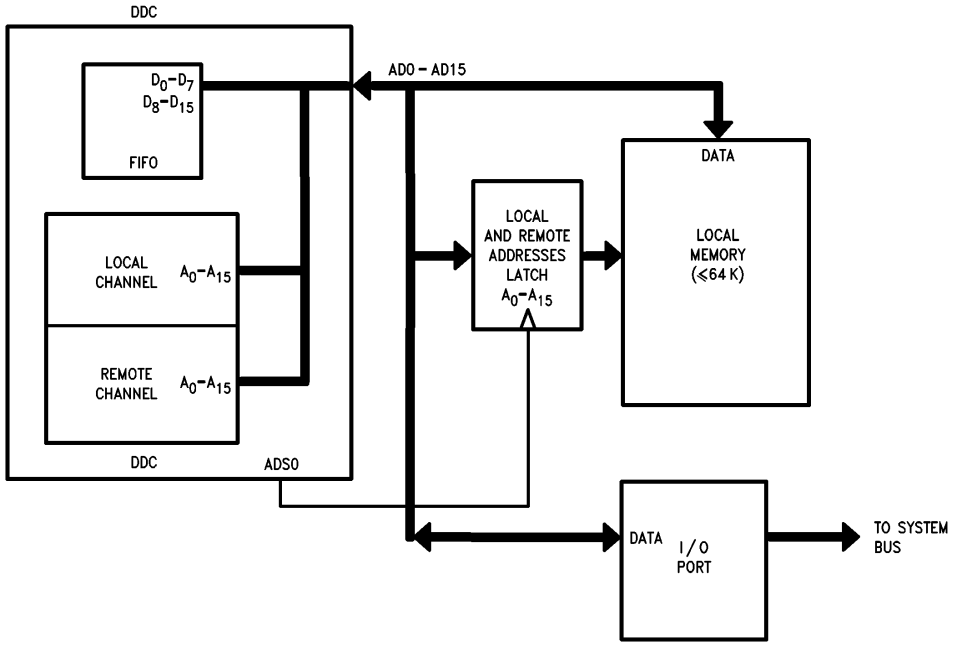
TRACKING OR NON-OVERLAPPING MODE

In Tracking or Non-Overlapping Dual DMA mode the DMA controls the local and remote transfers in such a way that the local buffer memory appears to the system as a large



TL/F/8663–74

(a). The DDC in Single Channel DMA Mode



TL/F/8663–75

(b). The DDC in Dual Channel DMA Mode

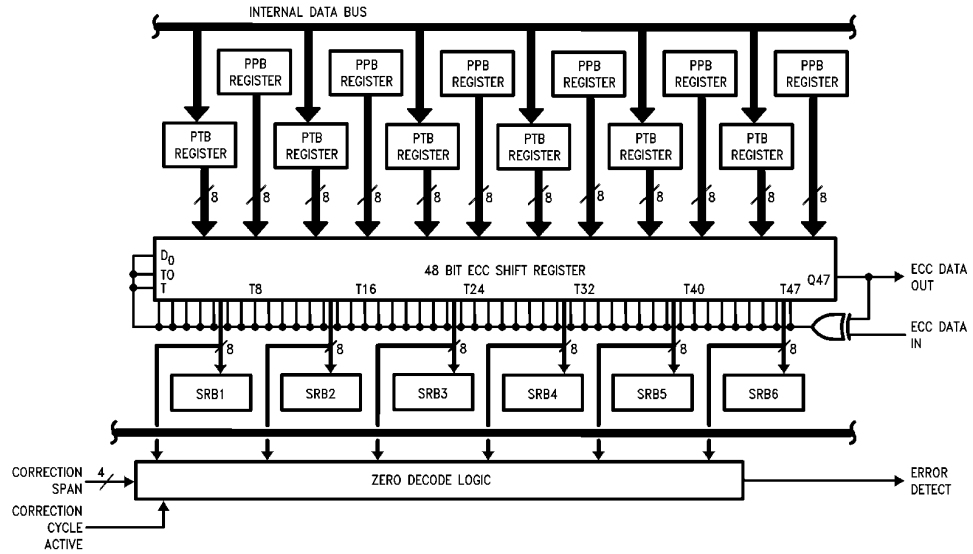
FIGURE 4.7

FIFO. This allows the system to transfer data to or from the local buffer memory whenever the system is ready, but with protection against overlapping of disk data. Basically this mode of operation is more applicable to multi-sector operations where the DDC efficiently interleaves bursts of data into and out of the buffer memory using both channels. Each channel has 16 address bits, allowing up to 65k of buffer memory to be used by the two channels. While doing a remote transfer in a disk read operation, data is read (bursted out) from the same local memory area where it was written to (bursted in) during the local transfer. Similarly, in a disk write operation, data is read from the same local memory area where it was written to during the remote transfer. In both cases, buffer memory addresses for local and remote transfers are issued such that data is never overlapped. If the two channels track very closely, then large amounts of contiguous data can be transferred, making the buffer memory appear to be a multi-megabyte FIFO.

The protection against overlapping of disk data is enabled by use of the DMA Sector Transfer Counter in the DDC. The counter is initially reset at the start of the operation. It is then incremented each time the source has transferred a sector of data into the buffer memory, and is decremented each time the destination has transferred a sector of data from the buffer memory. Whenever the count is zero, destination transfers are inhibited, so preventing the destination from catching up and overtaking the source transfers.

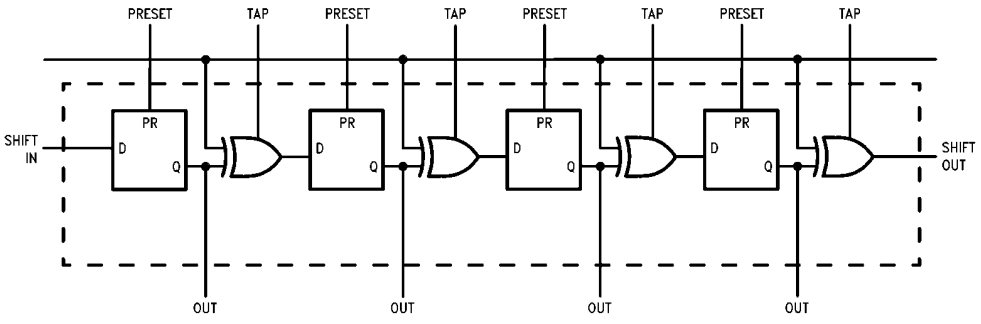
NON-TRACKING OR OVERLAPPING MODE

Some systems require that the two DMA channel are completely independent of each other. For this type of application the DDC can be configured to set up both DMA channels independently. The Local and Remote operations may be from different areas of memory or common areas. The local DMA may already be performing an operation when the Remote DMA is instructed to begin an operation. Likewise a Remote operation can be in progress when the Local operation is initialized. One operation can be for reading memory and the other for writing. This puts the burden on the user to protect from overwriting the buffer memory. In other words, the controlling microprocessor has the responsibility of ensuring that no memory overwriting occurs when both local and remote transfers are in progress. This mode



(a). ECC Shift Register Logic

TL/F/8663–76



TL/F/8663–77

(b). Detail of 4-Bit Section of Shift Register

PPB = Polynomial Preset Byte
PTB = Polynomial Tap Byte
SRB = Shift Register Buffer

FIGURE 4.8

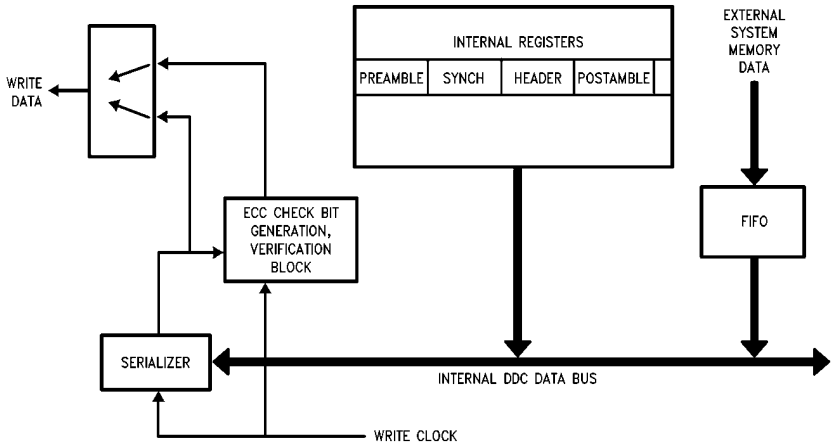
also gives the user the freedom to use the remote DMA controller with no restrictions, even for general non-disk transfers, such as for high level commands or status tranfers.

4.3.5 Error Detection and Correction

A fixed CRC code for detecting errors only, and a programmable ECC code for detecting and correcting errors can be used via the on-chip Error Detection and Correction block. The DDC has full polynomial programmability for 32 bits or 48 bits of ECC appendage, along with a programmable correction span from 3 to 15 bits and a programmable preset. The DDC can also be easily interfaced to external ECC circuitry if desired.

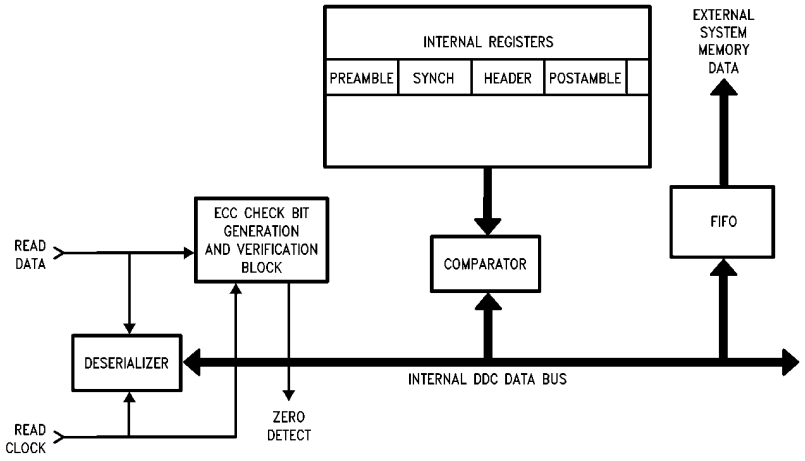
The Error Detection and Correction block mainly consists of a 48-bit shift register with XOR taps. It generates and then appends the check bits to header and data fields. The CRC uses the standard CCITT polynomial that provides 16 generated check bits. The CRC-CCITT code is hardware implemented in the DDC. The ECC code may be a Fire code or a computer generated code with either 32 or 48 generated check bits. The selected 32- or 48-bit ECC polynomial can be implemented by means of the 48-bit shift register and the Polynomial Tap and Preset Byte registers (addresses 02H thru 0DH).

The ECC Shift register logic is shown in *Figure 4.8*. In this figure the internal data bus connects to the Polynomial Preset Byte Registers (PPB). These register bits feed into the shift register latches. Any bit set in the PPBs will preset a corresponding flip-flop before an ECC operation begins, while all others will not be set. The Polynomial Tap Byte Registers feed the XOR gates in the ECC Shift Register. When a PTB bit is reset, the associated XOR gate is enabled for a particular ECC register bit. This effectively creates the ECC polynomial tap. The outputs of each shift register flip-flop bit input to a set of output buffers which drive the internal DDC data bus. This enables reading of the ECC registers. The ECC outputs also go to a combinational logic block that decodes the contents of the ECC shift register and the correction span. If at the end of a detection cycle the ECC shift register contains zero then no error was detected.



TL/F/8663–78

(a) Disk Write Operation



TL/F/8663–79

(b) Disk Read Operation

FIGURE 4.9

INTERNAL CHECK BIT GENERATION AND CHECKING

When writing to the disk, the CRC/ECC shift register is preset from the Preset ECC registers. At the same time that the DDC is outputting either the Header field or Data field bytes as a serial data stream through the Serializer, it is feeding them serially into the CRC/ECC shift register as shown in *Figure 4.9(a)*. When the last bit of the Header or Data field has been transmitted out of the DDC, the DDC begins shifting out the generated check bits from the CRC/ECC shift register starting with the MSB and ending with the bit 0. After the specified number of check bits have been appended the DDC internally switches to the next field.

When reading from the disk, the shift register is first preset from the Preset ECC registers before the read data operation begins. The incoming Header or Data field is serially fed into the CRC/ECC shift register as shown in *Figure 4.9(b)*. When all the Header or Data field bits and all the generated check bits have entered the CRC/ECC shift register, the status of the bits in the CRC/ECC shift register is checked for the all zeroes condition. If this condition is met, it signifies the field contains no errors. If any of the CRC/ECC shift register bits are high, the field contains an error. The Header and the Data field errors are indicated by the Status and Error registers respectively.

INTERNAL ERROR CORRECTION

The DDC is capable of correcting from 3 to 15 contiguous bits in error for selected 32- or 48-bit polynomials. The value desired is set in the ECC Control Register, in other words, it can correct a span of the selected amount. The DDC can be put in the Correction Mode through the Operation Command register. The CRC/ECC shift register contains a non-zero 32- or 48-bit pattern which is used to determine the location of the bytes in error and the error pattern. The most significant 3 to 15 bits of the 32 or 48 bits are selected as the Syndrome bits, while the rest are checked for a zero detect. During the correction mode, the CRC/ECC shift register is reverse shifted. Also, reverse shifting guarantees that the correction cycle will be completed within the time it takes to read one sector of the disk.

When the reverse shifting of the shift register begins, the Data Byte Counter register begins decrementing from its preloaded value of the number of data and ECC bytes in the sector. Another 3-bit counter is used to keep track of byte boundaries in the serial bit stream of the whole sector. Reverse shifting continues until all zeroes are detected in the (32-C) or (48-C) bits of the CRC/ECC shift register (where C is the correction span selected). When this occurs and the 3-bit counter contains all zeroes, the clock is stopped. At this point the C syndrome bits contain the bit error pattern of the byte indicated by the Data Byte Counter register. If the 3-bit counter count was not zero when the zero-detect occurred, then the CRC/ECC shift register has to undergo further reverse shifts to byte align the right byte in error. If the Data Byte counter register count goes to zero and the zero-detect is not obtained, then the error is non-correctable. If either the zeroes condition is determined or the Data Byte counter decrements to zero, an INTERRUPT is issued to indicate to the microprocessor that the correction cycle has finished.

The results of the correction cycle are indicated by the Status register. In the case of a correctable error, the error must be in either the Data field or the check bits of the ECC field or overlapping both fields. If the error is only in the ECC field then the memory data is correct and no further action is needed to complete the correction. But if the error is in the Data field then it can be corrected by XORing the C syndrome bits in the CRC/ECC shift register with the contents of the relevant memory location determined from the final Data Byte Counter register count.

EXTERNAL ECC

Some users may wish to use an ECC polynomial code with a different number of check bits. Some encoding schemes require a wider error correction span, or some users may prefer some high integrity ECC codes such as Reed-Solomon code. For these reasons DDC has been configured to interface easily to external ECC circuitry.

When the DDC is configured to utilize an external circuit for ECC code, the external ECC code may use any polynomial that generates from 1 to 31 bytes of check bits. The external circuitry is informed by the DDC when data is valid and when to generate check bits (for writing) or detect (when reading) through SDV, EEF and EXT STAT pins. Refer to Section 4.2 for pin description. The external ECC may be used to encapsulate internal CRC/ECC field as a confirmation of error detection.

4.3.6 The Serializer-Deserializer

This section of the DDC interfaces to the disk. The Serializer takes byte wide data either from the internal registers or the FIFO into a shift register and serially outputs the bits in a continuous bit stream, starting with the most significant bit. This serial data is then fed into the Error Detection and Correction block for check bit generation. When the CRC/ECC appendage is about to begin, the serializer stops shifting out and the Error Detection and Correction block begins shifting out the check bits, again most significant bit first. At the end of the appendage the Serializer starts shifting out further information to finish the segment. The serial data passes through the MFM Encoding and Precompensation block, if selected, or is output to the external encoder.

During a read operation, the incoming serial NRZ data feeds into the Deserializer and the CRC/ECC block, most significant bit first. The Comparator continually checks the incoming data for a synchronizing pattern that matches the pattern loaded into the internal pattern registers. Once a match occurs, the DDC then knows the byte boundary such that all further bytes from the deserializer are byte synchronized to the boundary first established. CRC/ECC fields, postamble, and preamble fields are not required to be deserialized, and do not enter the deserializer. Once the address mark and/or synch byte have aligned, the header bytes preloaded into the internal registers are sequentially output to the Comparator as each incoming byte is ready. The Comparator checks all the header bytes in turn for a match. If a full match is detected, the DDC checks the CRC/ECC appendage and prepares for the following data field. Finally, the data field is read, and serial data bytes are converted to parallel. They then enter the FIFO from the Deserializer to be transferred by the DMA.

The basic blocks associated with the Serializer/Deserializer are shown in *Figure 4.10*. For Serialization, data from either the Pattern Registers, Sector Counter, or FIFO is multiplexed to a holding latch. The holding latch will load the Serializer/Deserializer shift register at the appropriate time

(determined by the disk controller's PLA). This data is shifted through the EEC/Data MUX and the MFM or NRZ logic to the Serial Data output pin. When serializing data the write clock feeds the shift register.

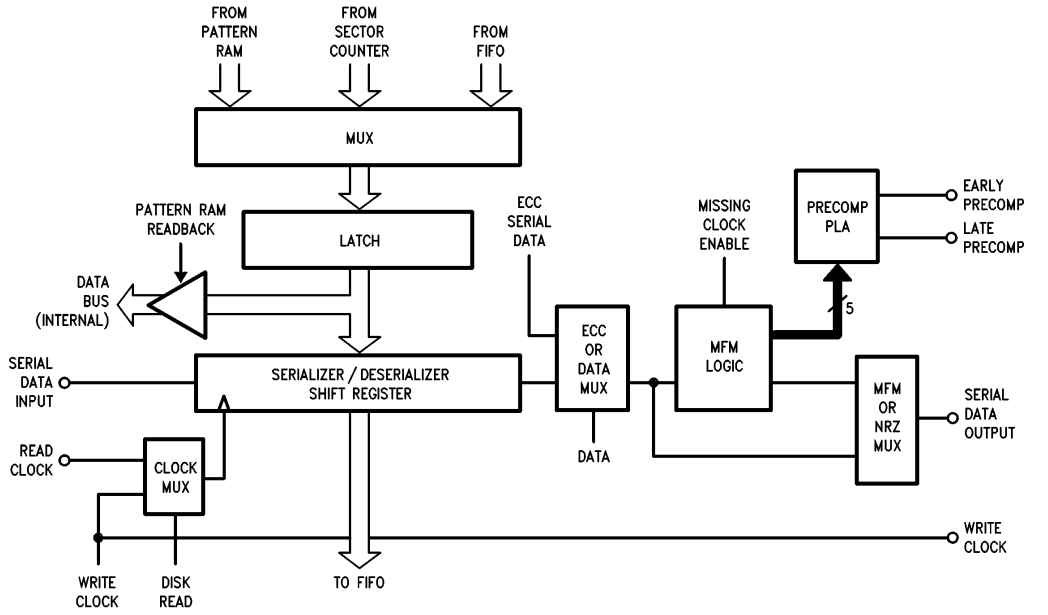
For Deserialization of data, Read Data and Read Clock feeds the internal data bus. Once byte alignment is determined, a byte clock controls loading data into the FIFO.

MFM ENCODING AND PRECOMPENSATION

The DDC can be set to output MFM encoded data and Precompensation information in a disk write operation. The MFM Encoding and Precompensation block of the DDC consists of a 5-bit shift register, and MFM and Precompensation encode logic. The NRZ data coming out of the Serializer passes through the shift register and then is fed into the MFM and Precomp encode logic. The MFM Encode logic converts this NRZ data into MFM and also inserts the missing clocks when Address Mark fields are required to be written to the disk.

The DDC can be programmed to output two control signals, EARLY PRECOMP or LATE PRECOMP, if precompensation is desired. The information on these output pins is then used by the external Precomp Circuitry (MUX, Delay logic, etc.) to perform the actual Precompensation. These pins perform an algorithm that compensates for the bit shifting that occurs when the magnetic flux transitions are recorded on the disk.

The MFM Encoder and precompensation block are shown in *Figure 4.11* also. After serialization, logic performs the MFM encoding. 5 bits from the encoder monitor previous and subsequent data to determine whether early, late, or no precompensation is needed.



TL/F/8663–80

FIGURE 4.10. Serializer/Deserializer Block Diagram

CHAPTER 5 The DDC Registers and Commands

5.0 INTRODUCTION

In this Chapter, the DDC's (DP8466) registers and its command operation will be described in more detail. Initially, a general overview of the registers is given. This section can be used to supplement the data sheet register descriptions. Then details on the various methods of formulating commands and operations are described. Understanding the wide variety of operating modes presented in this section will enable a better understanding of which modes are suitable for which applications.

5.1 INTERNAL REGISTERS AND COUNTERS

In this section a detailed description of internal Command, ECC/CRC, Format and DMA pattern, control and count registers, and various counters is provided. A summary description of the registers is given below, as well as a description of important command bits. Additional details of the DDC's registers are given in the DP8466's Data Sheet.

5.1.1 Command Registers and Counters

The Command registers and counters are listed in *Figure 5.1* and explained in the following paragraphs. These registers include the drive and Operations Command registers, Disk Format register, Error and Status registers, Sector Counter and Sector Operations Counter registers.

Name	Hex Address
Drive Command Register	10H
Operation Command Register	11H
Disk Format Register	35H
Status Register	00H
Error Register	01H
Sector Counter	12H
Number Of Sector Operations	13H
Header Byte Count Register	0FH
Interlock Complete Signal	
Header Diagnostic Readback	36H

FIGURE 5.1. Command Registers and Counters

DRIVE COMMAND (DC) REGISTER (Address = 10H)

The drive command register is shown in *Figure 5.2.* This register is loaded when the DDC is required to perform a command. Disk operations are started after loading this write-only register. This register can be loaded to start a new command when the Next Disk Command bit is set in the Status register. The bit descriptions are given below.

Re-Enable (RED)

When loading a disk command a zero should be written to this bit to permit normal operation. A one should be written to the bit to re-enable the DP8466 after either a reset via the RESET pin or RES bit in the Operation Command Register.

Start Operation (SAIS)

Bit SAIS determines when the operation for the command being written to the drive command register shall begin. If SAIS is high, the operation will begin when the DDC detects either an INDEX or SECTOR PULSE for hard sectored drives, or immediately for soft sectored drives. If SAIS is low the operation only begins when the DDC detects an INDEX PULSE.

Single/Multi Sector Operation (MSO)

Bit MSO of the drive command register indicates whether the operation is for just one sector or a number of sectors. If MSO is set high then the DDC can perform a multi-sector operation. Multi-sector operations usually are handled on logically (by sector address) contiguous rather than physically contiguous sectors. The DDC can perform both types of multi-sector operations.

D0	RED	RE-ENABLE	0 = No Action 1 = Re-Enabled
D1	SAIS	START OPERATION	0 = Start on Index Pulse 1 = Start on Index/Sector Pulse or Immediately
D2	MSO	SINGLE/MULTI-SECTOR OPERATION	0 = Single 1 = Multi
D3	FMT	FORMAT MODE	0 = Normal 1 = Format
D4 D5	HO1 HO2	HEADER OPERATIONS	00 = Ignore Header 01 = Compare Header 10 = Write Header 11 = Read Header
D6 D7	DO1 DO2	DATA OPERATIONS	00 = Invalid* 01 = Check Data 10 = Write Data 11 = Read Data

*Unless used with an Ignore Header operation. When HO1, HO2, DO1 and DO2 are written as 0 then no operation is performed. This is useful when Re-Enabling the DDC by setting the RED bit, for example.

FIGURE 5.2. Drive Command Register Bit Assignments

Format Mode (FMT)

The DDC can be set to format a disk by setting bit FMT high. The DO2, DO1, HO2, and HO1 bits must be set for a write-header/write-data operation, 1010. (For details please refer to the disk formatting section).

HEADER OPERATIONS (HO2, HO1)

Bits HO2 and HO1 determine the operation to be performed on the ID fields. The details are given below.

Ignore Header (HO2 = 0, HO1 = 0)

In an Ignore Header operation, after the byte alignment of Address Mark and/or Synch fields, the header bytes comparison and ECC/CRC checks are not performed. This results in reading or writing the associated data with respect to any sector encountered.

Compare Header (HO2 = 1, HO1 = 1)

Compare Header operation is the normal mode of header operation for locating the selected sector. In this operation the header bytes are compared with the corresponding values in the pattern registers and CRC/ECC checks are also carried out.

Write Header (HO2 = 1, HO1 = 0)

Write Header operation is normally performed during disk formatting. The Header bytes are written to the disk either from the pattern register or the buffer memory through FIFO depending upon the FIFO Table Format (FTF) bit of the Disk Format register.

Read Header (HO2 = 1, HO1 = 1)

Read Header operation performs CRC/ECC checks and transfers the header bytes into buffer memory through the FIFO for diagnostic purposes. If during this operation, a header containing a CRC/ECC error is encountered, the operation is aborted immediately and the header fault bit (Status Register) is set. An interrupt is generated, but no Error register bits are set.

DATA OPERATIONS (DO2, DO1)

Bits DO2 and DO1 determine operations to be performed on data fields.

Check Data (DO2 = 0, DO1 = 1)

After the preceding header operation and the byte alignment, the Check Data operation performs the CRC/ECC checks on the data fields. It does not transfer the data field to the FIFO and hence no data is transferred to memory via the DMA channels.

Read Data (DO2 = 1, DO1 = 0)

Read Data operation, on the other hand, transfers data to external memory via FIFO after performing CRC/ECC checks.

Write Data (DO2 = 1, DO1 = 1)

In Write Data operation, after the associated header operation, data bytes are written to the selected sector from the FIFO using the DMA channel.

OPERATION COMMAND (OC) REGISTER
(Address = 11H)

Operation command register, shown in *Figure 5.3,* is a write-only register and may be updated before each disk operation. This register controls some of the basic DDC operating modes, such as interrupts, starting remote DMA, starting a correction, and precompensation.

Reset (RES)

When RES is set high, the DDC enters the standby mode and remains in this mode until this bit is reset and a one is written to the RED bit in the Drive Command register. To properly reset the DDC this bit must remain set for 32 read clock periods and 4 bus clock periods (with both clocks applied). This bit has the same effect as setting the RESET input pin low. Read and Write Gate are deasserted, the FIFO is cleared, DMA requests are removed, the Error register is cleared and status register is cleared except the Next Disk Command is set, and the abort bit is reset. The parameter registers, sector counter and number of sectors registers are not affected.

Enable Interrupts (EI)

Bit EI, when set high, enables the DDC to issue interrupts when certain conditions are met, such as upon successful completion of a command.

Enable Header Complete Interrupts (EHI)

Setting bit EHI high will enable the DDC to issue an interrupt at the completion of a header operation so that it can be loaded with new information before the next ID operation begins. New commands can be loaded or some pattern information could be updated during the data operation of a sector.

D0	RES	RESET	0 = Normal Operation 1 = Reset DDC
D1	EI	ENABLE INTERRUPTS	0 = Disabled 1 = Enabled
D2	EHI	ENABLE HEADER COMPLETE INTERRUPTS	0 = Disabled 1 = Enabled
D3	SRI	START REMOTE INPUT	0 = No Remote Read 1 = Start Remote Read
D4	SRO	START REMOTE OUTPUT	0 = No Remote Write 1 = Start Remote Write
D5	EP	ENABLE PRECOMPENSATION	0 = Disabled 1 = Enabled
D6	SCC	START CORRECTION CYCLE	0 = No Cycle 1 = Start Cycle
D7	IR	INTERLOCK MODE	0 = No Interlock 1 = Interlock Mode

FIGURE 5.3. Operations Command Register

Start Remote Input (SRI)

The DDC initiates the transfer of data from the system to local memory when SRI is set high. This enables the start of remote transfers (non-tracking dual DMA mode) to the local buffer.

Start Remote Output (SRO)

The DDC initiates the transfer of data from local memory to system when bit SRO is set high. This starts remote transfers from the local buffer.

Enable Precompensation (EP)

The DDC will allow precompensation if bit EP of the operation command register is set high. This bit is valid only if the AMF/EARLY PRECOMP and AME/LATE PRECOMP pins are configured as write precompensation control pins.

Start Correction Cycle (SCC)

The internal correction cycle is initiated by setting bit SCC high (see ECC section).

Interlock Mode (IR)

In some situations, if it is desired to update the header bytes (during disk formatting) or issue a new drive command before the next command, the DDC must be put in Interlock mode by setting bit IR high.

In interlock mode, after every header operation, the DDC issues an interrupt after Header Match Complete flag, (HMC), in the status register goes high, indicating that the information can be updated before the beginning of the next header operation (or during the current data operation). Within this time, microprocessor has to update the information (header bytes or drive command register) and then write the header byte count to the Header Byte count register to indicate completion of update.

DISK FORMAT (DF) REGISTER (Address 35H)

Disk Format register, shown in *Figure 5.4,* is a write-only register and is usually updated when a different drive type is selected. This register controls some of the major format features of a disk, such as MFM, type of ECC/CRC, and configuring address marks.

D0	MFM	NRZ/MFM ENCODE	0 = NRZ 1 = MFM
D1	SAM	START WITH ADDRESS MARK	0 = Start with Preamble 1 = Start with AM
D2	HSS	HARD OR SOFT SECTOR	0 = Soft 1 = Hard
D3	FTF	FIFO TABLE FORMAT	0 = Use Registers 1 = Use FIFO
D4 D5	IH1 IH2	INTERNAL HEADER APPENDAGE	00 = None 01 = 16-bit CRC 10 = 32-bit ECC 11 = 48-bit ECC
D6 D7	ID1 ID2	INTERNAL DATA APPENDAGE	00 = None 01 = 16-bit CRC 10 = 32-bit ECC 11 = 48-bit ECC

FIGURE 5.4. Disk Format Register

MFM/NRZ Encode (MFM)

When writing to the disk, the DDC can output either MFM encoded data if MFM is high, or NRZ data if MFM is low.

Start with Address Mark (SAM)

If SAM is low, the format begins with the Preamble field followed by the Address Mark field. If SAM is high, the first field is Address Mark followed by the Preamble field. This supports ESDI or SMD drive formats.

Hard or Soft Sector (HSS)

The DDC can be configured for soft or hard sectored drives by setting bit HSS low or high, respectively.

FIFO Table Format (FTF)

If bit PTF is low, the header bytes are taken from the internal pattern registers during the disk formatting. If FTF is high, these bytes will be written from the FIFO through local DMA channel. This bit is used only during disk formatting.

Internal Header and Data Appendages (IH1, IH2, ID1, ID2)

The Internal Header Appendage (IH1 and IH2) and the Internal Data Appendage (ID1 and ID2) bits of the Disk Format register control CRC/ECC appendages for header and data fields. The appendage options which could be selected are no appendage (00), 16 bit CRC CCITT polynomial (01), and 32- and 48-bit programmable ECC codes (10, 11 respectively). If none of these internal ECC or CRC codes is selected, then an external header ECC code must be appended. Also, even if the internal codes are being appended, an external ECC code of up to 31 bytes may be added to encapsulate header, data and internal CRC/ECC fields.

STATUS (S) REGISTER (Address = 00H)

Figure 5.5 shows the flags in the Status register, which is a read-only register. This register provides status on current operation of the DDC. This includes DMA local and remote status, correction cycle status, operation error, and ready for next command's status. The flags are set or reset by conditions detected by the DDC. The flags are also reset when either the RESET input pin or RES bit in the operation command register are set. The flags in the Status register either provide the status of different operations in progress or the results of these operations.

D0	HF	HEADER FAULT
D1	NDC	NEXT DISK COMMAND
D2	HMC	HEADER MATCH COMPLETE
D3	LRG	LOCAL REQUEST
D4	RCB	REMOTE COMMAND BUSY
D5	LCB	LOCAL COMMAND BUSY
D6	CCA	CORRECTION CYCLE ACTIVE
D7	ED	ERROR DETECTED

FIGURE 5.5. Status Register

Header Fault (HF)

The Header Fault flag (HF) is set when a header field error is detected after a Compare Header operation. This is set when an ECC or CRC error is detected in any header field read. This may or may not be on the header that the DDC was looking for. During a disk operation if a header error was detected, and subsequently the correct sector was found, this bit will be reset. If the correct sector was not found, the DDC will timeout with the HF bit set. It is reset when the DDC is reset or when a new command is issued.

The HF will abort the operation immediately if the operation is a read header, and any header read has a CRC/ECC error. In this case no Error register bit is set.

Next Disk Command (NDC)

The Next Disk Command flag, when set, shows that the DDC is ready to receive a header byte update and another disk command. It is reset when a new disk command is issued to the DDC.

Header Match Completed (HMC)

In a Compare Header operation, after a header match, the Header Match Completed flag is set. This bit is reset at the end of the data operation. This flag is automatically set in Ignore and Write header operations or when any header field is encountered after a Read Header operation.

Local Request (LRQ)

The Local Request flag follows the LRQ exactly. It is set coincident with the LRQ output when the FIFO first requires a data transfer. The flag is reset whenever the LRQ pin is deasserted.

Remote Command Busy (RCB)

The Remote Command Busy flag is set at the start of a remote transfer operation and is reset at the completion of the last memory transfer. This can be used to determine if the remote DMA channel is in operation.

Local Command Busy (LCB)

The Local Command Busy flag remains set through the entire period the local DMA channel is busy in transferring data between the FIFO and buffer memory. This is the same function as the RCB except local DMA.

Correction Cycle Active (CCA)

The Correction Cycle Active flag is set at the beginning of a Correction Cycle (when the Start Correction Cycle bit is set in the Operation Command Register) and is reset at the end of the cycle whether the error is located or not.

Error Detected (ED)

Error Detected flag is set if any of the error flags in the Error register is set. This is the logical ORing of all the Error register bits.

ERROR (E) REGISTER (Address = 01H)

Error register, shown in *Figure 5.6,* is also a read-only register. The flags of this register are set by conditions within the DDC and reset by the next new command to the Drive Command register. The flags are also set low when either the RESET input pin is set low, or the Reset bit (RES) in the Operation register is set.

D0	HFASM	HEADER FAILED ALTHOUGH SECTOR MATCHED
D1	DFE	DATA FIELD ERROR
D2	SNF	SECTOR NOT FOUND
D3	SO	SECTOR OVERRUN
D4	NDS	NO DATA SYNCH
D5	DL	FIFO DATA LOST
D6	CF	CORRECTION FAILED
D7	LI	LATE INTERLOCK

FIGURE 5.6. Error Register

Header Failed Although Sector Matched (HFASM)

The HFASM (Header Failed Although Sector Matched) flag, when set, indicates that the Sector byte(s) of the header field match correctly but there is an error in other header byte(s). This flag can only be set if the Enable HFASM Function (EFH) bit of at least one of the Header Control registers is set high during a Compare Header operation. This bit will be set if any one of the header byte(s) with its EHF bit set matches but any other header bytes don't match. For example, assume a 6 byte header with the first two bytes having their EHF bit set. If during a compare header operation the first byte matched, but any of the 2nd through 6th bytes don't match this HFASM bit is set.

When executing a Compare Header-Check Data command, and this flag is set, the operation is aborted allowing the header bytes to be read from the FIFO for disk diagnostics. If this bit is set during a Compare Header-Read (or Write) Data, the command is aborted, but the header is not stored in the FIFO.

Data Field Error (DFE)

After a successful header match, if an internal CRC/ECC or external ECC error is detected during a Read Data or Check Data operation, the Data Field Error flag will be set.

Sector Not Found Error (SNF)

If the Header Match Completed flag of the Status register is not set for two consecutive index pulses in a Compare Header operation (i.e., the correct header was not found), then the Sector Not Found bit is set to indicate that the desired sector cannot be found. The operation is aborted and an interrupt is issued.

Sector Overrun (SO)

If during the time when data is being transferred between disk and the FIFO, either the SECTOR PULSE or INDEX PULSE inputs go active, then the sector is assumed to have overrun and the Sector Overrun flag (SO) is set. Operation is aborted. RGATE or WGATE are deactivated, and an interrupt is generated.

No Data Synch (NDS)

If an INDEX PULSE (hard or soft sectored drives) or a SECTOR PULSE (hard sectored only) is encountered while the DDC is looking to byte align on the first data synch byte (synch 1 or 2), this bit is set. Also if the DDC recognizes the first synch byte but not subsequent synch bytes then this bit is also set.

FIFO Data Lost (FDL)

This bit is set if the FIFO overflows during a Read Data operation. This normally would occur when the host does not allow the DMA to empty the FIFO faster than the Disk Data is being read. FIFO Data Lost is also set during a Write Data operation when the DDC empties the FIFO writing to the disk, and attempts to read the empty FIFO again. In either case the operation will be aborted.

Correction Failed (CF)

If by the end of a Correction Cycle (Data Byte Counter decrements to zero) the error has not been located, then the error is not correctable and the Correction Failed flag is set.

Late Interlock (LI)

If the Interlock Complete register is not written to by the time the next header field arrives, and the DDC is in Interlock mode, then the Late Interlock flag will be set.

START SECTOR (SC) REGISTER (Address = 12H)

The Start sector (and Number of Sector Operations Counter) facilitates multi-sector operations. This counter can be programmed to replace the header byte designated by the user to be the sector number. Thus, in a multi-sector operation, the Sector Counter is initialized with the sector number to start on. As a header is compared, and its data field is read or written, the sector counter is incremented at the end of that sector's header operation. This enables immediate operation on the next logical sector. Operation continues until the Number of Sector Operations Counter decrements to zero. The sector counter is enabled if bit substitute sector counter bit of any Header Control register is set high, and the contents of Sector Counter will be substituted for the corresponding Header Byte.

NUMBER OF SECTOR OPERATIONS COUNTER (NSO) REGISTER (Address = 13H)

In a multi-sector operation, the Sector Operations Counter is preset to the logical number of sectors to be consecutively operated on. It is decremented after every sector's header operation and when decremented to zero, terminates the active command.

HEADER BYTE COUNT (HBC)/INTERLOCK REGISTER (Address = 0FH)

This 4-bit read-write register, normally used during formatting, is loaded with the number of header bytes to be written to (or read from) disk. The allowable number of header bytes is from 2 to 6. On read-back, only the three least significant bits of this register are valid.

Another important function of this register is when the DDC is in the Interlock mode (explained in formatting section). During a multi-sector operation, if it is desired to update any header byte (for example in the case of disk formatting) or if the next drive command has to be changed, then this register must be written with the actual header byte count value after updating the header bytes. This will basically strobe the internal hardware to recognize that interlock (update) has occurred.

FIFO HEADER DIAGNOSTIC READBACK (HDR) REGISTER (Address = 36H)

This is a read-only register and allows the FIFO contents to be read one byte at a time. Normally, data or header bytes may be read for diagnostic purposes through this register (described later). There is no way to write to the FIFO except under DMA control. In order to read the header bytes in the same order as they are read from the disk, the Reverse Byte Ordering bit in the local transfer register must be reset.

5.1.2 Error Correction/Cyclic Redundancy (ECC/CRC) Registers and Counters

The ECC/CRC registers and counters are listed in *Figure 5.7*. These registers enable programming of various modes of ECC, the ECC pattern, and access to the ECC shift register for performing correction cycles. They are explained in the following paragraphs.

Name	Hex Address
ECC Shift Register Out0 Register	02H
ECC Shift Register Out1 Register	03H
ECC Shift Register Out2 Register	04H
ECC Shift Register Out3 Register	05H
ECC Shift Register Out4 Register	06H
ECC Shift Register Out5 Register	07H
Polynomial Preset (Byte0) Register	02H
Polynomial Preset (Byte1) Register	03H
Polynomial Preset (Byte2) Register	04H
Polynomial Preset (Byte3) Register	05H
Polynomial Preset (Byte4) Register	06H
Polynomial Preset (Byte5) Register	07H
Polynomial Tap (Byte0) Register	08H
Polynomial Tap (Byte1) Register	09H
Polynomial Tap (Byte2) Register	0AH
Polynomial Tap (Byte3) Register	0BH
Polynomial Tap (Byte4) Register	0CH
Polynomial Tap (Byte5) Register	0DH
ECC Control Register	OEH
Data Byte Count (LS) Register	08H
Data Byte Count (MS) Register	09H

FIGURE 5.7 ECC/CRC Registers and Counters

ECC SHIFT REGISTER OUT0–OUT5 REGISTERS (Address = 02–07H)

The 48-bit long CRC/ECC shift register of the DDC can be read through these 6 read-only registers at any time. If 32 byte ECC is used then only registers 0, 1, 4, and 5 are used. After a correction cycle has occurred, these registers contain the ECC syndrome bits. The memory address of the sector in error and the data bit in error are calculated by the μP, and then the bits of these registers are XORed with the data in order to correct the error (assuming the error is correctable).

POLYNOMIAL PRESET (PPB) 0–5 REGISTERS (Address = 02–07)

The selected ECC polynomial preset pattern is loaded into the ECC/CRC shift register from these six Polynomial Preset registers. These are write only registers. The preset bit pattern could be all ones, all zeroes or a combination. This is the value the ECC shift regsiter is loaded with prior to shifting in the ECC pattern. The most significant bit of PPB5 is the most significant polynomial bit, X^{47}, and the least significant shift register tap is the least significant bit of PTB0. For 32-bit ECC, PPB2 and PPB3 are set to all zeroes, and are not used.

POLYNOMIAL TAP BYTE (PTB) 0–5 REGISTERS (Address = 08–0DH)

The ECC shift register is tapped at every bit by an XOR element. Wherever an exclusive-OR tap is required into the 32-bit or 48-bit shift register a zero should be set in the corresponding PTB0–5 bits. All polynomial elements not in the equation (and hence not tapped) must be disabled by setting all these bits to a one. The tap X^{32} (or X^{48} for 48-bit polynomial) is always present and is not programmable. The taps X^{31} (or X^{47}) to X^{0} are fully programmable. The MSB of PTB5 corresponds to the most significant tap, X^{47}, and the LSB of PTB0 is the least significant tap, X^{0}. For 32-bit ECC, PTB2 and PTB3 are not used and must be set to all ones. In this case PTB5's MSB becomes X^{31}.

ECC CONTROL (EC) REGISTER (Address = 0EH)

The ECC Control register, shown in *Figure 5.8,* is a write-only register. This register works in conjunction with the Disk Format register to set the ECC modes. The bit description is given below.

D0 D1 D2 D3	CS0 CS1 CS2 CS3	CORRECTION SPAN SELECT 0011 thru 1111	 3 bit span thru 15 bit span
D4	HE	ENCAPSULATION HEADER	0 = Encapsulated 1 = Not Encapsulated
D5	IEO	INVERT ECC OUT	0 = Normal 1 = Inverted
D6	IDI	INVERT DATA IN	0 = Normal 1 = Inverted
D7	DEN	DATA ENCAPSULATION	0 = Encapsulated 1 = Not Encapsulated

FIGURE 5.8. ECC Control Register

Correction Span Select Bits (CSO–CS3)

The number of bits which the ECC circuit attempts to correct, generally known as the correction span, is determined by CS0–CS3. Errors longer than the correction span will be

treated as non correctable errors. The allowable correction span for 32-bit ECC is 3 to 15 bits and for 48-bit ECC it is 3–15 bits. Setting the CS bits to any correction span that is outside the maximum allowable range of 3–15 bits causes the CRC/ECC to default to a 3 bit correction span.

Header Encapsulation (HEN)

When this bit is reset, the bit patterns of the Synch and/or Address Mark fields are included in ECC/CRC calculations. Some disk formats want these bytes included in check bit calculations (i.e., IBM 3740 floppy format). When this bit is set the Synch and/or Address Mark fields are excluded from the CRC/ECC calculation, and only the header field bytes are included.

Invert ECC Data Out (IEO)

When the shift register data out bit is set high, all the data and check bits coming out of the ECC shift register are inverted in a disk write operation. Otherwise the ECC data is not inverted.

Invert Data in (IDI)

This bit controls data and check bits when they enter the ECC shift register during a disk read operation. When this bit is set high, both data and check bits will be inverted. When low true data is input.

Data Encapsulation (DEN)

The DEN bit performs the same function for the data field as the HEN bit does for the Header field. When reset DEN will include the Synch and Address Mark fields in the CRC or ECC calculations. If set these fields are not included in the check bit calculations.

DATA BYTE COUNTER REGISTERS
(Address = 08H, 09H)

The Data Byte Counter registers are used during a correction cycle, and are preset by the μP prior to starting a correction cycle. They are set to the sum of the number of bytes in the data and ECC fields of the sector just read. During the correction cycle, the data byte count is decremented after shifting by 8 bits in the ECC shift register each byte. At the completion of the correction cycle, and if the error is correctable, the contents of the data byte counters are added to the starting address of the sector in error to determine the location of the memory byte or bytes in error. Details on this, are provided later.

5.1.3 Format Pattern and Count Registers

The Pattern, Count and Control registers used during disk formatting are listed in *Figure 5.9* and explained in the following paragraphs.

PATTERN REGISTERS
(Address = 30–33H, 3A–3FH, 14–19H)

The pattern registers hold byte information for the various fields of a formatted disk. These registers are written to the disk during a format operation. The Synch or Address Mark, Header, and ID postable pattern are read and compared to the pattern registers during a Compare Header operation. The Data Address Mark, and Data Synch are compared when doing a data field read, and are written to the disk during a disk data write or format. Associated with each pattern register (except the header pattern registers) is a byte repetition counter register that sets the field length, described below.

All the pattern registers listed in *Figure 5.9* are preloaded with the value of their respective fields such as ID and Data fields. The fields which are allowed in the DDC pattern registers are ID and Data Preamble, Address Mark, Synch, Postamble and Gap. Up to six header byte patterns can be programmed, thus enabling a header field of six bytes (excluding synch and preamble).

Name	Hex Address
ID Preamble Register	31H
ID Preamble Byte Count Register	21H
ID Synch #1 (AM) Pattern Register	32H
ID Synch #1 (AM) Byte Count Register	22H
ID Synch #2 Pattern Register	33H
ID Synch #2 Pattern Register	23H
Header (Byte0) Pattern Register	14H
Header (Byte0) Control Register	24H
Header (Byte1) Pattern Register	15H
Header (Byte1) Control Register	25H
Header (Byte2) Pattern Register	16H
Header (Byte2) Control Register	26H
Header (Byte3) Pattern Register	17H
Header (Byte3) Control Register	27H
Header (Byte4) Pattern Register	18H
Header (Byte4) Control Register	28H
Header (Byte5) Pattern Register	19H
Header (Byte5) Control Register	29H
ID External ECC Byte Count Register	2BH
ID Postamble Pattern Register	3CH
ID Postamble Byte Count Register	2CH
Data Preamble Pattern Register	3DH
Data Preamble Byte Count Register	2DH
Data Address Mark Pattern Register	3EH
Data Address Mark Byte Count Register	2EH
Data Synch Pattern Register	3FH
Data Synch Byte Count Register	2FH
Data Format Pattern Register	3BH
Sector Byte Count (L) Register	38H
Sector Byte Count (H) Register	39H
Data External ECC Byte Count Register	2AH
Data Postamble Pattern Register	30H
Data Postamble Byte Count Register	20H
Gap Pattern Register	3AH
Gap Byte Count Register	34H

FIGURE 5.9. Format Registers and Counters

During an operation these registers must not be read, as this will interfere with the DDC's internal access to these registers. This could cause internal PLA's to misinterpret these registers, and lead to sporadic misbehavior of the DDC. These registers may be written to any time. If written to during an operation they will take effect immediately.

One data byte pattern register is provided. This pattern is used during a format operation as the data field byte. It is repeated for the length of the data field in the sector.

BYTE COUNT REGISTERS
(Address = 20–23H, 2A–2FH)

The Byte Count registers determine the number of times each field's pattern can be repeated. All of ID and Data Preamble, Address Mark, Synch, Postamble, External ECC, and Gap patterns can be repeated for maximum 31 times. The Gap pattern, on the other hand, can be repeated for 255 times. As mentioned earlier, there can only be six Header bytes and 64K data bytes in any format for the selected

drive. The length of Header and Data bytes is controlled by the Header Control and Sector Byte Count registers, respectively (described below).

During an operation these registers must not be read, as this will interfere with the DDC's internal access to these registers. This could cause internal PLA's to misinterpret these registers, and lead to sporadic misbehavior of the DDC. These registers may be written to any time. If written to during an operation they will take effect immediately.

HEADER BYTE CONTROL REGISTER
(Address = 24–29H)

These six read/write registers control the associated six header bytes. Each of the six registers is 4-bits long and performs the same functions. One of these is shown in *Figure 5.10* and the functional description of each bit is given in the following.

D0	HBA	HEADER BYTE ACTIVE	0 = Header Byte Disabled 1 = Header Byte Enabled
D1	SSC	SUBSTITUTE SECTOR NUMBER	0 = Header Byte Used 1 = Sector Counter Used
D2	EHF	ENABLE HFASM FUNCTION	0 = Header Used Normally 1 = Header Interpreted as Sector Number
D3	NCP	NOT COMPARE	0 = Header Used for Compare 1 = Header Disabled

FIGURE 5.10. Header Byte Control Register (One of Six)

Header Byte Active (HBA)

This bit determines whether the corresponding Header byte is to be included in the Header field or not. If set low, the corresponding header byte will be omitted from the header field and setting it high will include the corresponding byte in the header field. Only 4 out of 6 header bytes can be disabled. Also, only two consecutive header bytes can be disabled.

Note: All the other bits in this register must also be set to zero if the header byte is to be disabled.

Substitute Sector Counter (SSC)

This bit when set high, enables the DDC to substitute the Sector Counter register's contents in the header byte pattern register instead of the actual header byte during a write operation, or to compare the Sector Counter register's contents with the corresonding header byte during a read operation. This is normally done in a multi-sector operation to enable automatically incrementing the sector number.

Enable Header Failed Although Sector Matched Function (EHF)

If bit EHF of any header control register is set high, then the associated header byte is designated as that byte that must match in order to enable generation of an HFASM. In this mode, if this header byte matches but any of the other header bytes don't, then an HFASM error and an interrupt is generated. In a Compare Header-Data operation, the header bytes are loaded into the FIFO and can be examined by the host by reading the FIFO Diagnostic Register (Address = 36H). This can also be used during a Compare Header-Read Data, but the FIFO will not store the header bytes, see Error Register description, HFASM.

When this bit is reset the corresponding header byte is compared normally.

Not Compare (NCP)

When this bit is set low, the Header byte will be written and compared normally. On the other hand, if this bit is set high, the corresponding Header byte will always be declared matched regardless of the actual comparison results. In other words the comparison is disabled. This can be used to read a group of sectors.

5.1.4 DMA Registers and Counters

The DMA registers and counters enable programming of the DMA start address (in single or dual channel mode), transfer length, and the various modes of operation. The Operation Command register controls actual starting of the Remote DMA operation. The DMA registers are listed in *Figure 5.11* and explained in the following paragraphs.

Name	Hex Address
DMA Sector Counter	37H
Local Transfer Register	36H
Remote Transfer Register	37H
Remote Data Byte Counter (L)	1AH
Remote Data Byte Counter (H)	1BH
DMA Address (Byte 0) Counter	1CH
DMA Address (Byte 1) Counter	1DH
DMA Address (Byte 2) Counter	1EH
DMA Address (Byte 3) Counter	1FH

FIGURE 5.11. The DMA Registers and Counters

DMA SECTOR COUNTER
(Address = 37H)

This read only register is used only when the DDC is configured in the dual channel tracking mode. This counter keeps track of the number of sectors transferred by the remote DMA channel (local RAM to/from system), and the local DMA channel (DDC to/from local RAM). When this register is 0, remote channel transfers are inhibited. This counter ensures that the source channel will not overtake the remote channel. This eliminates the chances of overwriting while transferring data to or from the local memory, or transferring non-data.

LOCAL TRANSFER (LT) REGISTER
(Address = 36H)

This write-only register, shown in *Figure 5.12*, controls the data transfers between the DDC and buffer memory, using the local DMA channel or single channel mode. It is configured at the time of initialization and normally need not be written to again. The Local Transfer register is not affected by reset or abort operations.

Local DMA Enable (SLD)

This bit when set high, enables the local DMA channel, Tracking or Non-Tracking. If this bit is not set high, the on-chip DMA will not transfer data. This bit is used to enable control of starting/stopping a DMA operation. If it is permanently disabled, external DMA circuitry can be used.

Local Word Data Transfer (LWDT)

This bit determines the length of the data word to be transferred between the DDC and buffer memory. When set to 0, single 8-bit bytes are transferred each DMA cycle and the address increments by 1. If this bit is set, 16-bit words are transferred each DMA cycle and the address increments by 2.

Reverse Byte Ordering (RBO)

This bit controls the order of bytes in a 16-bit word transfer. When RBO is set low, the first byte to be read from the disk will be placed in the least significant half of the word (AD0–AD7) and when RBO is set high, the LS byte will be mapped to AD8–AD15. This is only valid for data entering the FIFO. This byte should be reset for 8 bit transfers.

Local Slow/Fast Read and Write (LSRW)

This bit can add one wait state cycle to the DMA transfers. When this bit is reset the Read and Write cycle is 4 clock periods, but if this bit is set, one wait state is added and the read or write cycle is 5 clock periods. This extends the RD and WR strobes by one bus clock period, and allows the DDC to access slower memories.

Long Address (LA)

This bit determines the Local DMA address bus width and is only valid if Local DMA is enabled and the Remote channel is disabled. When Long Address is low, 16 address bits are issued and strobed by ADS0 pin. When Long Address is high, 32 address bits are issued, the lower 16 bits are strobed by ADS0 pin, the upper 16 address bits are strobed by ADS1/RRQ pin. The most significant 16 address bits are only issued when a rollover from the least significant 16 address lines occurs or on the first DMA cycle of a multi-byte transfer. When the most significant 16 address bits are issued, that DMA cycle is 5 clock periods long if no internal or external wait states are used.

Dump FIFO/Exact Burst (LTEB)

This bit controls how the data is burst to/from the FIFO. If this bit is reset the FIFO will fill or empty completely. When the disk is being read, the FIFO will wait until the FIFO is filled to the programmed threshold. At this time the DDC will completely empty the FIFO to buffer RAM even if more data entered the FIFO during the burst. During a disk write, when the FIFO is emptied to the programmed threshold, the DMA will fill the FIFO.

When this bit is set, the DMA will only transfer a fixed number of bytes to/from the system. For reading the disk, when the FIFO fills to the programmed threshold, the DMA will burst the exact number of bytes that are in the FIFO. Any bytes entering the FIFO after the burst begins will be transferred at the next burst. For a disk write, when the FIFO empties to the selected threshold, an exact number of bytes will be DMAed to the FIFO, whether the FIFO has emptied more or not.

Local Burst Length Select (LBL1, LBL2)

Bits LBL1 and LBL2 offer different burst lengths and the thresholds according to when data will be transferred to or from the FIFO. These burst lengths/thresholds could be 2, 8, 16 or 24 bytes, if these bits are programmed, 00, 01, 10, or 11, respectively.

REMOTE TRANSFER REGISTER (Address = 37H)

This write-only register, shown in *Figure 5.13*, determines the mode of transfers between the local buffer and the system I/O port (remote DMA channel) if the DDC is in dual channel mode. The register should be loaded at initialization and normally need not be written to again. It is not affected by reset or abort.

Remote DMA Enable (SRD)

This bit, when set high, configures the DDC in dual channel DMA mode, and enables the remote DMA channel.

Remote Word Data Transfer (RWDT)

The data bus may be configured to be either 8 or 16 bits during remote transfers between buffer memory and main system. If RWDT is high, transfers are 16 bits wide, and the remote address information is incremented by 2 each memory cycle and bit A0 remains low. If RWDT is low transfers are 8 bits wide, and the remote address increments by 1.

Enable External Wait (EEW)

If EEW is high, the EXTERNAL STATUS pin of the DDC will be enabled to supply wait states in both the local and remote DMA bus cycles. A side effect using this feature is that external synchronization and external ECC cannot be used. When EEW is low, no external wait states can be inserted, and external synchronization and external ECC may be used.

Remote Slow Read/Write (RSRW)

This bit allows for slower memory or slower system cycle time and is only valid if the Remote DMA Enable bit is high. In this case, if RSRW is set, each remote DMA cycle becomes five clock periods rather than four, and both the $\overline{RD}$ and $\overline{WR}$ strobes are widened by one clock period.

D0	SLD	SELECT LOCAL DMA	0 = Disabled 1 = Enabled
D1	LWDT	LOCAL WORD DATA TRANSFER	0 = 8-Bit 1 = 16-Bit
D2	RBO	REVERSE BYTE ORDER	0 = AD0–7 = LSB, AD8–15 = MSB 1 = AD0–7 = MSB, AD8–15 = MSB
D3	LSRW	LOCAL SLOW READ AND WRITE	0 = 4 Clock Transfer 1 = 5 Clock Transfer
D4	LA	LONG ADDRESS	0 = 16-Bit 1 = 32-Bit
D5	LTEB	LOCAL TRANSFER EXACT BURST	0 = Transfer Until FIFO Empty 1 = Transfer Exact Burst
D6 D7	LBL1 LBL2	LOCAL BURST LENGTH (FIFO THRESHOLD)	00 = 1 Word/2 Bytes 01 = 4 Words/8 Bytes 10 = 8 Words/16 Bytes 11 = 12 Words/24 Bytes

FIGURE 5.12. Local Transfer Control Register

Tracking Mode (TM)

This bit configures the DDC in Tracking or Non-Tracking DMA modes when it is set high or low, respectively. In Non-Tracking mode, the DMA channels are independent and addresses are allowed to overlap, while in Tracking mode, channel addresses are maintained at least one sector apart.

Remote Transfer Exact Bursts (RTEB)

When the DDC is performing a remote transfer, the condition of this bit determines when RRQ is de-asserted after the exact number of words or bytes, specified by RBL1 and RBL2, have been transferred. If RTEB is low, RRQ wil remain asserted until the whole count, specified by the Remote Data Byte Counter, has been transferred.

Remote Burst Length (RBL1, RBL2)

These bits select the burst transfer lengths during a remote transfer operation between buffer memory and a system I/O port if RTEB is set high. These length could be 2-byte (1-word), 8-byte (4-word), 16-byte (8-word) and 32-byte (16-word).

D0	SRD DMA	SELECT REMOTE	0 = Disabled 1 = Enabled
D1	RWDT	REMOTE WORD DATA TRANSFER	0 = 8-Bit 1 = 16-Bit
D2	EEW	ENABLE EXTERNAL WAIT STATE	0 = Disabled 1 = Enabled
D3	RSRW	REMOTE SLOW READ AND WRITE	0 = 4 Clock Transfer 1 = 5 Clock Transfer
D4	TM	TRACKING MODE	0 = Non-Tracking 1 = Tracking
D5	RTEB	REMOTE TRANSFER EXACT BURST	0 = Transfer Whole Count 1 = Transfer Exact Burst
D6 D7	RBL1 RBL2	REMOTE BURST LENGTH	00 = 1 Word/2 Bytes 01 = 4 Words/8 Bytes 10 = 8 Words/16 Bytes 11 = 12 Words/24 Bytes

FIGURE 5.13. Remote Transfer Control Register

REMOTE DATA BYTE COUNT REGISTERS

(Address = 1AH (LSB), 1BH(MSB))

These registers determine the byte count required in a remote data transfer. They are preloaded with a maximum count equal to the desired total DMA transfer, and are decremented by the DDC after each transfer until a count of zero is reached. This counter is 16 bits wide, therefore up to a total of 65,536 bytes can be transferred. Presetting this register to all zeroes transfers 65,536 bytes. The count can be read at any time during the transfer.

This register is also used in tracking mode DMA to keep track of whether a sector is ready to be transferred by the remote channel.

DMA ADDRESS (BYTE0–3) REGISTERS

(Address = 1CH–1FH)

The DMA address registers issue dual channel local and remote addresses during the local and remote transfers or single channel addresses. When using the dual channel DMA mode, registers 0 (LSB) and 1 (MSB) are used for holding and incrementing the local DMA channel address. Registers 2 (LSB) and 3 (MSB) contain the address information for remote DMA transfers. These registers can be preset to any address (within 64k) and can be read at any time.

In single channel DMA mode, the 4 registers are concatenated to form a 32-bit address, thus the single DMA channel can address up to 4 Gigabytes.

5.2 DDC COMMANDS

The DDC can be configured to perform various disk and related operations. These include disk read and write operations, error correction operation, formatting operations, and DMA operations. DMA and error correction commands are considered separately later. To understand the operation better, it is useful to break up the DDC's execution of a command into three phases:

1. Command Entry — μP loads bytes and command word prior to execution.
2. Command Execution — Once loaded the DDC performs the operation.
3. Result — After execution is terminated the μP reads the DDC to determine whether an error terminated the operation.

Since the DDC is primarily reading and writing to the disk drive, it has a rich set of disk read and write functions and each of these can be used is several modes. This creates a large array of commands that provides versatility. However there are about 10–15 commands/modes that would normally be used. This section will present the basic command operations first, then common commands are shown as combinations of the operations and finally a simplifying list is created.

A typical read/write command is composed of two operations; ID field operations and disk data field operations. Bits DO2, DO1, HO2, HO1; FMT in the Drive Command Register, determine the command to be executed. The two least significant bits enable some specific options to the commands. A list of all the possible commands is given in *Figure 5.13*. These commands are executed by setting up all the registers with the desired modes and header information, and as the last step the Drive Command register is loaded with the command. Once loaded the DDC will execute the command.

The header and data operations are described individually below. They repeat some of the information given in the register bit description in Chapter 4. Following the header and data operations, the more useful combinations of these are described.

5.2.1 Header Operations

Ignore Header

The DDC will use the preamble for PLO locking, and will check for the ID field's byte synch fields. The DDC will ignore the actual header contents, and treat any values as a match, and a data field operation will proceed on the subsequent data field.

DO2	DO1	HO2	HO1	FMT	
0	0	0	0	0	No Operation, No Format (No Operation)
0	1	0	0	0	Ignore Header, Check Data, No Format
1	0	0	0	0	Ignore Header, Write Data, No Format
1	1	0	0	0	Ignore Header, Read Data, No Format (Recover Data)
0	1	0	1	0	Compare Header, Check Data, No Format
1	0	0	1	0	Compare Header, Write Data, No Format (Normal Write)
1	1	0	1	0	Compare Header, Read Data, No Format (Normal Read)
0	1	1	0	0	Write Header, Check Data, No Format
1	0	1	0	0	Write Header, Write Data, No Format
1	0	1	0	1	Write Header, Write Data, Format (Normal Format)
0	1	1	1	0	Read Header, Check Data, No Format (Get Header Info)
1	1	1	1	0	Read Header, Read Data, No Format
0	0	0	1	X	Compare Header, No Data Operation *** ILLEGAL COMMAND ***
0	0	1	0	X	Write Header, No Data Operation *** ILLEGAL COMMAND ***
1	1	1	0	X	Write Header, Read Data *** ILLEGAL COMMAND ***
0	0	1	1	X	Read Header, No Data Operation *** ILLEGAL COMMAND ***
1	0	1	1	X	Read Header, Write Data *** ILLEGAL COMMAND***

FIGURE 5.14. The DDC Commands

Compare Header

This operation usually preceeds a normal disk read or write in which a particular sector is operated on. After preamble and synch fields are compared, the DDC compares every byte in the header to the pattern registers. Only if a complete match of the header bytes and no CRC/ECC error occurs, then the data field operation is executed.

Write Header

The DDC will write header information typically when formatting the disk, but operations allow the DDC to write individual headers in a hard sectored disk as a method of correcting a header. In this operation the entire ID field is written, preamble, both byte synch fields, header bytes, and CRC/ECC bytes.

Read Header

This is used when the host desires to know what a header is, usually when trying to determine where a disk drive head is located. This operation will compare the synch fields and then read the header bytes into the FIFO. CRC/ECC is checked.

5.2.2 General Data Operations

No Data Operation

This bit combination is valid only when the header operation is ignore header. Using this with other header operations will cause ''unpredictable'' results. Using this with the Ignore Header function is a NOP command and can be used when the user wishes to change non-command bits in the command register without executing a disk command.

Check Data

This is essentially a data NOP. If executed the DDC will read the data field just like a read data operation, but no data will be transferred to the system. The data field CRC/ECC is checked at the end of the operation.

Read Data

To fetch data from the disk drive this operation is used. The DDC first checks the data synch field to byte align and then it reads the data from the disk drive and sends it to the FIFO for transfer to external memory. After all bytes have been transferred, the data field's CRC/ECC bytes are checked.

Write Data

This is the inverse of the read data command. The DDC will begin by writing the data preamble field and synch fields. Data is input from the external memory into the FIFO, and this data is written to the disk. During the writing of data, CRC/ECC is being generated, and is appended to the data field.

5.2.3 When a Command Starts

As a command is loaded, the microprocessor can decide when the DDC should start the operation. There are two choices, which are programmed by loading bit SAIS in the Drive Command Register, Start at Index or Sector.

If this bit is set, the operation will be started at the beginning of the next sector (if hard sector) or immediately (if soft sectored). This is normally used for a normal read or write operation. Since the operation starts asynchronously to where the head is located over the track, starting immediately means that the sector will be read/written within one disk revolution worst case. If the command is started on an index pulse, it is likely that the head will pass the sector before it gets to the index to start the command, wasting a full revolution.

If this bit is reset when a command is loaded, the DDC will wait for the disk to revolve until the index pulse is seen. This mode would usually be used for track oriented operations. For example, a format and/or multi-sector operation would be most useful if started at the beginning of a track rather than the middle.

It is possible to execute any valid command starting either at the next sector or at the index pulse, thus many specialized command combinations are possible.

5.2.4 Multi-Sector Versus Single Sector Operations

All of the disk command op codes listed below have a multi-sector bit that determines whether the operation is a multi-sector or single sector operation. The DDC can be programmed to read one specific sector, several sectors, or an entire track in one operation. In a multi-sector operation the DDC will read a group of logically or physcially contiguous sectors. Logically contiguous means the DDC reads sectors with sector numbers that are in numerical order, but need not appear physically in order on the drive's track. For example, sectors are numbered physically on the drive as 5, 4, 6, 1, 3, 2, will be read 1, 2, 3, 4, 5, 6. This enables interleaving sectors. In order to do this the multi-sector bit must be set and several other registers and modes must be determined.

Usually a single sector operation can be considered the default operation, and additional steps must be taken to execute a multi-sector command. To review, a single sector operation will first execute a header operation followed by a data operation and then terminate. For a normal read or write, the header bytes are compared to the header pattern registers, and when the desired header is found the data field is operated on.

In a multi-sector operation, the DDC will re-execute the same command over again. The number of times the command is executed depends on the value programmed into the Number of Sector Operations Register (Address = 13H) with a maximum of 255. For most read or write commands, multi-sector operations will also need to select and program the Sector Counter (although some commands won't). For a given disk ID format one of the header bytes is the sector number. The header byte that has been designated the sector number must have its control register programmed to substitute the sector counter for the header pattern register. The sector counter is then programmed with the number of the first sector to be read/written.

After the command starts, the first sector is operated on. Then the Sector Counter increments and the Number of Sector Operations Counter decrements. If this has not reached zero the command is re-executed until the Number of Sector Operations Counter does reach zero. Programming a 1 into the Number of Operations Counter will cause the command to execute once.

The preceding describes how a multi-sector read or write would normally be executed. An example of when the Sector Counter may not need to be enabled might be a track dump command which is a read header-read data. Since sector numbers are not compared all the header fields the data fields can be sequentially read.

5.2.5 Interlock Mode Operation

The Interlock mode can be used to enable a microprocessor to update commands or parameters on the fly just after the previous command finishes with the header operation. This enables the μP to execute a series of commands on contiguous sectors. These commands may be the same one repeated, a format for example, or a different one executed sequentially. Interlock is enabled by setting the Interlock bit, and enabling the header interrupt in the Operation Command register, or polling the Next Disk Command status bit.

In the Interlock mode, and when a command is first issued, an interrupt at the end of the header operation informs the μP that the DDC is ready to accept a new command. The μP updates the header, or command registers, and lastly writes to the Header Byte Counter Register. This update must take place before the end of the CRC/ECC field of the present sector. If the Header Byte Counter is not written to before the header of the following sector, the DDC will set the Late Interlock bit in the Error Register, and abort the operation.

The Interlock mode can be used either in single sector or multi-sector operations. In a single sector operation, the μP updates all the header and control registers. It then writes the new command into the Disk Command register, and finally writes to the Header Byte Counter. This enables different commands to act on physically sequential sectors, i.e. read to one, write from the next.

In a multi-sector operation, the μP sets the Number of Sector Operations to the number of sectors to be read or written. The μP writes the command to the Disk Command register once at the beginning of an operation. Then the μP updates the header and mode registers after each interrupt. Finally, before the end of each data ECC field the Header Byte Counter is loaded. The DDC will automatically repeat the command until the Number of Sector Operations counter reaches zero. This command mode is useful for operating on physically sequential sectors that are not logically sequential. For example, formatting a track with interleaved sector numbers.

While executing in interlock mode, any pattern or count registers may be written to. However, it is recommended to not write to data pattern or count registers, as timing of the μP write relative actual disk data being operated on will determine whether the write will effect the present or next sector. During the operation, the pattern or count registers **must not** be read, as this will cause spurious operation.

Actually, the Interlock mode is somewhat misleading. For any command being executed, the DDC will be ready for the next command after it has successfully completed a header operation. The only action the Interlock mode takes is enabling an Error bit that tells the μP when it didn't update the DDC in time.

5.2.6 Command Termination, Resetting, and Re-enabling

Once a command starts execution, it will perform its desired task, or an error will be encountered that will prevent the command from executing. These errors could result from reading the disk, losing data while not transferring it fast enough, or not finding the header it is looking for after two index pulses have occurred. In these cases, the DDC will terminate the operation, set an error flag, and set itself into an error state from which it must be reset before the next command can be executed.

The errors that the DDC recognizes are listed in the Error register. Errors caused by corrupted disk data or format are:

Data Field Error
No Data Synch
Sector Not Found
Sector Overrun

Errors caused by not transferring data or parameters to the disk fast enough are:

FIFO Data Lost
Late Interlock

Errors that occur because the disk controller is looking for a sector header that is non-existent, or the disk head is on the wrong track are:

Sector Not Found
Header Failed Although Sector Matched

Additionally, during a correction cycle an error will be indicated if the correction failed and a Correction Failed error is set.

When an error occurs the DDC will terminate the command, and will issue an interrupt (if enabled). Once the error is flagged, the CPU must read the error register and then reset the DDC.

To reset the DDC, the CPU first must set the Reset bit in the Operation Command register. Then it must reset the Reset bit. This has reset the DDC into the default state. Now the DDC can be re-enabled by setting the Re-Enable bit high, and the DDC is then ready to receive the next command.

5.2.7 Summarizing Most Useful DDC Commands

As one can see there are a multitude of possible commands that the DDC can implement, and the header-data operations with the various modes tend to be very cryptic. To try to simplify the commands, *Figure 5.15* lists most of the common commands, a mnemonic, and the command op code. These commands assume the disk format is fairly standard, with the header at least containing one byte for the sector number. These are by no means all of the commands. Some specialized ones may be desirable, and can be assembled from the previous descriptions. Or, if the user decides to be creative with the header format other commands or modes may be useful, and maybe encryption/decryption of the header for data security could be implemented.

These commands are header-data operations with the multi-sector and start on sector or index bit configured to their most common way. For example, Read Sector (SRD) starts on a sector pulse (or immediately) and is not multi-sector. Multi-Sector Read Track is multi-sector and normally would start on the index pulse (but doesn't have to) so that the entire track's data can be read starting at the physical beginning of the track.

The read, write, format track commands assume that the Number of Sector Operations register is loaded with the number of sectors per track.

The logical multi-sector read, write commands assume that either the Sector Counter is enabled, or the Interlock mode is used.

Command Name		Op Code	
Read Single Sector	RDSS	11010010	D2H
Read Sector ID	RDID	01110010	72H
Read Multi-Sector Logical	RDMS	11010110	D6H
Read Track	RDTK	11010100	D4H*
Read Track Blind	RDTB	11000100	C4H*
Read ID Multi-Sector	RDIM	01110100	74H
Read Track Data/ID	RDDI	11110100	F4H*
Write Single Sector	WRSS	10010010	92H
Write Multi-Sector Logical	WRMS	10010110	96H
Write Track	WRTK	10000100	84H*
Format Track	FMTK	10101100	ACH
Format Track No Gap	FMNG	10100100	A4H
Find ID	FNID	01010010	52H
Find ID Multi-Sector	FNMS	01010110	54H
Recover Header	RCID	01100010	62H
Re-Enable Controller	RENB	00000001	01H
No Operation	NOP	00000000	00H

*__Note:__ For an entire track operation, the Number of Sector Operations Counter should be set to the number of sectors per track.

FIGURE 5.15. Common Configurations of the Command Bits

SINGLE SECTOR READ (Compare Header-Read Data)
Op Code = 11010010
This command is used to perform a normal disk read operation by disabling the multi-sector operation and starting immediately/sector pulse. The header bytes loaded into the DDC are compared to header information read off the disk drive. The DDC continues to scan the drive until a match is found. Once a header has matched, data in the subsequent data field is read from the disk and transferred to the system via the internal FIFO and DMA operations.

READ SECTOR ID (Read Header-Check Data)
Op Code = 01110010
This is a single sector command, which starts immediately. It will read the first sector header that the drive head passes over, and transfers it to the external memory. This is useful if the system gets lost and would like to know where the drive head is without recalibrating the drive to track zero.

READ MULTI-SECTOR LOGICAL
(Compare Header-Read Data)
Op Code = 11010110
This is a multi-sector command that starts immediately. There are two modes that can be used for this command. One is to use the sector counter to sequentially read logical

sectors. The second is to use the Interlock mode. If the sector counter is used then the logical sectors are read sequentially by sector number. In the Interlock mode the logical sectors can be read in any logical sequence, depending on μP update of header bytes.

This command can be modified to do a single sector read in multi-sector mode, by setting the Number of Operations to one and, preferrably, changing the Start bit from start on index to start immediately.

READ TRACK (Compare Header-Read Data)
Op Code = 11010100
The Read Track command has the same op code as the Multi-Sector Logical Read, except that the command is started on an index pulse. This will cause a read of all sectors ensuring that all other header bytes are compared and header CRC/ECC is checked. Of course the Number of Sector Operations counter must be set to the number of sectors per track. If the track is to be read in a logical order, then the sector number header byte can be compared to the Start Sector register. For a physically contiguous read the sector number header byte can be set to not compare.

READ TRACK BLIND (Ignore Header-Read Data)
Op Code = 11000010
This command will not compare the header field for a match, but will read the first data field that the DDC encounters and all subsequent data fields no matter what the header contains. The DDC will read in the data from the drive and DMA it to external memory.

READ ID MULTI-SECTOR (Read Header-Check Data)
Op Code = 01110100
This is a multi-sector command that starts at an index pulse and reads every header on the track (assuming the Number of Sector Operations equals the number of sectors on the track). This can tell the system what the entire track's header format.

READ TRACK ID AND DATA (Read Header-Read Data)
Op Code = 11110100
The read track command uses the Read Header-Read Data command in multi-sector mode, starting on the index pulse, and setting the Number of Sector Operations counter to the number of sectors on a track. This will read both the header and data fields of all sectors on a track. This can be used as a diagnostic tool to dump the entire contents of the disk drive's track.

WRITE SINGLE SECTOR (Compare Header-Write Data)
Op Code = 10010010
This command is used to perform a normal disk write operation to an individual sector. The multi-sector bit is reset and the command is started on a sector pulse or immediately. The header bytes are compared as in a disk read. Once the header matches, data is written to the associated data field by the DDC.

WRITE MULTI-SECTOR LOGICAL
(Compare Header-Write Data)
Op Code = 10010110
This is the same as the Multi-Sector Logical Read command. This one starts immediately, and can use the Interlock mode or Sector counter to change the header information. The sector counter enables logically sequential sector writing, and Interlock mode enables μP to update sectors on the fly.

As in the read command, by programming the Number of Sector operations counter to 1 results in a multi-sector mode single sector write. Also the start operation bit should be set to start on a sector pulse.

WRITE TRACK (Compare Header-Write Data)
Op Code = 10010100
The Track Write is similar to the Multi-Sector Logical Write command, but several modes are set up differently. First this command starts on an index pulse, and second the header byte corresponding to the sector number is programmed to not compare. This causes every sector to be written to in a physically sequential order.

FORMAT TRACK (Write Header-Write Data)
Op Code = 10101100
This command is commonly used when disk formatting is to be performed. The format bit in the disk command register normally should be set to execute this command. The entire ID Field and Data Field, one or more sectors, are written to the disk either from the header byte registers or from the FIFO, depending on which mode of disk formatting is selected (refer to chapter on Disk Formatting). The remaining header and data fields are written from the respective pattern registers.

FORMAT TRACK NO GAP (Write Header-Write Data)
Op Code = 10100100
There is a special format option which can be used in hard sectored drives by not setting the format bit in the Disk Command register and using a multi-sector operation. This enables writing of a format without intersector gaps. This may be useful if the drive puts servo information in the gaps. This would be overwritten with the normal format.

FIND HEADER/ID (Compare Header-Check Data)
Op Code = 01010010
This command is normally used to perform a diagnostic operation, operating almost like the Compare Header-Read Data command. It will first scan and compare the header information. The data field is read but not transferred.

FIND ID MULTI-SECTOR (Compare Header-Check Data)
Op Code = 01010100
This command is a multi-sector version of the Find Header, and can be used to verify all the headers on a track are valid.

RECOVER HEADER (Write Header-Check Data)
Op Code = 01100010
This command will start immediately, and rewrite the header field for the first sector encountered. This can be used as a method of recovering a damaged or unreadable ID field. To do so actually requires the interlock mode executing this command following a Find Header command. To recover the sector in error, the sector physically preceeding is found and the Find header is executed followed immediately by this command. The new header is then written over the damaged one.

RE-ENABLE (Re-enable bit set)
Op Code = 00000001
This command is used as part of the Reset/Re-Enable operation which is normally used to recover from an error condition. First the Reset bit in the Operation Command register is set, then reset. To finally enable the DDC for another command, the Re-Enable bit is set (see previous section).

NO OPERATION (Ignore Header-No Data Operation)
Op Code = 00000000
As mentioned previously, this is a NOP command. The DDC will not perform any operation.

ILLEGAL COMMANDS

Figure 5.14 lists several commands as illegal commands. They are not implemented by the internal PLA, and executing these commands will cause erroneous results.

CHAPTER 6 System and Disk Interfacing with DDC

6.0 INTRODUCTION

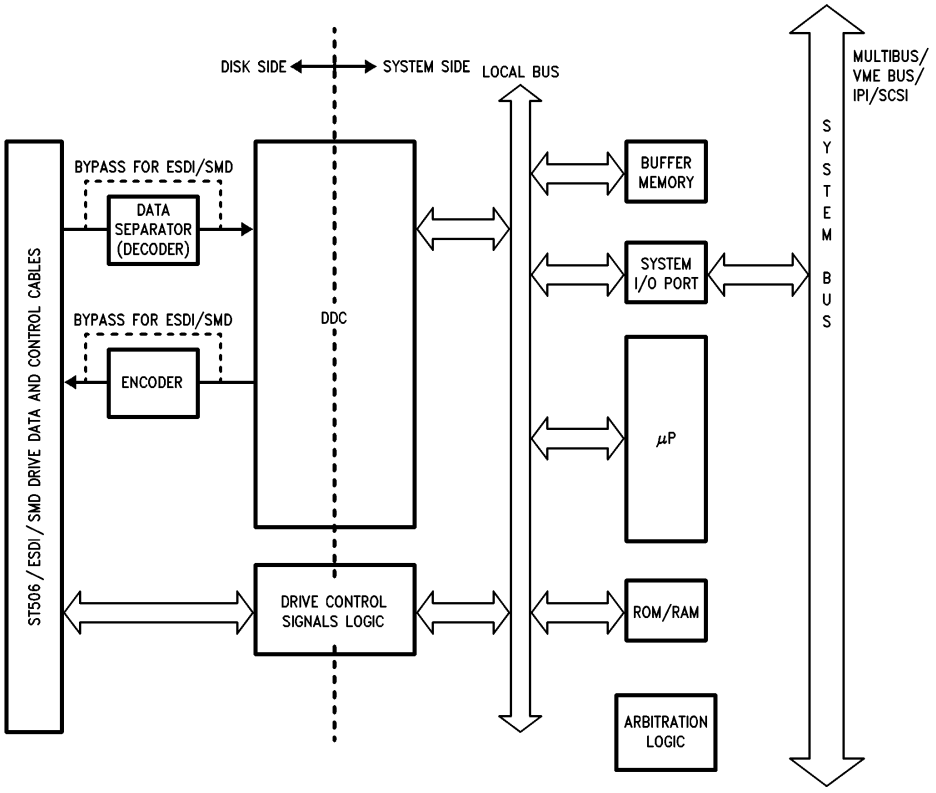
In a typical disk controller design, the DDC interfaces to the controlling microprocessor, memory and/or main system bus (such as MULTIBUS®, VME bus or IPI and SCSI) on the system side, and to the disk drives (via various disk interfaces such as ST506, ESDI and SMD) on the disk side. *Figure 6.1* shows the DDC in a typical disk controller design. As mentioned in previous chapters, the DDC (disk data controller) controls only the disk data path. It takes 8- or 16-bit wide data from the system bus, serializes it and then writes it to the disk in a disk write operation. While reading the disk data from the disk, the DDC deserializes the data and puts it back to the system bus. Before any disk operation can be performed, a track seek operation must be performed. The drive control signals required to perform a seek operation, can be generated using additional simple circuitry.

In this chapter, various DDC hardware interfaces to the host system and disk drive will be discussed in detail. DDC programming algorithms for carrying out disk operations are discussed in detail in chapter 7. The hardware interfacing is divided into two parts, system side and disk side.

6.1 SYSTEM SIDE INTERFACE

The DDC goes through three configuration modes when it performs a disk operation. When a microprocessor accesses the DDC to initialize it for a particular disk operation, it becomes a peripheral to the microprocessor (peripheral mode). While performing a disk operation and whenever the FIFO requires a data transfer to or from the external memory, the DDC becomes the bus master (master mode) and uses its on-chip DMA channels to perform the desired data transfers. If the desired data transfers are carried out by an external DMA controller, the DDC becomes a slave to the external DMA (slave mode controller).

In the following sub-sections the DDC's hardware interfacing with the microprocessor (peripheral mode), memory (master mode), and external DMA (slave mode) are discussed. The logic needed for arbitration of bus control between the DDC and microprocessor, and, to provide drive control signals are also discussed. In addition to the hardware connections, typical timing for various signals is also discussed.



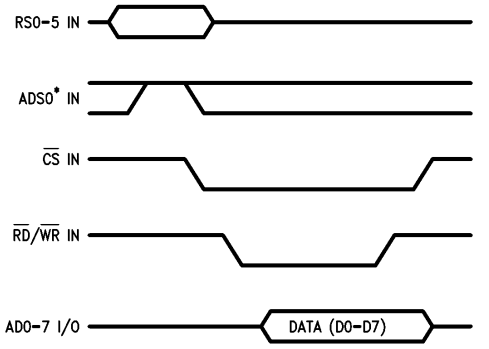
TL/F/8663–81

FIGURE 6.1. A Typical DDC-Based Disk Controller

6.1.1 Microprocessor-DDC Interface

The DDC must be initialized by a microprocessor (or microcontroller) in order to perform various disk operations. *Figure 6.2(a)* shows a basic microprocessor-DDC interface. When the microprocessor accesses the DDC, all 64 internal registers appear to it as unique memory or I/O locations. Each register can be randomly accessed and operated on. Only eight bits of data can be transferred to or from these registers using pins AD0–7. All the registers can be individually selected using six register select lines, RS0–5. Using these dedicated lines with an address strobe input, ADS0, the chip can be used in both multiplexed and demultiplexed address bus environments. Basically, the ADS0 and RS0–5 together act like a flow through latch.

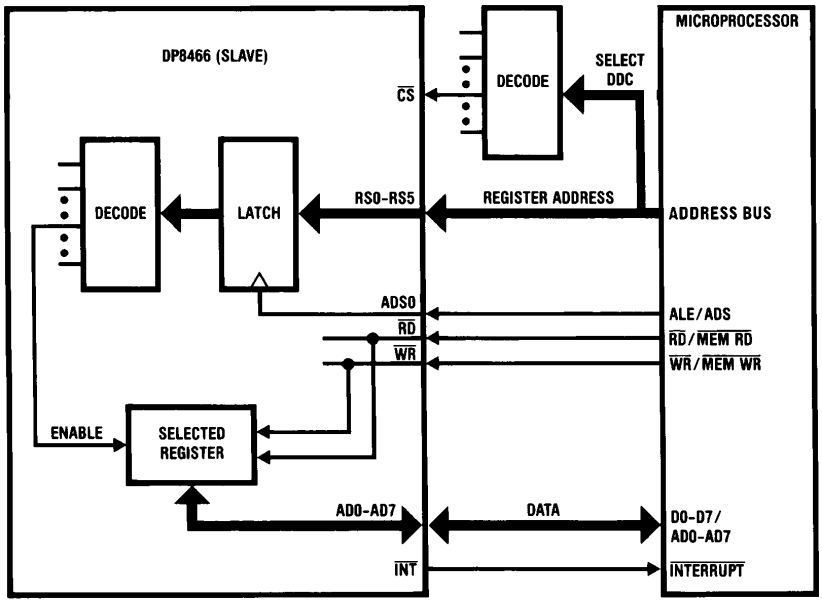
With multiplexed address and data lines, a positive strobe pulse on ADS0 will latch the address. The ADS0 line may be derived from a microprocessor address strobe line such as ALE. In systems with a dedicated address bus (demultiplexed), ADS0 may be pulled high to allow address information to flow through the latch. Finally, by applying $\overline{CS}$ and $\overline{RD}$ or $\overline{WR}$ strobes, the selected register is accessed. *Figure 6.2(b)* shows a typical timing for a multiplexed and demultiplexed system bus when the DDC is in Peripheral mode.



TL/F/8663–83

FIGURE 6.2(b). DDC Register Read/Write in Peripheral Mode

*Latched Register Select: ADS0 = Active
Non-Latched Register Select: ADS0 = Tied High



TL/F/8663–82

FIGURE 6.2(a). A Simplified Microprocessor Interface in Peripheral Mode

6.1.2 Bus Arbitration Logic

When the FIFO fills up (during a read operation) or empties (during a write operation) to the progammed threshold level, the DDC issues a local request (LRQ) to carry out local buffer memory transfers. Similarly, the DDC issues a remote request (RRQ) when local buffer memory needs a data transfer on the remote channel. Upon receiving the request (LRQ or RRQ), the current bus master should generate an acknowledge (LACK or RACK) to transfer bus control to the DDC. The request essentially puts the microprocessor (current bus master) on hold. The DDC keeps LRQ or RRQ asserted for the entire selected data burst transfer. This enables the DDC to remain bus master during this time while the microprocessor is on hold. See *Figure 6.3(a)* for typical RRQ-RACK and LRQ-LACK timing.

Generally, the LRQ-LACK or RRQ-RACK pins of the DDC are connected to the HOLD-HOLDA type (or BUSREQ-BUSACK type) of pins on the microprocessor through some additional combinational logic. This additional logic would be responsible for providing a smooth bus arbitration between the DDC and other possible bus users (like microprocessor). It must remove the possibility of any bus contention and may also prioritize DDC's DMA requests with other requests which may be present in the system. See *Figure 6.3(b)* for typical bus arbitration logic connections.

The Address strobes (ADS0 and ADS1) from the DDC and similar address strobe signals from the microprocessor may also be used in the bus arbitration logic, to generate a common set of address strobes for the system. This is useful if the demultiplexing address latches are to be shared between the DDC and the μP.

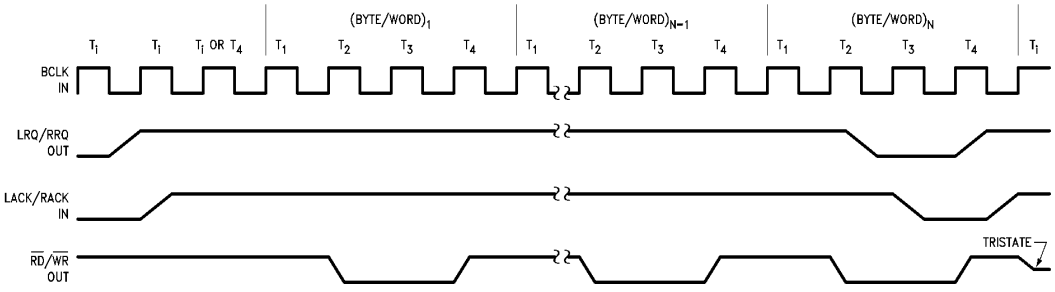
The DDC samples the RACK and LACK inputs during T_i(idle) or T_4 for each memory transfer. In general, the arbitration logic should leave LACK/RACK high for the duration of the burst. It may also toggle LACK/RACK so long as LACK/RACK are high during T_4 when the local channel has transferred one sector (in tracking mode-dual DMA), or when the remote DMA is enabled (in non-tracking mode).

6.1.3 The DDC-Memory Interface

After becoming bus master, the DDC communicates with the buffer or system memory if its on-chip DMA is chosen for data transfers. This communication takes place via a 16-bit multiplexed address/data bus, AD0–15, and supported by $\overline{RD}$, $\overline{WR}$, ADS0 and ADS1 strobes. The DDC is programmed in one of the three basic DMA modes; single channel, dual channel non-tracking and dual channel tracking. (See chapter 7 for detailed explanation on DMA programming). The hardware aspects of using the DDC in one of these modes are discussed below.

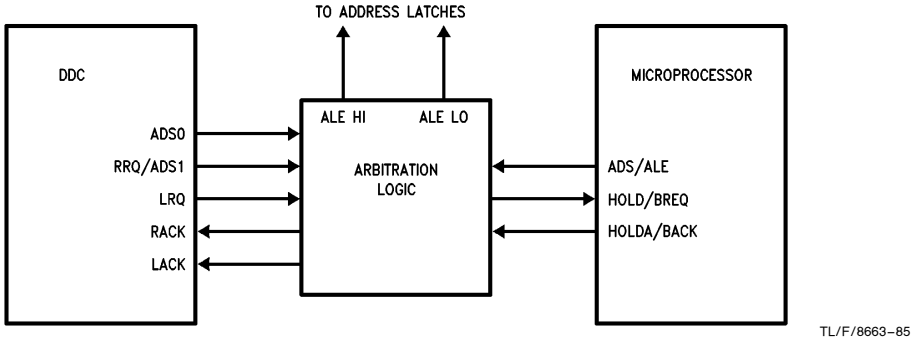
SINGLE CHANNEL DMA

In single channel DMA mode, the DDC takes care of data transfers between the FIFO and external memory using its local DMA channel. In this mode, the DDC can be set to generate either 16-bit or 32-bit of address. This gives system architects a great deal of flexibility. With 32-bit single channel DMA, up to 4 GBytes of memory can be accessed which is ideal in a system where the buffer memory is located within the main system memory. Factors like sector length, number of sectors per track and DMA burst length should be considered in determining the appropriate buffer memory size.



TL/F/8663–84

FIGURE 6.3(a). A Typical LRQ/RRQ-LACK/RACK Timing for a Data Burst Transfer



TL/F/8663–85

FIGURE 6.3(b). Arbitration Logic Connections

DUAL CHANNEL DMA

For systems where disk data is first buffered in a local memory before being transferred to the host memory via a host I/O port, the DDC offers two channels of DMA to handle the necessary data transfers. The ''local'' DMA channel controls data transfer between the FIFO and local buffer memory, while the ''remote'' DMA channel controls data transfer between the local buffer memory and the host I/O port. The two DMA channels can be operated independently of each other or the remote channel can be made to track the local channel. In either case, both channels can address a 64k address space. In the rare cases where a dual bus architecture and more than 64k bytes of local buffer memory are required, the DDC should be programmed for single channel, 32-bit addressing mode to handle FIFO-to-buffer transfers, leaving the buffer-to-host transfers to an external DMA controller.

For the purpose of the following discussion, it is helpful to introduce the concept of ''source'' and ''destination'' channels. In a disk read operation, the source channel is the local DMA channel, and the destination channel is the remote DMA channel. Conversely, in a disk write operation, the source channel is the remote DMA channel and the destination channel is the local DMA channel. In both cases, the source channel controls data transfers from the data source to buffer memory, while the destination channel controls data transfers from buffer memory to data destination. As an example, in a disk read operation, the source channel is the local DMA channel and controls data transfers from the FIFO to the buffer memory, the remote channel becomes the destination channel and controls transfers from buffer memory to the host.

Non-Tracking Dual Channel DMA Mode

In this mode, the local and remote channels operate as two independent DMA channels. This allows the microprocessor to tightly control data movement between the FIFO, buffer memory and the host I/O port. It also allows the microprocessor to process disk data in the buffer memory before transferring them to the destination, for example. This extra degree of control imposes an extra burden on the microprocessor. It is now responsible for preventing any destination channel transfers that may result in a data overrun situation. This can occur since the destination channel will continue to transfer data out of the buffer memory even though that section of memory has not been written to by the source channel.

Tracking Dual Channel DMA Mode

In this mode, the DDC keeps track of data transfers occurring over the two DMA channels. It forces the two DMA channels to track each other appropriately to prevent data overruns.

The tracking mechanism is implemented through a DMA sector counter, or DSC, which keeps track of the difference between the number of sectors transferred via the source and destination channels. The DSC is incremented every time a sector of data has been transferred into the buffer memory via the source channel, and decremented each time a sector of data has been transferred out of the buffer memory via the destination channel. The destination channel will not initiate the transfer of a new sector of data unless the DSC is non-zero. Thus the destination channel will not initiate the transfer of a new sector until it has been completely transferred into memory by the source channel. With appropriate choice of DMA data burst lengths, the local buffer memory then appears to the DDC as an extension of the internal FIFO.

DATA TRANSFER TIMING

Memory Read/Write Cycles

A standard DMA memory cycle consists of 4 BCLK periods, with the exception of the extended DMA memory cycle associated with the 32-bit addressing single channel DMA mode.

Referring to *Figure 6.4(a)*, a standard memory cycle consists of four bus states, T_1 to T_4, each lasting for one BCLK period. The cycle begins with the rising edge of BCLK in T_1, at which time the 16-bit memory address is put out on the address/data bus. ADS0 is also asserted during T_1 and can be used by the memory to strobe in the address on its negative edge. The low order address bits (A0–A7) are put out on port AD0–7, and the high order address bits (A8–A15) on port AD8–15. The address remains on these ports for the remainder of T_1. At the start of T_2 the address is removed and, depending on whether a read or write access is required, either the Read Strobe ($\overline{RD}$) or the Write Strobe ($\overline{WR}$) is asserted. These strobes stay asserted until the end of T_3. If a write access to buffer memory is required, the DDC will put its FIFO data on the address/data ports at the beginning of T_2 until the end of T_3. If a read access to buffer memory is required, then the DDC expects valid memory data to be placed on these ports during T_2 to T_3 so that the required data setup time referenced to the positive edge of $\overline{RD}$ is met. At the beginning of T_4, the appropriate acknowledge input (LACK/RACK) is sampled. If the sampling occurred when the acknowledge input is high, then the next memory cycle will be permitted to start at the end of T_4. Otherwise the DDC will relinquish control of the bus. If the acknowledge input is de-asserted prior to the rising edge of BCLK in T_4, the current memory cycle will be completed before the DDC frees up the bus.

The extended DMA memory cycle is used only in the 32-bit addressing single channel DMA mode. Since there are only 16 address/data lines available, the 32-bit addresses must be split into two groups of 16 bits and multiplexed onto the address/data ports. The low order 16-bit address is handled exactly as for the 16-bit addressing DMA modes. Additionally, a separate address strobe (ADS1) is provided so that the high order address bits can be stored in an external address latch.

As shown in *Figure 6.4(b)*, the extended DMA memory cycle is essentially the standard DMA memory cycle with an additional state T_0 inserted prior to T_1. At the start of T_0, the high order address bits are placed on the address/data ports for the duration of T_0. The address strobe ADS1 is also asserted and can be used by the memory system to latch in the high order address on its negative edge. Events occurring from T_1 to T_4 are exactly the same as in the case of the 16-bit addressing DMA modes.

Once initialized at the beginning of a DMA block transfer, the external high order address latch only needs to be updated whenever the lower order 16-bit address rolls over. Thus the extended DMA memory cycle is only required at the beginning of a block transfer, and each time the lower order address rolls over. Consequently, even when 32-bit addressing is required, the vast majority of DMA memory cycles will be the standard 4-clock cycle.

Prior to commencing any DMA transfers, the DDC must request and be granted control of the address/data bus. Whenever a DMA channel (local or remote) is ready for a transfer, the corresponding request output (LRQ or RRQ) is asserted to request control of the address/data bus. When such control is granted by the system via the appropriate acknowledge input (LACK or RACK), the DMA cycles will

commence. *Figures 6.5(a)* and *6.5(b)* illustrate the signal timings for a burst DMA transfer over the local and remote DMA channels.

Referring to *Figure 6.5(a),* a local transfer is requested by the DDC by asserting LRQ high when the FIFO threshold is reached. The DDC samples the LACK input at each positive edge of BCLK thereafter until a logic high is detected on the LACK input. At which point the DDC assumes control of the bus and starts a burst transfer. The burst spans over N memory cycles (where N is equal to the FIFO threshold in byte mode or half the FIFO threshold in word mode), or until the FIFO is full or empty. In the last transfer cycle of a burst, LRQ is de-asserted at the start of T_2 and the DDC relinquishes control of the bus at the end of T_4.

The remote transfer operation can be programmed to transfer data in bursts or to transfer the entire block in one stream. If burst mode is selected, the remote channel will transfer data in bursts with the user-programmed number of bytes per burst as illustrated in *Figure 6.5(b)*. As in the case of local transfers, the DDC first requests bus control by asserting remote request (RRQ) high. The remote acknowledge input (RACK) is then sampled at each positive edge of BCLK until it (RACK) goes high. The DDC then starts the first of a series of N transfer cycles, where N is the programmed number of bytes or words per burst. In the last cycle of the burst, RRQ is de-asserted at the beginning of T_2 and the DDC gives up control of the bus at the end of T_4 if the entire block has been transferred. Otherwise, RRQ is re-asserted at the start of T_4 to request bus control for the next burst. This allows other peripherals with equal or higher priority to gain access to the bus between bursts. The bus arbitration logic must de-assert RACK prior to T_4 to ensure that the DDC will not initiate another memory cycle. It should also hold off RACK to the DDC until such peripherals have completed their transfers.

If the remote channel is programmed to transfer an entire block in a continuous stream, then RRQ will not be de-asserted until T_2 of the last transfer cycle. If the system cannot tolerate such prolonged bus usage by the DDC, then the bus arbitration logic may de-assert RACK to force the DDC to relinquish bus control. If RACK is de-asserted prior to T_4, the DDC will complete the current transfer cycle and relinquish bus control. Otherwise the following transfer cycle will be completed before the DDC will free up the bus. When this technique is used to gain access to the bus, the arbitration logic must delay the granting of bus access to another device for time T_d from the de-assertion of RACK, where T_d is given by:

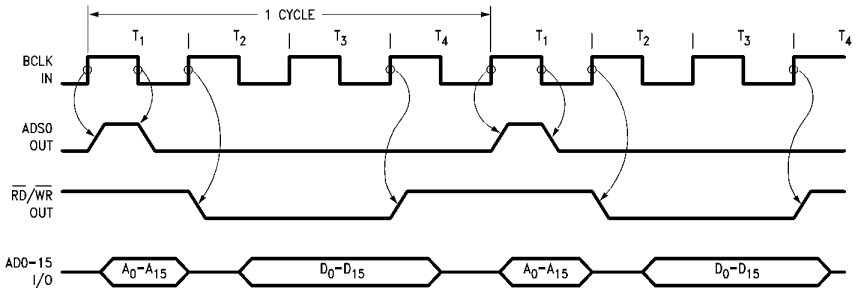
$$T_d = (\text{Period of BCLK}) + (\text{Duration of one DMA memory cycle}).$$

Note that the duration of a DMA memory cycle may vary depending on the number of wait states that may be inserted, as explained in a later section.

In both tracking and non-tracking modes, the local channel has priority over the remote channel for bus access.

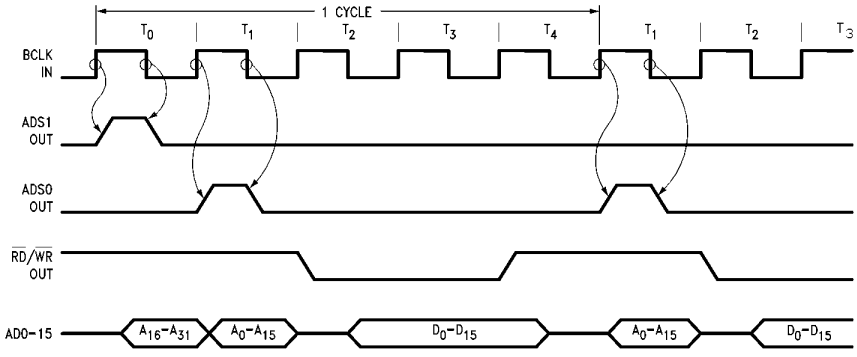
Bus Latency

The DDC can be operated at a 20 MHz Bus Clock (BCLK). With this frequency data can be transferred between the FIFO, local memory and system I/O port at the rate of 5 Mega-Transfers/sec (or 10 MBytes/sec) assuming a 4 clock periods memory cycle. With 15 Mbits/sec disk data rate, the 32-byte FIFO can be filled in approximately 17 μs.



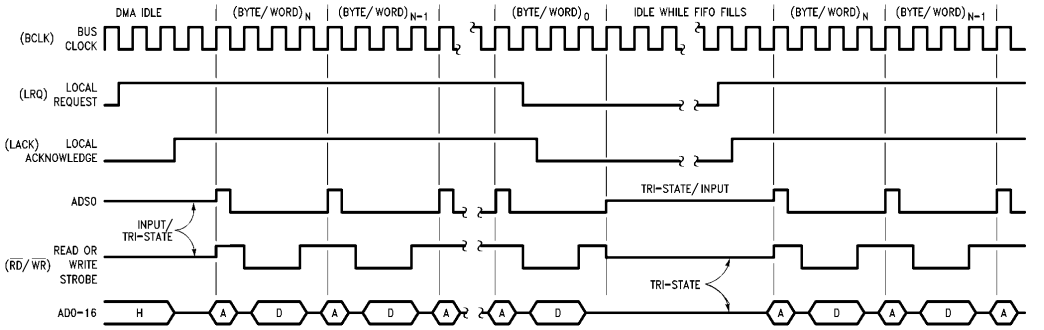
TL/F/8663–86

FIGURE 6.4(a). 4-Clock DMA Memory Cycle (DDC in 16-Bit Single and Dual Channel DMA Modes)



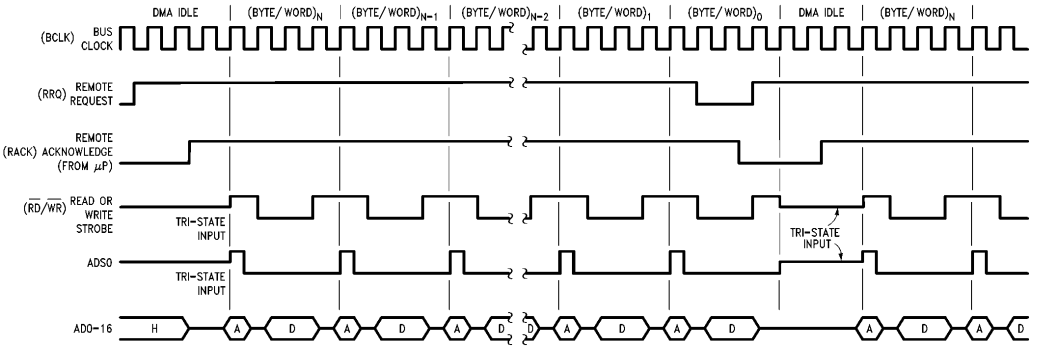
TL/F/8663–87

FIGURE 6.4(b). 5-Clock Extended DMA Memory Cycle (DDC in 32-Bit Single Channel DMA Mode)



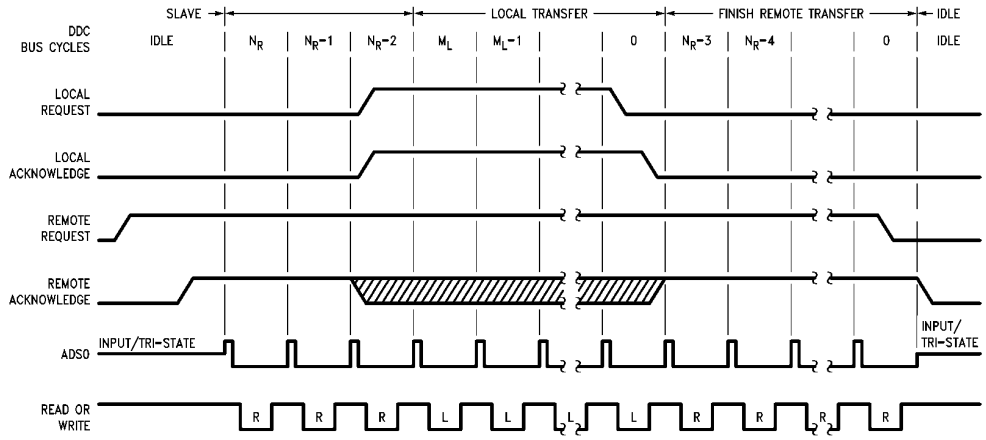
TL/F/8663–89

FIGURE 6.5 (a). Typical Local DMA Burst Transfer Timing
(H = Host, A = DDC Address, D = DDC Data)



TL/F/8663–88

FIGURE 6.5 (b). Typical Remote DMA Burst Transfer Timing
(H = Host, A = DDC Address, D = DDC Data)



TL/F/8663–90

FIGURE 6.5 (c). Interleaved Remote and Local Transfer Timing

Note N_R = Remote DMA Transfer Cycle Number N
M_L = Local DMA Transfer Cycle Number M

At 20 MHz bus clock (i.e. clock period of 50 ns and memory cycle of 200 ns) and assuming word-wide DMA transfers, the DDC will take 16 × 200 ns = 3.2 μs to empty the FIFO. This essentially shows that microprocessor bus is free for 17 − 3.2 = 13.8 μs (81.2% of the time). In other words, the DDC will need bus control after every 17 μs.

The selection of the FIFO threshold level should be based on the maximum amount of time the system takes to respond to a DMA request, often called bus latency. If the system has a longer latency, then a smaller threshold should be programmed. This will allow greater time for the system to respond without overflowing the FIFO. The disadvantage to programming lower FIFO thresholds is that more requests are made, tying up the system bus more often. For example, with an 8 byte threshold a request would be made about every 4.5 μs.

A second consideration is whether to have an exact burst, or burst until the FIFO empties. If the system has significant latencies then the FIFO should be emptied. If the system must not relinquish the bus for too long then a fixed burst size must be chosen.

The bus latency should be calculated for a given disk and DMA transfer rate to determine appropriate FIFO threshold. Programming the DMA for various burst transfer options is discussed in depth in chapter 7.

Local and Remote Interleave Timing

In dual channel DMA mode, the local and remote transfers can be interleaved using the DDC in tracking mode (see chapter 7 for details). A typical local and remote data transfer interleave timing is shown in *Figure 6.5(c)*. In this case a remote transfer is interrupted by the higher priority local transfer. Taking the same example given in the previous sub-section, the remote transfers can be carried out during the 81.2% of time when the local channel is not using the bus.

Slow Read/Write

The Read or Write strobes ($\overline{RD}$ or $\overline{WR}$) can be extended by adding extra wait state(s). It can be done internally by setting the LSRW bit in Local Transfer register (and) or the RSRW bit in Remote Transfer register. This will generate an extra cycle (T_W) between the T_2 and T_3 clock cycles as shown in *Figure 6.6(a)*. These strobes can also be extended by setting the EEW bit in the Remote Transfer register in conjunction with driving the EXT STAT input high. This will add extra wait states between clock periods T_3 and T_4, as shown in *Figure 6.6(b)*. The DDC samples the EXT STAT input at the positive edge of T_3 during each DMA (local or remote) memory cycle. If EXT STAT is sensed high and the EEW bit of the Remote Transfer Register is set, then a wait state (T_W) of a bus BCLK period will be inserted after T_3. During such wait states, the DDC continues to sample EXT STAT at the positive edges of BCLK. If EXT STAT is sampled high, a new wait state will be inserted at the end of the current one. If EXT STAT is sampled low, then the next state will be T_4.

6.1.4 DDC-External DMA Interface

If the on-chip DMA is not to be used, an external DMA controller must be used to service the FIFO. A typical DDC-external DMA interface is shown in *Figure 6.7(a)*. The LRQ asserted by the DDC is acknowledged by the external DMA and the DDC becomes a slave to the DMA controller. The DMA controller carries out the local transfers and deasserts LACK after LRQ is deasserted by the DDC. A typical FIFO Read/Write sequence by the external DMA is shown in *Figure 6.7(b)*.

6.1.5 Drive Control Signals Logic

The drive control signals generation is not incorporated on the DDC which makes it interfaceable with any type of disk

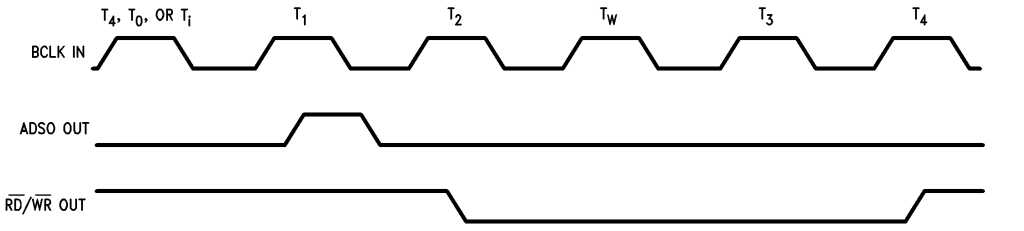


FIGURE 6.6 (a). DMA with Internal Wait States

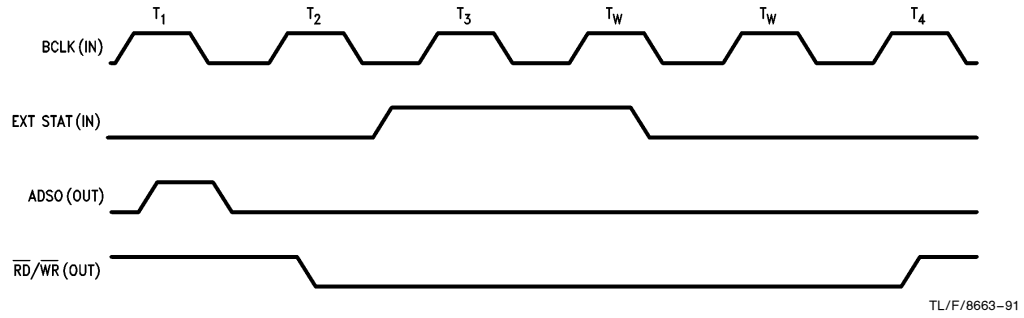


FIGURE 6.6 (b). DMA with External Wait States

interface. These signals can be generated simply with an I/O port associated with the controlling microprocessor. The microprocessor prepares the control signals for a particular drive interface and writes it to the I/O port. In high performance multi-drive, multi-interface systems, a dedicated microcontroller (COPS™, 8048 or HPC) or microprocessor (NSC800™, Series 32000® etc.) may be used to provide control signals for various drives. In this case, the controlling microprocessor (or host) instructs the microcontroller to perform a complete disk seek operation.

6.1.6 DDC in Typical System Configurations

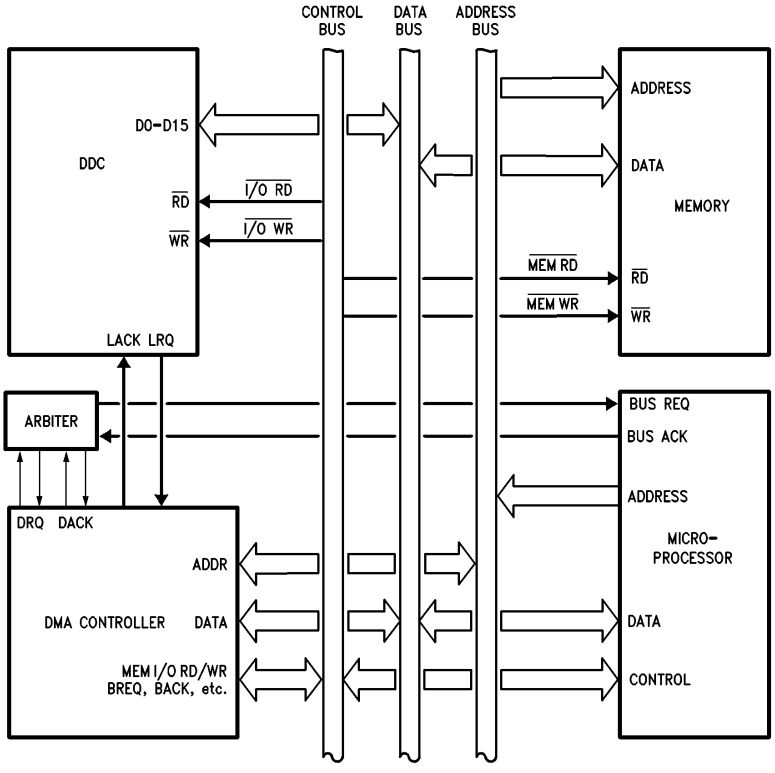
The DDC can typically be used in two types of system architectures, i.e. a single bus system or a dual bus system.

SINGLE BUS SYSTEM

Single bus systems usually are standalone systems controlled by a microprocessor. In such system, each component of the system communicates with the rest of the system through a single bus. For example, in a single board microcomputer, all the functional components like memory, I/O ports etc. communicate with the controlling microprocessor via main system bus.

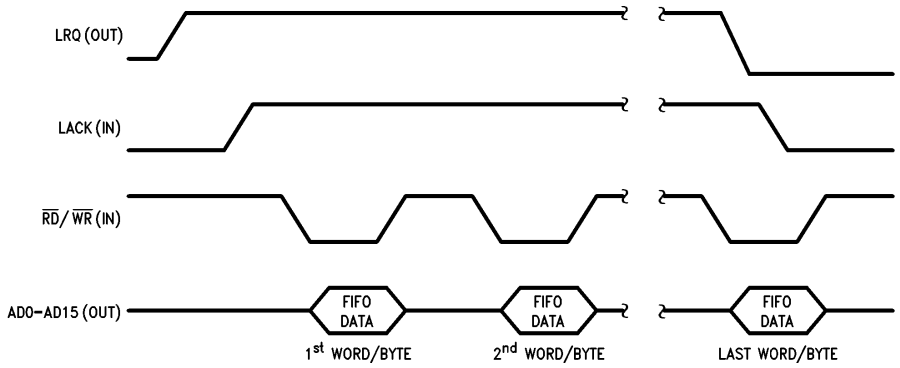
The DDC when used in such a system, becomes another component that communicates with the controlling microprocessor via the main system bus. It also communicates directly with the system memory using its on-chip DMA capability (usually the local channel). A single bus system generally requires a large system memory, so using the DDC in its single channel DMA mode becomes very worthwhile as it can access up to 4 GBytes of memory in this mode. The DDC in a typical signal bus system is shown in *Figure 6.8(a)*. The controlling microprocessor sets up the drive signal control logic to perform track seek operations and the DDC to perform disk operations.

In high performance single bus systems, it may be desirable to use a dedicated microcontroller to control the DDC and to generate the drive control signals instead of involving the main system microprocessor in such tasks. *Figure 6.9* shows the disk controller design in a 32-bit single bus system. Here an HPC or other single chip microcontroller is used to program the DDC for various disk operations and to generate drive control signals compatible with the desired interface. As mentioned above, the DDC is typically configured in single channel DMA mode when used in a single bus system environment. There is no restriction on using the DDC in dual channel DMA mode except total address capability. *Figure 6.10* shows a single bus system with the DDC in dual channel DMA mode, transferring data between the FIFO and 64k-pages of 16 MBytes system memory. External latches were added to extend the address range to 24 bits. To use them the host μP would load the most significant 8 bits with the 64k page to transfer data to/from. In this design the local and remote DMA channels are restricted to operating in the same page. To overcome this an additional latch and some logic gating the LACK and RACK signals could be used to enable operation of each channel in different pages.



TL/F/8663–93

FIGURE 6.7 (a). A Typical External DMA-DDC Interface



TL/F/8663–94

FIGURE 6.7 (b). DDC FIFO Read/Write by External DMA

DUAL BUS SYSTEM

Systems usually have two or more buses associated to them. One set of buses for local communication and another main system bus for communication with the host. These systems contain sub-systems (or modules) of the main system (the host). The purpose of the local bus architecture is to allow peripheral processors to control specific tasks, offloading the host CPU. Thus the local bus has in addition to peripherals, such as the DDC, a local microprocessor, memory and an interface to the main system. All the sub-systems access the main system through the I/O channels connected to the main system bus (such as MULTIBUS®, VME etc.).

The DDC easily fits into a dual bus system when configured in the dual channel DMA mode. The local DMA channel communicates with local memory and the remote DMA channel is used to transfer data between local memory and a system I/O port. The main system's DMA controller then takes the data to/from the I/O port from/to main memory. The DDC in a typical dual bus system is shown in *Figure 6.8(b)*. The local microprocessor receives commands from the host using DDC's remote channel and then sets up the drive control logic and the DDC for different disk operations. The arbitration block arbitrates the bus between the DDC (while in bus master mode) and the local microprocessor. *Figure 6.11* shows a dual bus system with local microprocessor using 48 kbytes of local memory space as ROM and RAM, and allowing only 16 kbytes of local buffer.

Figure 6.11 shows a possible local bus implementation. Here a local CPU such as the NSC800, 32008, 80188, 64180, etc. controls both the DDC and the disk control port. The local CPU is also in charge of receiving commands from the main system, caching disk sectors, and performing any logic to physical sector address translations.

In this design we have optionally segmented a special 16k for sector data buffers. This is not necessary, but does ensure the DDC will not access the local CPU's data memory inadvertently (this could be done in software as well). The HC646 and some read/write logic form the pass thru port for the commands/data from the main system. The DMA external controller, which may reside on the CPU (80188, 64180) or in the main system, controls transfers from main memory to/from the port. The DDC's remote DMA controls data transfers between local memory and the pass thru port.

INTELLIGENT PERIPHERAL BUS

A special case of the Dual Bus architecture are intelligent peripheral buses, such as SCSI and IPI. In this case the local bus is the intelligent disk drives bus, and the second bus is the SCSI or IPI bus which will connect to a main system through a host adapter. The dual channel DMA mode of the DDC is ideally suited to this application. The local DMA channel can manage transfers between the DDC's FIFO and the drives buffer RAM, while the remote DMA is passing information to/from the SCSI or IPI bus. The design of the system interface is very similar to *Figure 6.11*, except that the read/write interface logic block and the 'HC646 is replaced by a SCSI or IPI interface, and the systems remote DMA controller is located at the system end of the bus and not on the drive. With the DDC's two channel capability eliminates the need for an external DMA controller on the SCSI or IPI drive.

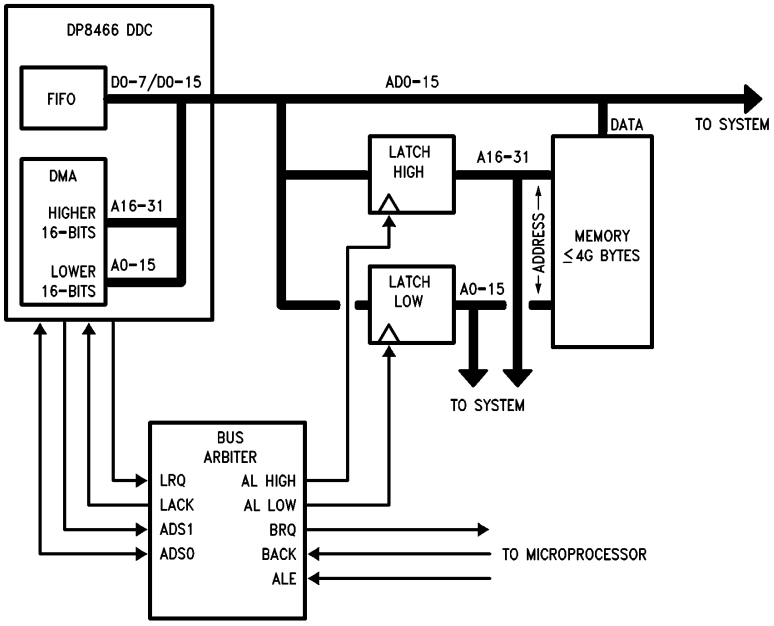


FIGURE 6.8(a). The DDC in a Single Bus System

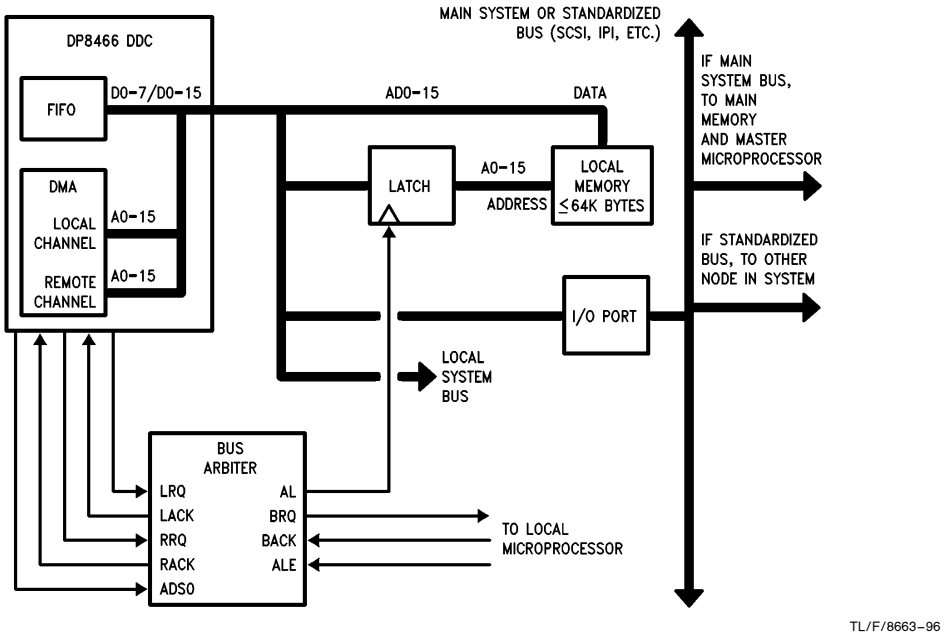
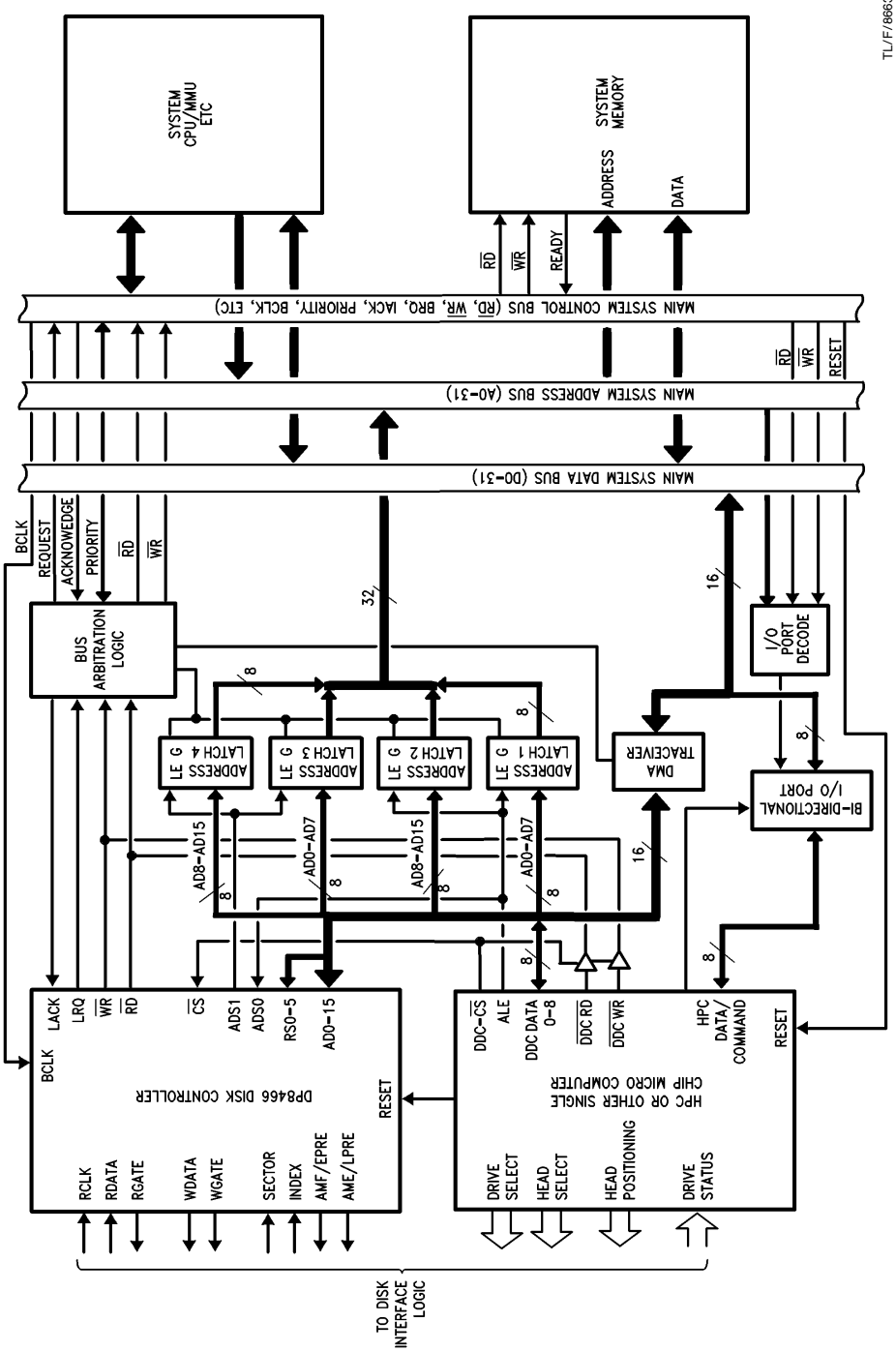


FIGURE 6.8(b). The DDC in a Dual Bus System



TL/F/8663–97

FIGURE 6.9. Conceptual Design of DP8466 Using Single Channel DMA and Controlled by a Microcontroller That Controls DDC and Drive Control Signals

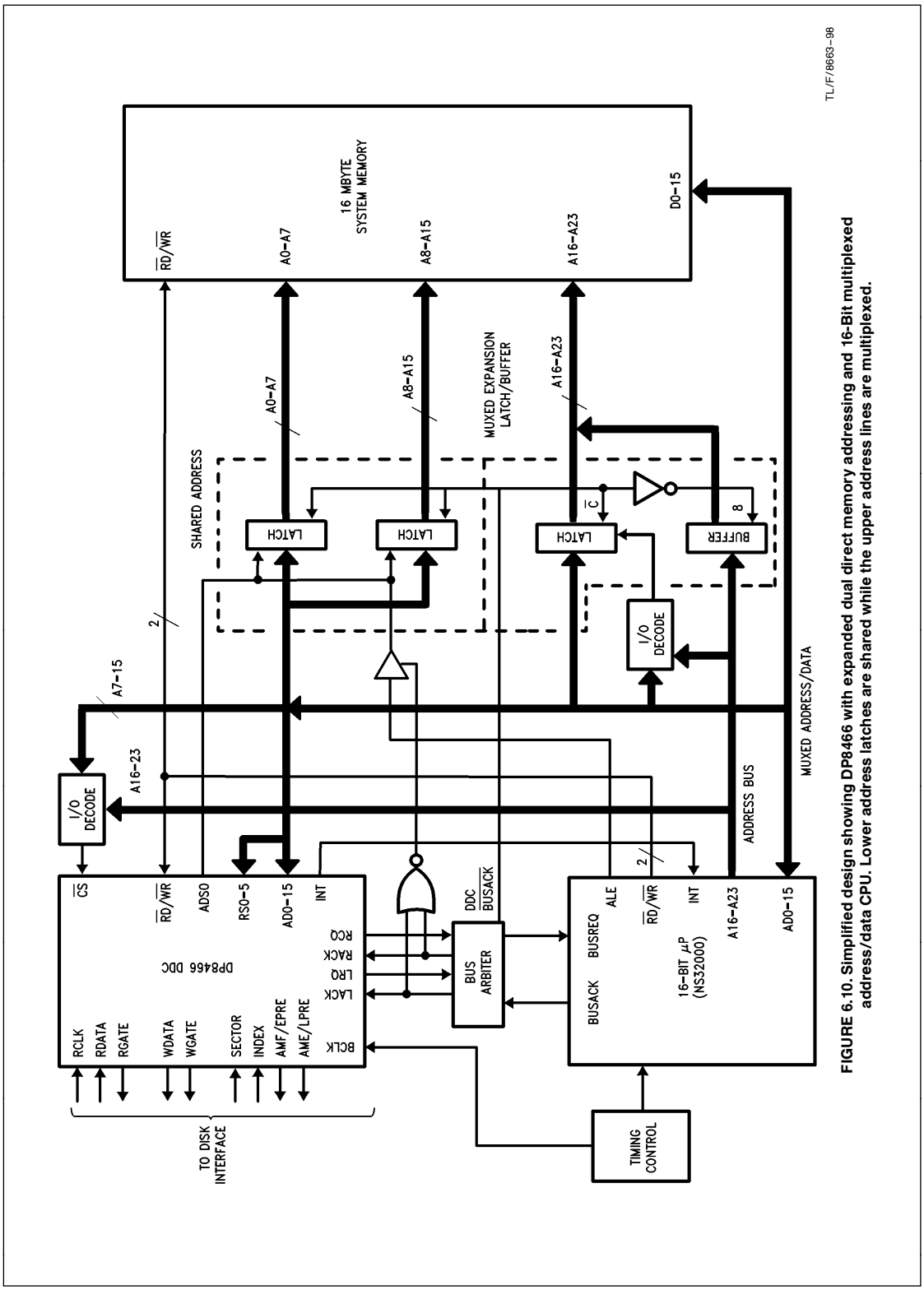


FIGURE 6.10. Simplified design showing DP8466 with expanded dual direct memory addressing and 16-Bit multiplexed address/data CPU. Lower address latches are shared while the upper address lines are multiplexed.

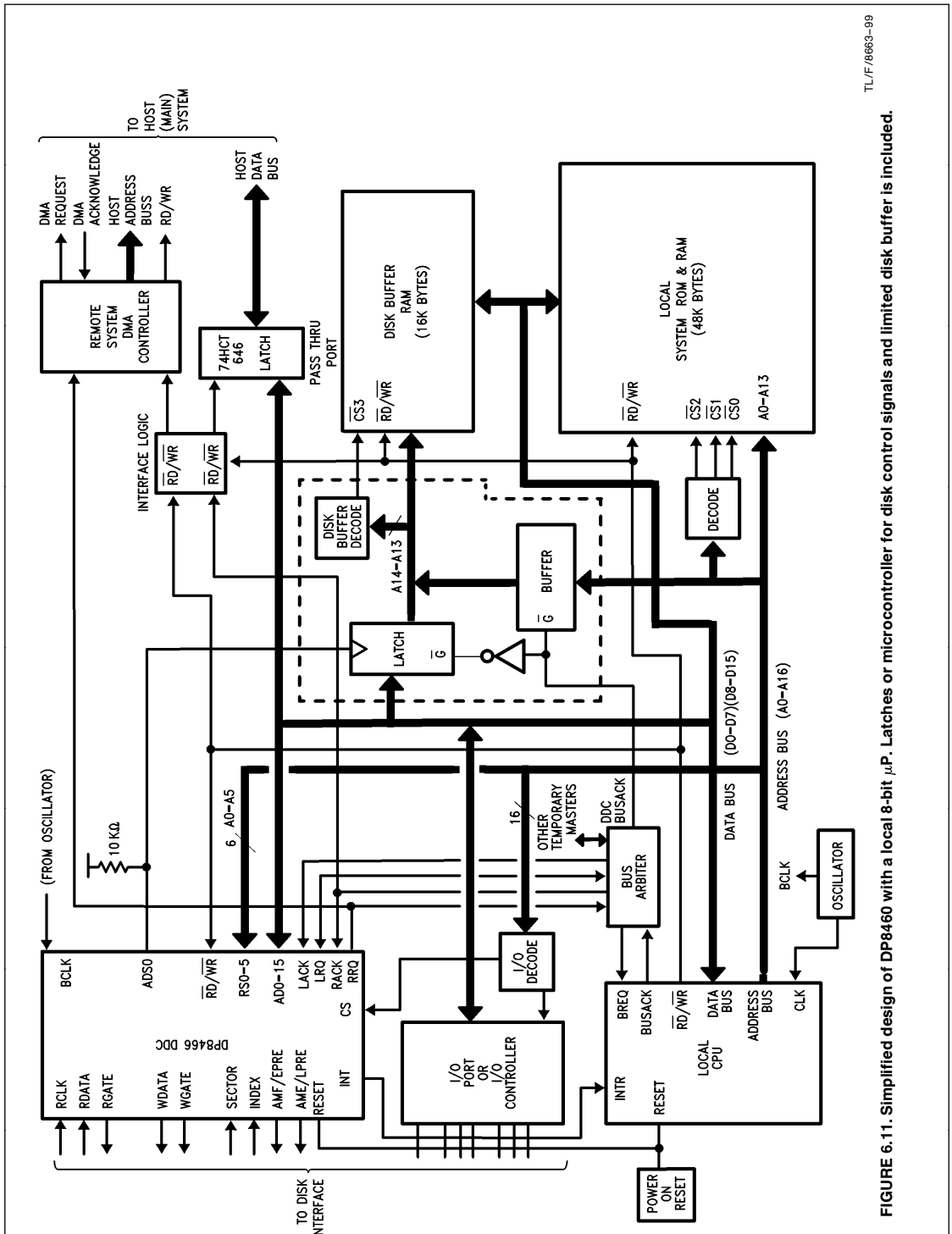


FIGURE 6.11. Simplified design of DP8460 with a local 8-bit μP. Latches or microcontroller for disk control signals and limited disk buffer is included.

6.2 DISK SIDE INTERFACE

The disk side hardware connections/orientation is a function of the disk interface standard used, the encoding/decoding scheme and the local intelligence used for disk control signals. The Disk System Controller essentially consists of a data separator (DP8465), the Disk Data Controller (DP8466) and some local microprocessor for disk control like the NSC800 or a microcontroller like the HPC. The Disk Pulse Detector (DP8464) is situated on the drive for all the interface standards, while the data separator is on the drive for the ESDI and SMD interfaces. *Figure 1.9(a)* in chapter 1 also shows the interface points for the various standards. If MFM encoding is used it can be programmed to be part of the DP8466, details can be found in chapter 7; whereas in case of 2,7 RLL code, the DP8463 (2,7 ENDEC) could be used in conjunction with the DP8462 (2,7 data synchronizer).

6.2.1 Generalized Disk Interface with DP846X Chip Set

The DP8464 pulse detector receives signals from the disk's read amplifier, and converts these signals to a digital pulse train. The DP8465 Data Separator receives digital pulses from a pulse detector circuit (such as the DP8464). After locking on to the frequency of these input pulses, the DP8465 separates them into synchronized data and clock signals. If the input pulses are MFM encoded data, the data is made available as decoded NRZ data to be deserialized directly by the disk data controller DP8466. If the RLL code is used, the synchronized data output is available to allow external circuitry to perform the data decoding function. All the digital input and output signals are TTL compatible. The raw MFM from the pulse detector in the drive is connected to the ENCODED DATA input of the DP8465. The DELAY DISABLE input determines whether attempting lock-on will begin immediately after READ GATE is set or after two bytes. Typically in a hard sectored drive, READ GATE is set active as the sector pulse appears, meaning a new sector is about to pass under the head. Normally the preamble pattern does not begin immediately, because gap bytes from the preceding sector usually extend just beyond the sector pulse. Allowing two bytes to pass after the sector pulse helps ensure that the PLL will begin locking on to the preamble, and will not be chasing non-symmetrical gap bits. Attempting to lock-on to a fixed preamble pattern speeds up lock-on, and after another two bytes the PLL will nominally have locked-on. Thus DELAY DISABLE should be set low for this kind of disk drive. For soft sectored drives, the controller normally will not wait for the index pulse before it attempts lock-on, so that READ GATE may go active at any time. Chances are the head will not be over a preamble field and therefore there is no need to wait two bytes before attempting lock-on. DELAY DISABLE can therefore be set high. If a non preamble field is passing by as READ GATE goes active. The DP8465 will not indicate lock, and so no data decoding will occur nor will MISSING CLOCK DETECTED go active. Normally, if lock-on has not been achieved after a certain time limit, the controller should de-activate READ GATE and then try again. For MFM encoded disk drives, the LOCK DETECTED output will be connected back to the SET PLL LOCK input. As the PLL achieves lock-on, the DP8465 will automatically switch to the slower tracking rate and decoded data will appear at the NRZ READ DATA output. Also the READ CLOCK output will switch from half the 2f clock frequency to the disk data rate frequency. If a delay is required before the changeover occurs, a time delay may be inserted between the two pins. The ZEROES/ONES PREAMBLE input selects which preamble the chip is to lock-on to. *Figure 6.12* gives schematics of the data separator in a disk system—when in the controller and when on the drive. For more specific details on the DP8465 refer chapter 3.

If the drive uses the RRL code such as ''2,7'', instead of MFM, the PLL function of the DP4862 may be used in conjunction with the 2,7 ENDEC (DP8463), as shown in *Figure 6.13*. The DP8463 performs encoding of NRZ data to RLL encoded data, and RLL encoded data back to NRZ data. It uses the SYNCHRONIZED DATA output of the DP8462 along with VCO CLOCK to lock-on to the preamble and then decode data. For more specific details on the DP8463 refer to it's data sheet.

The most important component in the disk side is the physical disk interface itself. We shall discuss the interface details for some major interface standards viz., STxxx, ESDI etc. Higher level interfaces, like SCSI, incorporate the whole controller board on the drive and interface with the host through a host adapter on to the system bus. However the DDC as such interfaces to the host microprocessor for commands and status.

6.2.2 Interfacing the DDC to the ST506/ST412HP Standard

The schematic in *Figure 6.14* shows the interfacing of the DDC to a ST506/ST412HP disk interface. The ST506/ST412HP interface standard has a control cable which is daisy chained to the drives (assuming a multiple drive system). On selecting the drive all the signals on the control cable are associated with the drive selected. The data cable is radial in nature and is multiplexed at the controller using the drive select output on the data cable. The ST506/ST412HP interface standard supports MFM encoding and soft-sectored formatting only. Since the DDC does not take care of the disk control signals, this is done using some local intelligence, indicated as the Disk Signals Controller block, DSC. The drivers are open collecter drivers as per the requirements of the interface standard. The DDC requires NRZ data, hence the MFM encoded data from the drive is sent to the data separator, which synchronizes the data and decodes it to NRZ. The missing clock detect output of the data separator is used to trigger the address mark found (AMF) input of the DDC. While writing data to the disk, the DDC provides MFM encoded data, and the necessary precompensation is provided using an external delay line in conjunction with the early and late precompensation outputs (EPRE/LPRE) of the DDC. The DDC and the DSC could interface to the system bus of the host system or could interface to the host system through an intelligent interface like SCSI or IPI, as mentioned in the previous section.

The basic operation of the interface is as follows. The DSC first selects the drive select lines. Then the head selected by the head select lines is positioned over the desired track by issuing step pulses in conjunction with the direction line. Once the head is positioned, indicated by the seek complete line, the DDC is ready to initiate a read/write operation. This interface's read path is through the data separator and hence a lot depends on the selection of the data separator. The data separator then feeds the controller. The write path consists of write data out of the DDC going to a precompensation block. The output of the precompensation block goes over the ST506 interface to the drive.

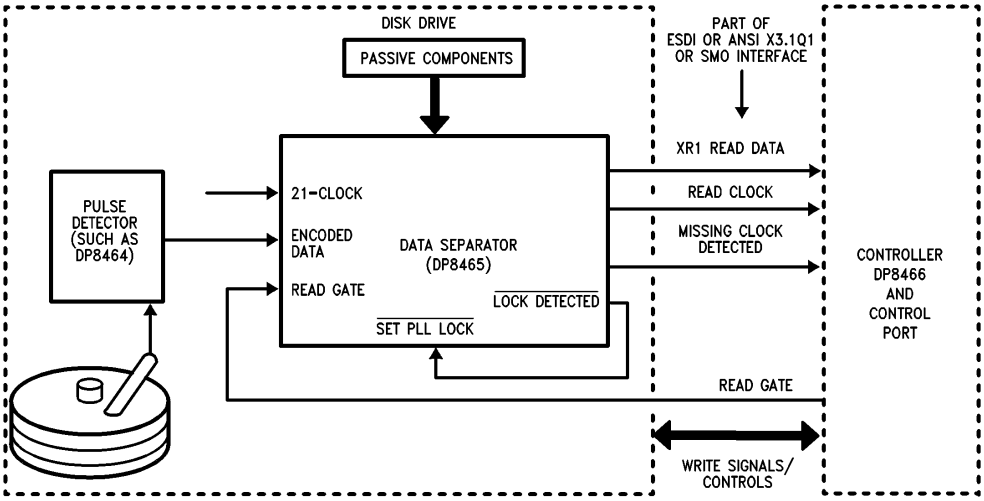


FIGURE 6.12 (a). Data Separator Residing in the Controller

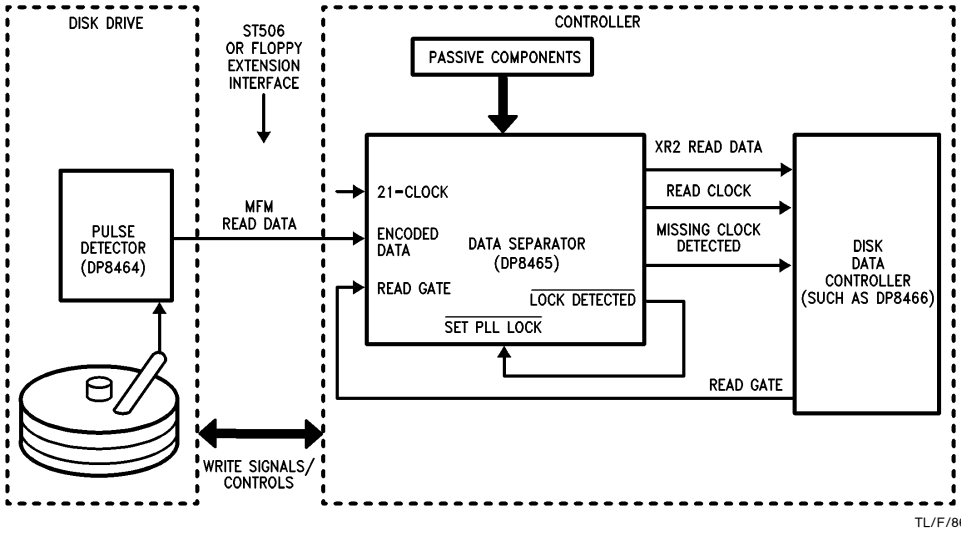
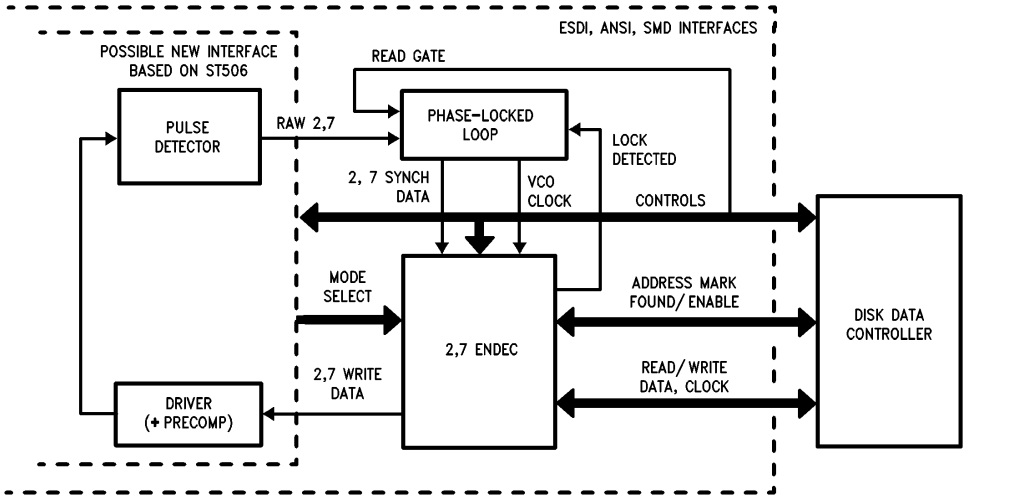


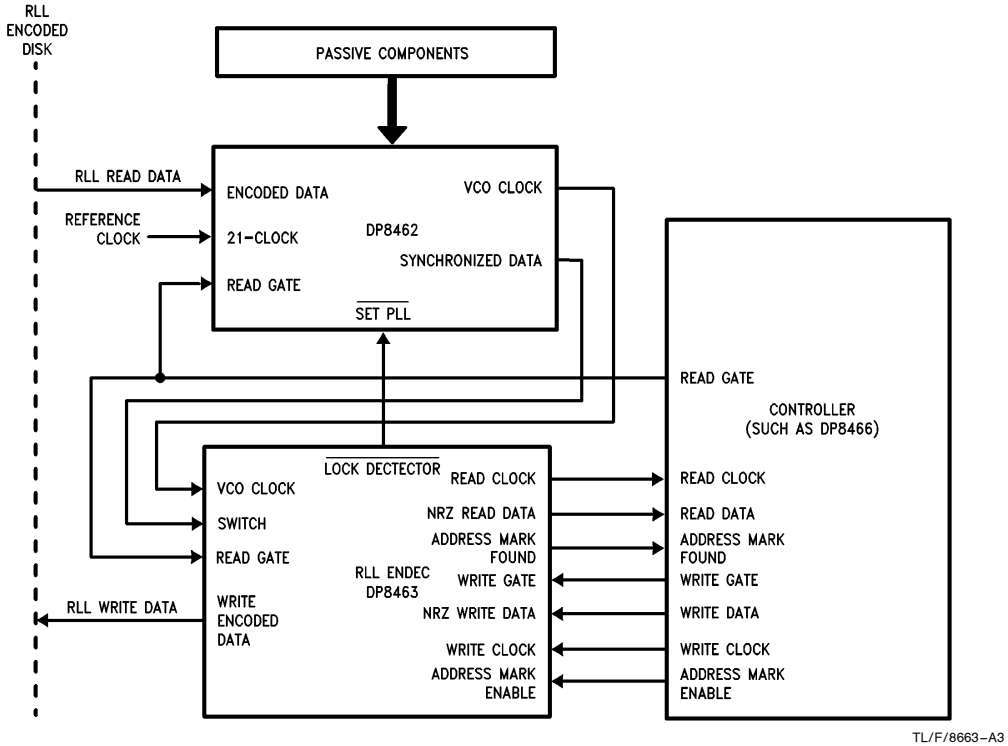
FIGURE 6.12 (b). Data Separator Residing in the Disk Drive

Interfacing the DP8463 2,7 ENDEC in a Disk System



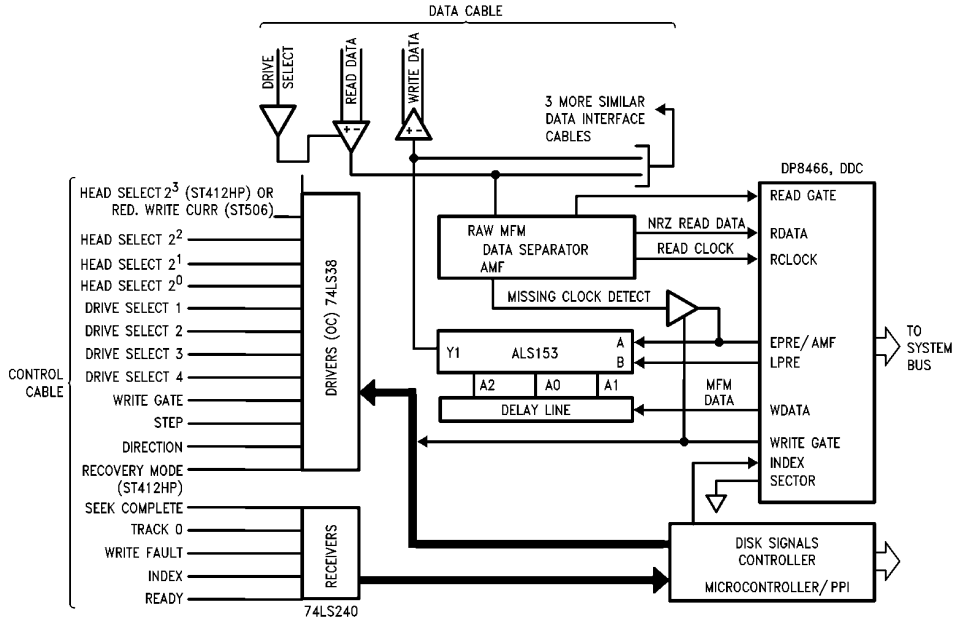
TL/F/8663–A2

FIGURE 6.13. DDC and DP8463-2,7 ENDEC (a) Generalized Disk System Block Diagram



TL/F/8663–A3

FIGURE 6.13. DDC and DP8463-2,7 ENDEC (b) Specific DP846X Solution



TL/F/8663–A4

FIGURE 6.14. Disk Data and Control Paths (ST506/ST412HP)

The necessary sequence of events (with associated timing restrictions) for proper read/write operation of the STxxx drive are shown in *Figure 6.15*. The DDC specifications are in compliance with the above requirements. The DDC has an on board encoder for MFM encoding and provides the precompensation outputs EPRE/LPRE, which are used by some external logic to generate the delays, as shown in *Figure 6.16(a)* and *(b)*

WRITE DATA PATH—PRECOMPENSATION

This consists of a differential pair that defines the transitions to be written on the track. The transition of the +MFM WRITE DATA line going more positive than the −MFM WRITE DATA line will cause a flux reversal on the track provided WRITE GATE is active. To ensure data integrity at the error rate specified for the interface, the write data presented by the DDC must be precompensated on the inner tracks. The optimum amount of precompensation is drive dependent, but usually is around 12 ns for both early and late written bits. In the DP8466 precompensation will be indicated on the EPRE and LPRE pins. Precompensation is issued for the middle bit of a 5-bit field. In the DDC, early and late precompensation will be enacted for all the combinations as shown below. All other patterns will not require precompensation. The center bit column is the present bit being output. The left two bit column is the bits previously shifted out, and the right two bit column is the two bits that will be shifted out next. In the following table a ''bit'' is a clock or data bit.

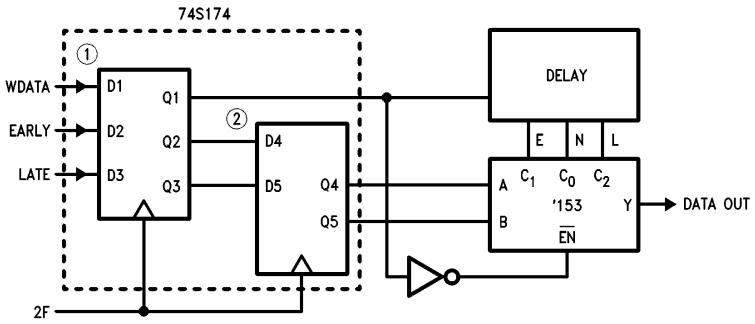
EPRE Patterns			LPRE Patterns		
00	1	10	00	1	10
00	0	11	00	1	11
01	1	00	10	0	00
01	1	01	10	0	01
11	1	00	10	1	10
11	1	01	10	1	11

READ DATA PATH—READ GATE

For the read data path the data separator is the critical block. Its design was discussed in chapter 3. In addition how the controller cycles the read gate will affect performance. In case of conventional soft sectored drives the Read Gate cycling by the DDC is different for different conditions. In the case when the address mark is not detected after lock has occurred (abort address mark function—internal to the chip), the DDC deasserts Read Gate 19 RCLKS after getting a non-zero bit on the Read Data line, where it has been receiving an all zeroes data. It will then reassert the Read Gate 17.5 RCLKS later. In the situation where there is a sync failure, Read Gate is deasserted 10 RCLKS after the failed sync word, and reasserts it 17.5 RCLKS later. In case of a Header failure or a CRC failure, Read Gate is deasserted 2 RCLKS after the last check byte and is reasserted 25.5 RCLKS later. *Figure 6.15(c)* shows the Read Gate timing details.

HANDLING THE READ GATE IN SOFT SECTORED ST506 DRIVES

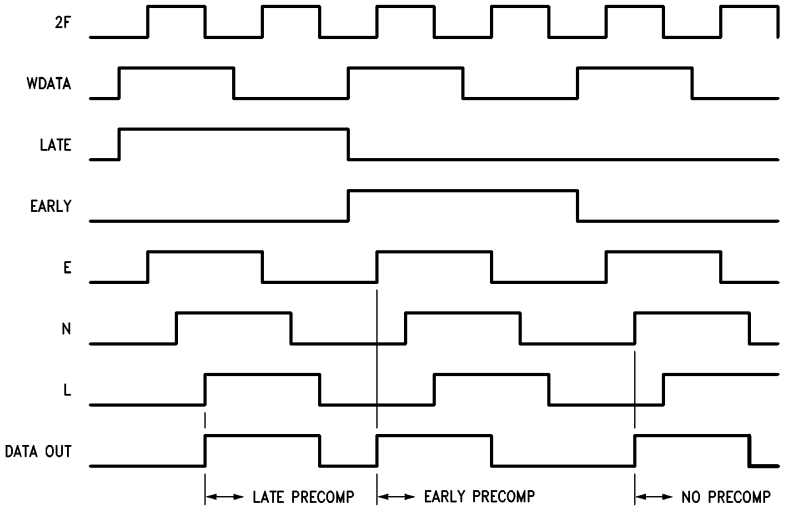
When reading soft-sectored drives, it is difficult to predict when Read Gate will be asserted. Data patterns can appear as preamble and the PLL will lock to these patterns. The controller is able to determine that the field is not a preamble by the address mark signal not being asserted because no missing clock violation was detected and deasserts Read Gate. However, Read Gate might be asserted over a write splice, which may result in the Data Separator failing to lock properly. It is usually up to the data separator design to ensure proper lock. The DDC does not implement a read gate on/off cycling algorithm. If the data separator can be thrown out of lock, the data separator should incorporate this algorithm. If not, this may lead to failure when accessing a sector in certain soft sectored drives. This is only a problem for drives using the conventional soft sectored format



TL/F/8663–I2

FIGURE 6.16 (a). Precompensation Circuitry (MFM Encoding)

Note: Latch ② essentially ensures that the mux is enabled before the early and late signals arrive.
The early, normal, and late compensated data are only relative to each other.
Precomp time is drive dependent, usually 12 ns.



TL/F/8663–I3

FIGURE 6.16 (b). Timing

(preamble, address mark, sync for both the ID and DATA segment), which is typical of lower performance, low capacity drives, operating at data rates of 5Mbits/sec and below, like the STxxx family. Newer standards replace the A1 (Hex) missing clock address mark currently used in soft sectored drives with an address mark which is a gap of no transitions at the beginning of the sector. Such a format is referred to as ''start with address mark'' format. The SMD and ESDI interfaces currently specify this type of operation and should be typical of those drives operating at 10 Mbits/s or greater.

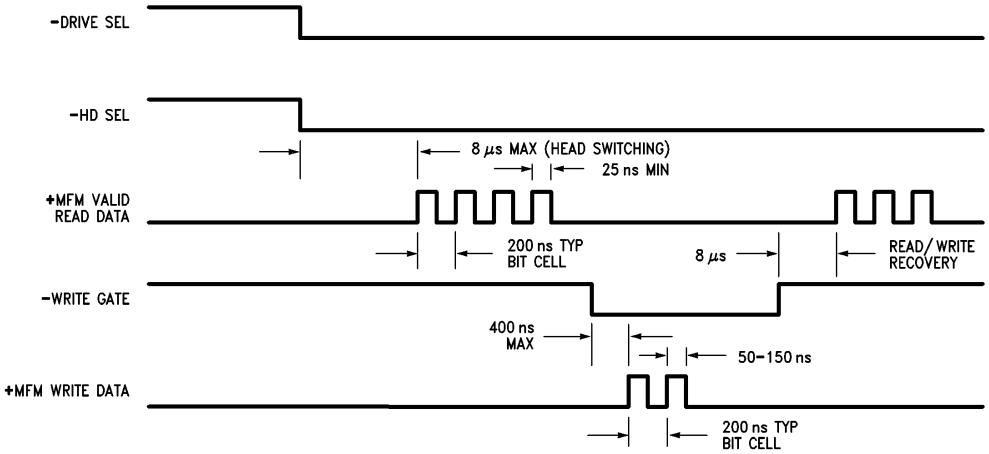
Operation of the DDC Read Gate in Soft Sectored Drives

In a soft sectored drive, the DDC asserts Read Gate at an arbitrary point over the track. This initiates a lock sequence on the Data Separator and the DDC begins to monitor the pattern entering its shift register. To avoid issuing garbage to the controller while it is locking up, the data separator usually will issue zeroes to the controller as it is looking for the first bit of non-zero data signifying lock. If using the 8465, when SET PLL LOCK is asserted, the Data Separator will begin to issue decoded data. In most applications, the LOCK DETECT output is connected to the SET PLL LOCK input, so that when the Data Separator has locked, it will issue data. However, if the Data Separator were to fail to lock and in some manner gets hung up where it will never be able to detect a preamble pattern, the system will be deadlocked since the DDC still is being given all zeroes from the Data Separator. The read operation will be aborted only after 2 revolutions. This type of failure can occur if Read Gate is asserted over write splice areas. The problem is really twofold, the Data separator is not issuing non-zero data to the DDC so that it can deassert Read Gate (the DDC thinks that the Data separator is still looking for the preamble) and the DDC is not placing a timeout on a response from the Data Separator.

Suggested External Logic to Timeout the Read Gate

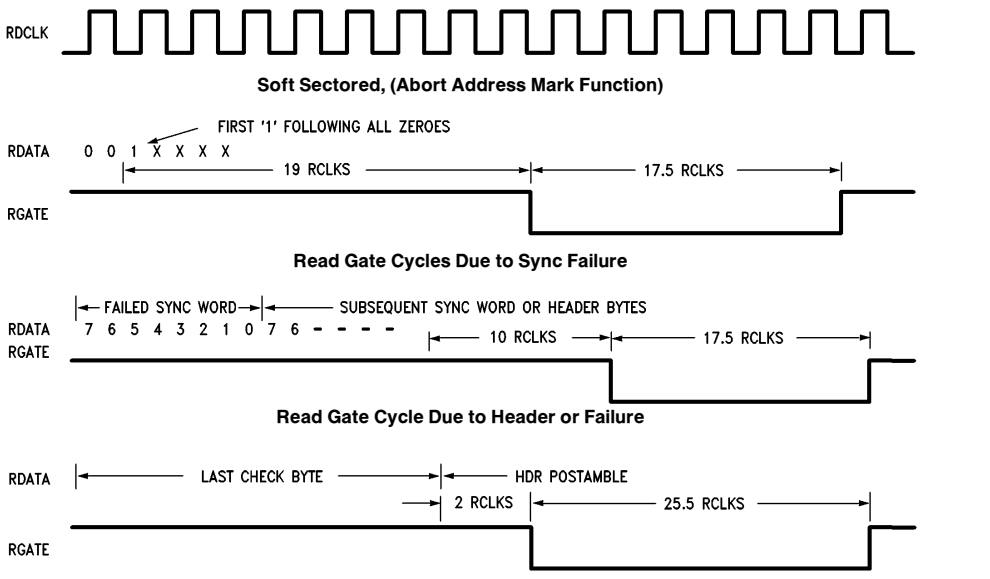
If necessary a simple external circuit can perform the time out function. When Read Gate is asserted a counter is started. After 3 to 4 bytes (allowing time for the Data Separator to acquire lock), the SET PLL Lock on the Data Separator is driven low. This enables data to be sent from the Data Separator to the DDC. The DDC monitors the data and if a '1' propagates through the deserializer without matching the first pattern programmed for the format, Read Gate is deasserted. The DDC deasserts Read Gate for approximately 19 bit times, *Figure 6.17(a)*, hence the Data separator should be able to resynchronize to the 2f clock within this time. Hence even in the worst case, the amount of preamble lost by this scheme is 6–7 bytes, (1 byte for the DDC to deassert Read Gate after receiving a '1', pattern not matched, 2 bytes of Read Gate deasserted time and 3–4 bytes of timeout, essentially the lock time of the Data Separator). The suggested circuit is shown in *Figure 6.17(b)*. This circuitry could also be implemented in a PAL.

Read/Write Data Timing



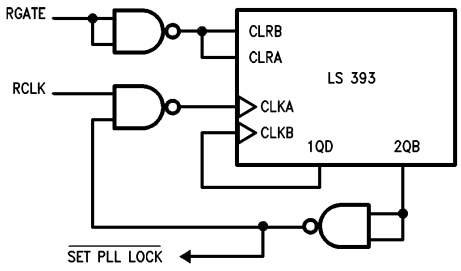
TL/F/8663–A5

FIGURE 6.15. ST506 Read/Write Data Timing



TL/F/8663–A7

FIGURE 6.17 (a). Read Gate Timing (Cycling)



Counter timer using dual 4 bit binary counters cascaded. Count is stopped by decoding bit 2QB which is reached after 32 RCLKS, counter is reset when RGATE is deasserted.

TL/F/8663–A8

FIGURE 6.17 (b). Read Gate Timeout Circuit

6.2.3 Interfacing the DDC to the ESDI Standard (Serial Mode)

The Enhanced Small Device Interface (ESDI), is a low cost, high performance interface suitable for the smaller memory devices currently on the market. It supports higher data transfer rate, 10–15 Mbits/s and provides for additional performance features desirable in higher performance systems. Two modes of implementation are possible: Serial mode of operation, utilizing NRZ data transfer along with serial commands and serial configuration/status reporting across the command cable (J1). Step mode implementation utilizes the same NRZ data transfer; however, the STEP and DIRECTION lines are used to cause actuator motion. This is similar to the ST506 type of interface as far as interfacing is concerned.

The ESDI interface consists of a 34-pin control cable and a 20-pin data cable. The control cable is attached in a daisy chain configuration while the data cable must be attached in a radial configuration. Hence all the control signals are associated with the drive selected. All the control lines are digital in nature (open collector TTL) and either provide signals to the drive (input) or signals to the host (output). The control signals and the serial commands are handled by the Disk Signals Controller, which is some local intelligence, as shown in *Figure 6.18*.

The ESDI interface can support both Hard and Soft sector drives. The interfacing of the, DDC is slightly different for hard versus soft sectored drives. *Figure 6.18* shows the interfacing of the DDC to a Hard sectored ESDI drive in the serial mode. In this configuration the sector pulse from the drive is used to identify the start of a sector. All lines associated with the transfer of data between the drive and the host system are differential in nature and may not be multiplexed. These lines are provided at the J2/P2 connectors on all drives. Four pairs of balanced signals are used for the transfer of data and clock.

NRZ Write Data: This defines the data to be written on the disk and is clocked by the WRITE CLOCK signal. This defi-

nition is compatible with that of the DDC. The Write data and Write Clock timings are shown in *Figure 6.19*.

NRZ Read Data: The data recovered by reading previously written information is transmitted to the host system via the differential pair of NRZ Read Data lines. This data is clocked by the READ CLOCK signal. These lines must be held at a zero level until PLL synch has been obtained and data is valid. This is compatible with what the DDC expects. One note of caution, when the PLL is locking on to the preamble, the DDC is receiving 0's and is looking for a non-zero synch byte. If the PLL goes off to harmonic lock or never locks and continuously outputs NRZ 0's, the DDC will assume it is forever looking at the preamble and will finally abort the command after two index pulses, flagging a sector not found error. It is up to the drive designer to prevent the PLL from going into harmonic lock. The DP8465, DP8461 and DP8462 are designed to remove harmonic lock. *Figure 6.19* shows the timing requirements for the read data.

Read/Reference Clock: *Figure 6.19* depicts the necessary sequence of events (with associated timing restrictions) for proper read/write operation of the ESDI drive. The Reference Clock signal from the drive will determine the data transfer rate. The transitions from Reference Clock to Read Clock must be performed without glitches. Read Clock and Read Data are valid within the number of PLL sync field bytes specified by the drive configuration after read enable and a PLL sync field is encountered. The interface Read/Reference Clock line may contain no transitions for up to two Reference Clock periods for transitions between reference and read clocks. The transition period will also be one-half of a Reference Clock period minimum with no shortened pulse widths. This is compatible with the specifications of the DDC, which has setup and hold times typically of the order of 15 ns. Reference Clock is valid when Read Gate is inactive while Read Clock is valid when Read Gate is active & PLL sync has been established.

Write Clock: Write Clock is provided by the DDC and must be at the bit data rate. This clock frequency is dictated by the Read/Reference clock during write operations. The DDC complies with the ESDI standard which requires the Write Clock to be active when Write Gate is active. See *Figure 6.19* for timing.

READ AND WRITE TIMING

Write Gate

The active state of this signal enables write data to be written on the disk. The low to high transition of this signal often creates a write splice and then initiates the writing of the data PLL sync field by the drive. The timing restrictions on Write Gate in the ESDI specification are as follows:

1) When formatting, Write Gate should be deactivated for 2 bit times minimum between address area and the data area to identify to the drive the beginning of the data PLL sync field.

2) It should be asserted at least two and a half reference clock periods after Write Clock.

3) The time lapse from deactivating Read Gate to activating Write Gate should be a minimum of 5 reference/read clock periods.

4) To account for data-encoding delays, Write Gate must be held on for at least two byte times after the last bit of information to be recorded.

5) It should be deactivated at least 1 μs before a head change and may not be activated until 15 μs after a head change.

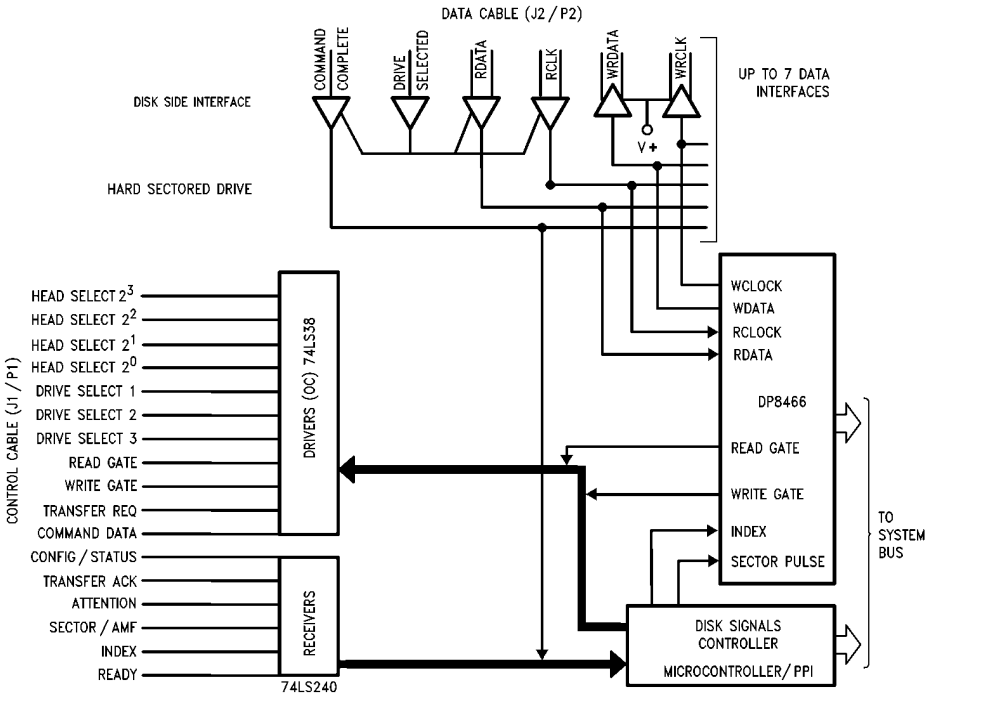


FIGURE 6.18. Data and Control Paths (ESDI-Serial Interface) Hard Sectored Drive

Read Gate

The active state of this signal, or low level, enables data to be read from the disk. This signal should become active only during a PLO sync field and at least the number of bytes defined by the drive prior to the ID or Data Sync bytes. The timing restrictions on Read Gate for the ESDI specification are as follows:

1) Read Gate must be false when passing over a write splice area. It must be deactivated 1 bit time min. before a Write Splice area and may be enabled 1 bit time min. after a Write Splice area.

2) The time lapse before Read Gate can be activated after deactivating the Write Gate is 10 μs.

The Read/Write Gate timings for format, write and read operations are discussed below with reference to the DDC, demonstrating its compatibility with the ESDI specifications.

Format Sector

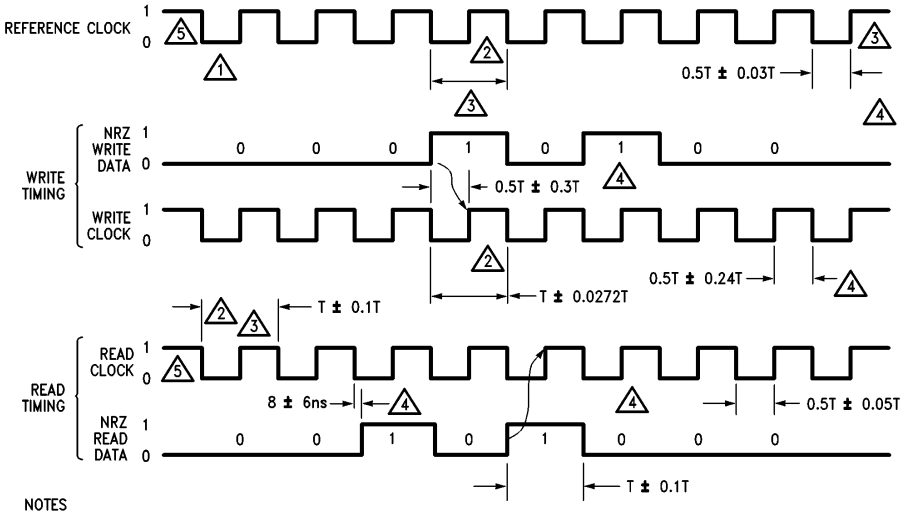
In the case of the DDC, during a format operation, it will do a continuous operation on the whole track. Thus after the index hole is sensed the DDC will assert WRITE GATE within 3.5 bit times. Write Gate will remain asserted until the index hole is sensed again. When each sector pulse is sensed (index for sector 0) the DDC will immediately start writing the various fields. After the data postamble is written the DDC will fill the rest of the sector with the gap until the next sector pulse is sensed. *Figure 6.20* shows the basic timing and the ESDI recommended format.

Write Sector

In case of a compare header-write data operation, the DDC will assert Read Gate within 3.5 Read Clocks from the rising edge of the Index or sector pulse. Read Gate is de-asserted 2 Read Clock periods after the ID check field, (plus a small propagation delay). Read Gate will remain de-asserted for the entire postamble. Due to internal delays, Write Gate is asserted 3 bit times into the data preamble field. Hence this would meet the 5 bit time minimum spec. between the read gate de-assertion and write gate assertion assuming at least one ID postamble byte is programmed. At the end of the write operation, Write Gate is removed 0 bit times after the data postamble. Hence a 3 bit time 'pad' is created between header postamble and data preamble updating data. This pad will contain data preamble as written during format operation. Refer to *Figure 6.20* for basic timing. It should be noted that a write splice of 8 bits is associated with the assertion of Write Gate. Hence if a write header operation is involved, then the read gate should not be asserted in the splice area.

Read Sector

In case of a compare header-read data operation, the DDC will assert Read Gate within 3.5 Read Clock cycles from the rising edge of the Index or sector pulse. Read gate is de-asserted 2 Read Clock cycles after the ID check field. Read Gate is re-asserted 11.5 bit times from the data preamble. This is 8.5 bit times from the point where Write Gate is as-



NOTES

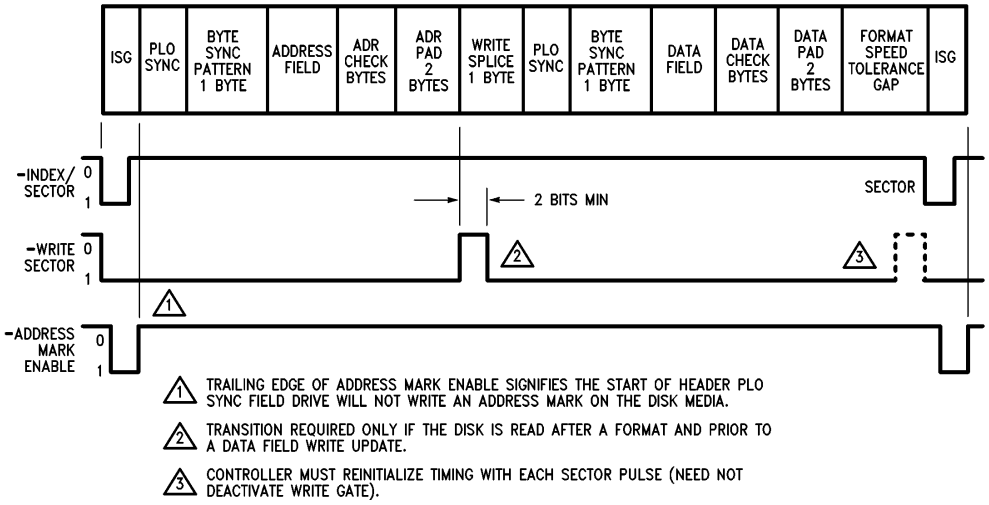
1. ALL TIMES IN ns MEASURED AT I/O CONNECTOR OF THE DRIVE T IS THE PERIOD OF THE CLOCK SIGNALS AND IS THE INVERSE OF THE REFERENCE OR READ CLOCK FREQUENCY.
2. SIMILAR PERIOD SYMMETRY SHALL BE IN ± 4 ns BETWEEN ANY TWO ADJACENT CYCLES DURING READING OR WRITING.
3. EXCEPT DURING A HEAD CHANGE OR PLO SYNCHRONIZATION THE CLOCK VARIANCES FOR SPINOLE SPEED AND CIRCUIT TOLERANCES SHALL NOT VARY MORE THAN −5.5% TO +5.0% PHASE RELATIONSHIP BETWEEN REFERENCE CLOCK AND NRZ WRITE DATA OR WRITE CLOCK IS NOT DEFINED.
4. TIMING APPLICABLE DURING READING OR WRITING.
5. REFERENCE CLOCK IS VALID WHEN READ GATE IS INACTIVE READ CLOCK IS VALID WHEN READ GATE IS ACTIVE AND PLO SYNCHRONIZATION HAS BEEN ESTABLISHED.

TL/F/8663–B0

FIGURE 6.19. NRZ Read/Write Data Timings

serted. This accommodates the 8 bit time write splice generated due to write driver turn on time as required in the ESDI specification. Read Gate is de-asserted 2 bit times into the data postamble. Of particular importance is that the read operation avoid reading the write splice. As can be seen from the write sector discussion, the write splice will occur 3 bits into the preamble, and the read will occur after 11.5 bits. Thus Read Gate is disabled during the actual write splice. When the read and write operations use the same format parameters, Write Gate will cause a splice 8.5 bits before Read Gate is asserted assuming insignificant delay in the read path from the media. However this provides a maximum of 8.5 bit times the write data out could be delayed by the write data encoder, in the drive, prior to being written on the media. *Figure 6.20* gives the basic timing. The user may have to account for additional delays specific to the drive.

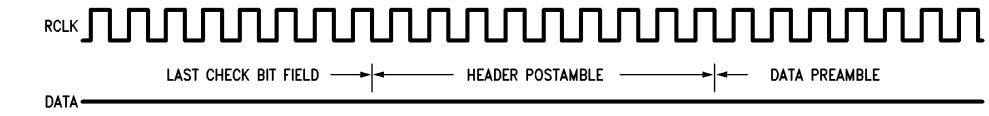
Write Gate in ESDI with Index/Sector, AME



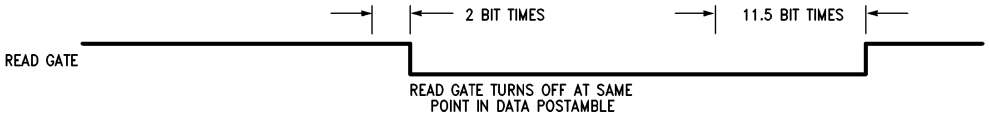
TL/F/8663–B1

FIGURE 6.20 (a). Read/Write Gate Timing for DDC

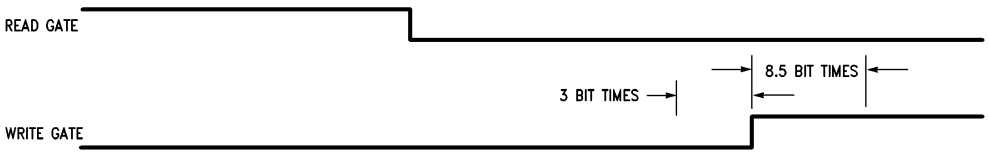
Read/Write Gate Timing at End of ID; Beginning of Data Field



Compare Header — Read Data Operation



Compare Header — Write Data Operation



TL/F/8663–B2

FIGURE 6.20 (b). Read/Write Gate Timing for DDC

6.2.4 Special Consideration for ESDI Drives

SOFT SECTORED

Interfacing the DDC, to a soft sectored ESDI drive is slightly different. There are no sector pulses demarcating sectors. An Address Mark pattern of no flux transitions is used to identify the start of the sector. The ideal soft sectored format (as supported by the DDC) consists of the Preamble field followed by the Address Mark (AM) and Sync fields. However in the case of ESDI the soft sectored format specification supports an AM at the beginning of the sector followed by the Preamble and Sync fields. This is a special field of no flux transitions which is essentially used to mark the start of the sector. Address Mark detection and generation is done using a handshake protocol between AME and AMF on the drive cable. Since the AMF on the DDC is geared to look for a missing clock violation in the address mark pattern, it cannot be used in the ESDI handshake. Also the DDC generates AME only during the format operation. Hence to achieve successful operation of the ESDI soft sectored drive, the DDC formats the drive as a soft sectored drive and while reading, the AMF signal from the drive is used to generate a sector pulse, as the DDC is configured to operate in the hard sectored mode. It also requires manipulation of the format parameters as shown in *Figure 6.22*. *Figure 6.21* shows the schematic of the interface at a block level.

Let us first consider manipulation of the format parameters to achieve proper operation. *Figure 6.22* gives the sequence of events. Consider the first three fields of the format. The DDC views them as Preamble, AM and Sync, *Figure 6.22(a)*. The DDC format is programmed as AM, Preamble and Sync for the first three fields, *Figure 6.22(b)*. If the format operation is initiated with SAM (start on address mark) bit set in the Disk Format register, the recording on the media is shown in *Figure 6.22(c)*. The AME output on the DDC is asserted during the Address Mark (field 1). During a Compare header-read/write operation the first field of the format parameter RAM in the DDC is programmed to be the Preamble. The count for the second field is set to zero, so that it is skipped, while the third field is progammed to be the Sync field. When data is read, the field 1 of no transitions (AM) results in AMF becoming active which generates the sector pulse for the DDC, *Figure 6.22(d)*.

In the ESDI specification, AME (address mark enable) line, when active with Write Gate causes an Address Mark to be written on the media. When AME is active without Write Gate or Read Gate, it causes a search for Address Marks. On detection of the end of Address Mark, AMF (address mark found) responds. The trailing edge of AME with Write Gate true initiates the writing of the header PLO Sync field.

To incorporate the AME/AMF handshake, external logic is required with the DDC. Since the DDC generates AME only during a format operation, the circuit would have to generate AME to the drive during a read operation and incorporate the handshake with AMF from the drive. This AMF is used as a sector pulse input to the DDC. This technique then results in a sector pulse corresponding to each address mark pattern demarcating each sector. However at sector 0 this poses a problem. The index pulse is followed by the post index gap and then the address mark field of sector 0 which would generate a sector pulse. As per ESDI requirements this circuit would have a delay and present the index pulse over the sector 0 pulse and block out the sector 0 pulse to the DDC. The other constraints to be maintained in order to comply with ESDI spec requirements are:

1) AME should be asserted min 100 ns after min write gate is asserted and be deasserted 100 ns before write gate is deasserted.
2) AME can be asserted again at least 10 μs after write gate deassertion.

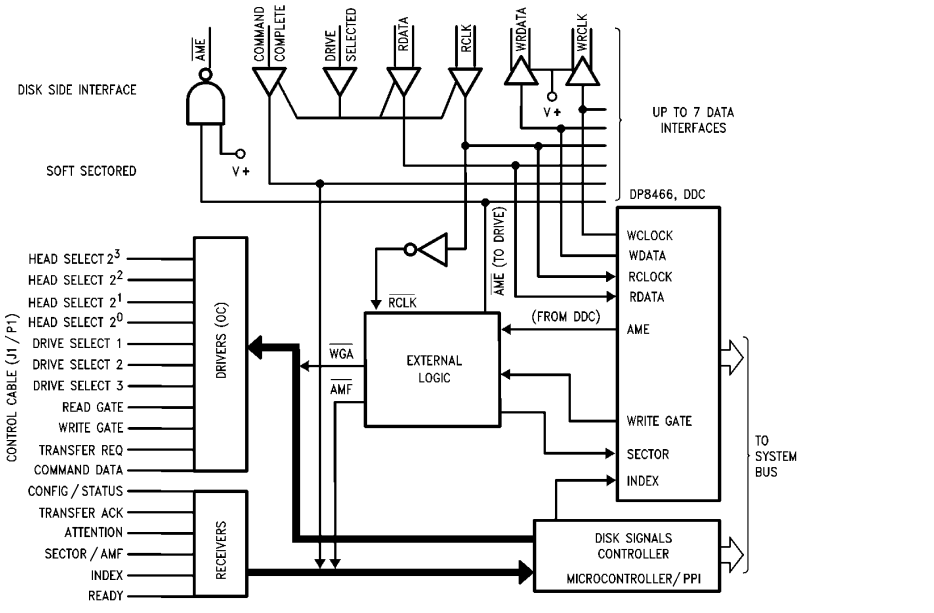


FIGURE 6.21. Data and Control Paths (ESDI-Serial) Soft Sectored Drive

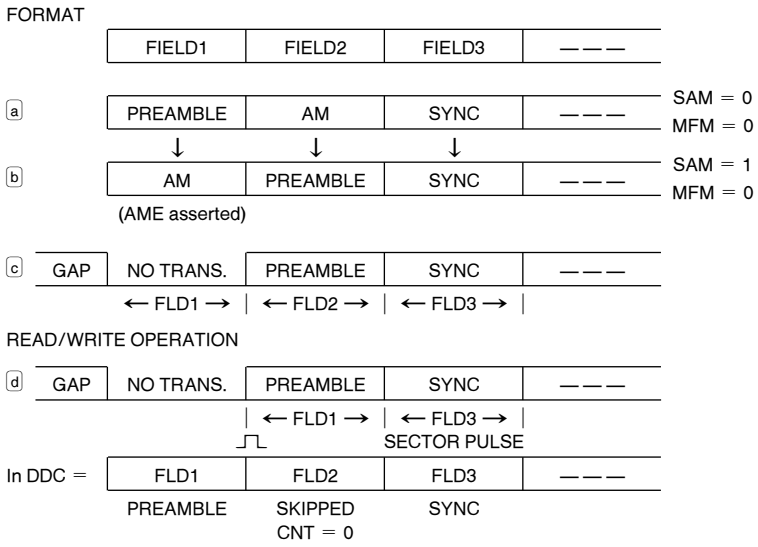


FIGURE 6.22. Manipulation of Format Parameters in DDC for ESDI Soft Sectored Operation

HARD SECTORED ESDI

For an ESDI drive which is hard sectored, the ESDI specification calls for an Inter Sector Gap (ISG) which is to precede and follow the index/sector pulses. The gap is needed to provide the drive with an area for the embedded servo (if used), and gives the controller time to assert read gate. While formatting the drive, the end of the ISG is indicated by the removal of the address mark enable signal (AME). This signal is needed by the drive to indicate the beginning of the PLL field, necessary when the disk encodes the PLL field with a non-standard pattern (as with 2,7 encoding with 3t preambles). The DDC is capable of generating the AME signal with the necessary timing. The DDC needs to be in the hard sectored mode, and have the Start with Address Mark bit (SAM bit of the DISK FORMAT register) enabled. The DDC ID Preamble field now becomes the ISG following the index/sector pulse, and the ID Sync 1 field becomes the PLL preamble field. While not formating, this feature is not needed, and should be disabled.

When the header field of an ESDI drive is read (or compared), the read gate to the drive needs to be delayed until after the ISG. The DDC generates read gate only after receiving a sector or index pulse, so by delaying the sector and index signals to the DDC the read gate will be delayed.

COMBINED SOLUTION

A solution to the above problems can be provided by 1 PAL device and something to provide a delay (possibly another PAL device) *Figure 6.23(b)*. The interface solutions can be grouped into two main areas: Address marks and Index/Sector.

The address mark control needs to provide the following:

(1) Direct connection of AME to the drive and AMF to DDC sector input when formating a hard sectored drive.

(2) Delay the leading edge of the AME by the width of post index ISG when formating a soft sectored drive.

(3) Provide AME/AMF handshaking when not writing the disk, and properly change from reading to writing and back with soft sectored drives.

The index/sector control needs to provide the following:

(1) Delay the index and sector pulses from a hard sectored drive when not formating.

(2) Generate index and sector pulses to the DDC from AMF and Index when using a soft sectored drive while not formating.

The Address Mark Machine

The address mark machine consists of a pair of multiplexers which feed the AME input to the disk drive and the AMF input to the DDC. A state machine ensures the proper sequence of events, while a timer provides delay. The address mark machine works in the following way:

When in the soft sectored mode and not formating, it will assert AME to the drive. When AMF is detected, AME is removed until AMF is no longer detected. This uses states 0 and 3 of the address mark state machine, as shown in the diagrams, *Figure 6.23(a)*.

When write gate is detected, AME is removed, and write gate to the drive is generated a short time later. When the write operation is ended, write gate is removed from the drive, and AME is enabled a short time later.

In all other modes of operation, the state machine is deactivated and write gate is delayed to the drive by one bit clock time (circuit convenience). The multiplexer continues to provide the drive and DDC with proper signals.

The Index/Sector Machine

The index/sector machine consists of a pair of multiplexers which feed the index and sector inputs of the DDC. While formating, the multiplexers connect the index and sector signals from the drive to the DDC.

When not formating a hard sectored drive, the state machine waits for either a sector or index pulse. When this is detected, the machine waits for a delay and then generates a sector or index pulse. A flip flop, borrowed from the address mark machine, is used to record whether an index or sector pulse should be generated (the address mark machine is disabled when in the hard sectored mode). The delayed index or sector pulse is generated for another delay period of time.

With a soft sectored drive, the state machine waits for either an index or AMF signal. If index is detected, the machine will wait until an AMF is detected. An index pulse will then be sent to the DDC without a sector pulse. If a AMF pulse is detected without index, the state machine will generate a sector pulse.

6.2.5 Interfacing the DDC to the SMD Interface Standard

The Storage Module Device (SMD) interface is a high performance interface, extremely popular with 8″–14″ drives. It provides features similar to the ESDI interface, with some differences. It supports higher data rates from 10 Mbits/s to 24 Mbits/s and utilizes NRZ data transfer along with parallel command and status reporting across the command cable. The SMD interface consists of a 60-pin control (A) cable and a 26-pin data (B) cable. The control cable is attached in a daisy chain configuration while the data cable must be attached in a radial configuration. The control signals are handled by a separate Disk Signals Controller block, which is some local intelligence, as shown in *Figure 6.24*. Only the interfacing of the data path signals are discussed as they are of relevance to the DDC. The control signals could be easily done by a local microprocessor. All lines associated with the transfer of data between the drive and the host system are differential in nature and may not be multiplexed. These lines are provided on the B cable of all drives and are briefly discussed below:

Write Data

This line carries NRZ data, to be written on the disk surface and must be synchronized with Write Clock. This definition is compatible with that of the DDC.

Write Clock

This is a retransmitted clock signal of the servo clock (IF Write Clock) issued by the controller.

Servo Clock

This signal is used by the control unit in the drive to synchronize Write Data with the Clock. Servo Clock may be available during unit ready status except during read operations, or at all times.

Read Clock

This line transmits Read Clock. The Read Data is synchronized with 1F Read Clock. This line may be valid only during a read operation, or may be multiplexed with the Servo Clock signal at other times.

Read Data

This line transmits the recovered data in the form of NRZ data synchronized with 1F Read Clock.

Figure 6.25 shows the basic timing requirements for the above signals for the Fujitsu M2311 micro disk drive hereafter referred to as MDD. In general ''SMD'' type drives have similar timing requirements.

READ AND WRITE GATE

Read and Write Gate are the two signals which are used to read/write data from/to the specified track/sector. In the SMD interface, they are present on the Bus Out, (bit 1—read gate, bit 0—write gate), when enabled by tag 3. Write Gate enables the write operation and is validated only when Unit Ready, On Cylinder and Seek End are true and Seek Error, Fault, File Protect, Offset are false. If Write Gate is turned on in cases other than the above conditions, fault occurs and writing is inhibited. At this juncture it would be appropriate to mention that there are certain drive dependent constraints which must be taken care of while interfacing to the SMD drives. These are drive dependent, and representative values observed in most of the drives are given below:

Write circuit turn-on delay: approximately 8 bit times.

Head select transient: A 5 μs delay minimum must be provided between head select and initiating read gate. Normally this is provided by selecting the appropriate length for the gap after the data postamble and the gap after the index/sector pulse.

Read-after-Write transients: A minimum delay of 10 μs must be provided between the trailing edge of write gate and the leading edge of read gate. This could also be done by adjusting the lengths of the gap after the data postamble and the gap after the index/sector pulse.

Read/Write Encoding/Decoding delays: Through encoding and decoding circuitry, a read data signal will be delayed by approximately one byte against a write data.

Write-after Read transient: A minimum delay of 0.3 μs must be provided between the trailing edge of read gate and the leading edge of write gate. This is accomplished by having at least one byte of header postamble.

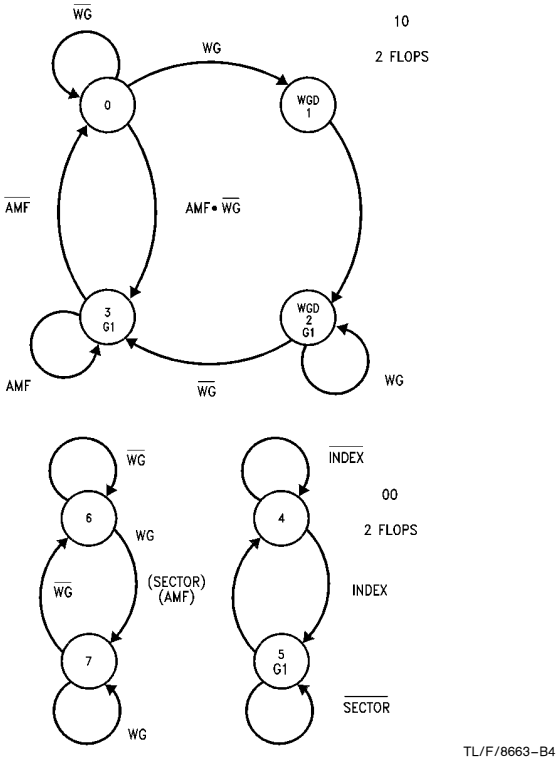


FIGURE 6.23 (a). AME/AMF/Index/Sector Logic State Diagrams

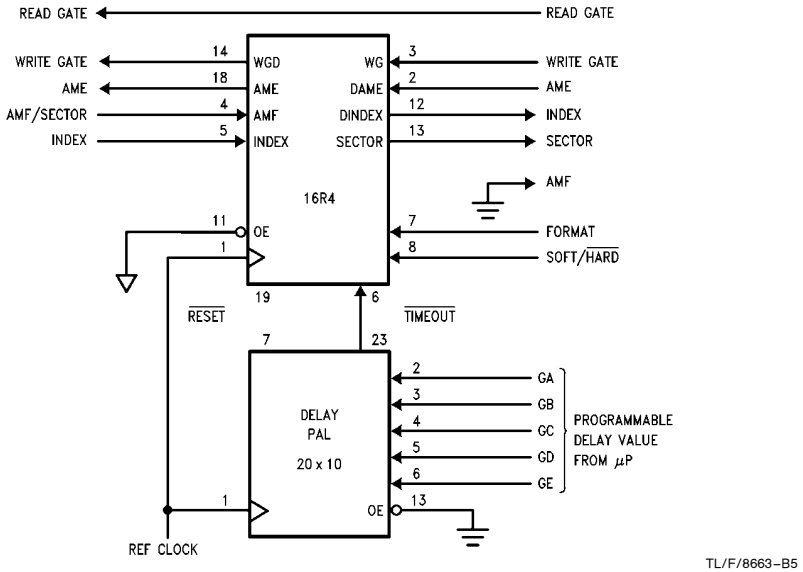
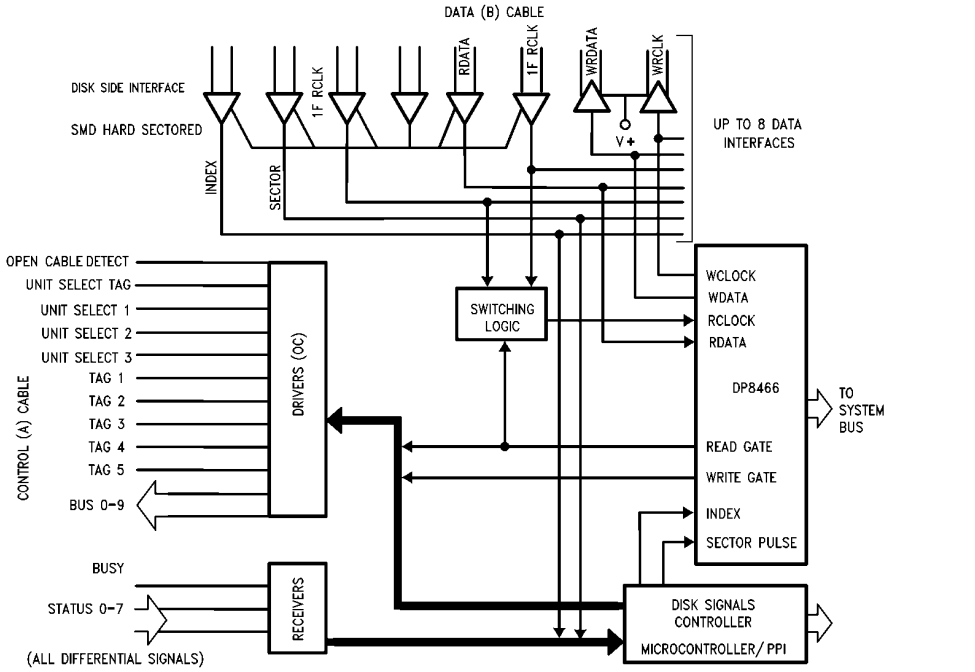


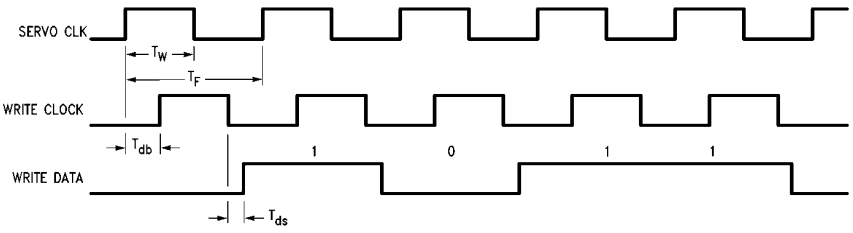
FIGURE 6.23 (b). ESDI AME/AMF Handshake Logic and State Diagram



TL/F/8663–B6

FIGURE 6.24. Data and Control Paths — SMD (Hard Sectored Drive)

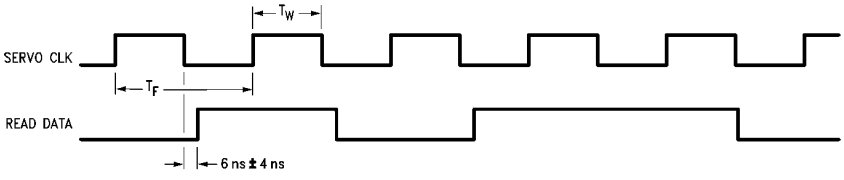
1F Write Clock, Write Data/Write Clock Timing



TL/F/8663–B7

T_W = TF/2
TF = (Function of Transfer Rate)
Tdb = Continuous delay within 2 bits
Tds = 0 ±10 ns

FIGURE 6.25 (a). Write Clock/Write Data Timing



T_W = TF/2

TL/F/8663–B8

FIGURE 6.25 (b). Read Clock/Read Data Timing

Preamble length: The synchronization time required to allow the PLL to synchronize is 11 bytes before the sync pattern of address and data fields.

READ/WRITE TIMING

Representative read/write timing for format write, data read and data write operations are shown in *Figure 6.26*. The corresponding timings for the DDC are given in *Figure 6.20*.

FORMAT WRITE

During a format operation or write header operation, the drive requires that the Write Gate be asserted by 600 ns (max.), from the index/sector pulse and Write Gate must be de-asserted at least 1 byte after the check byte field in the header, refer *Figure 6.26(b)*. In the case of DDC, Write Gate is asserted 3.5 bit times after the Index pulse in a format operation or after the index/sector pulse in a write header operation (hard sectored drive). Write Gate is de-asserted at the end of header postamble field. Hence if the header postamble is kept at least one byte long, the DDC should satisfy the requirements of the drive. *Figure 6.26(a)* shows the recommended format used by ''SMD'' drive manufacturers like CDC, etc.

DATA WRITE

During a data write operation (essentially a compare header-write data operation) read gate is asserted by the DDC 3.5 bit times from the rising edge of the index/sector pulse. The recommended format for the MDD has a post index/sector gap and requires the Read Gate to be asserted 6 μs (approx. 8 bytes) from the index/sector pulse. This can be done in a similar fashion as outlined for the ESDI spec. (refer section—Handling the Post Index/Sector Gap in the ESDI format). Read Gate is de-asserted 2 bit times after the ID check bits field, which satisfies the requirement of 8 bit times maximum. Write Gate is then asserted 3 bit times after the header postamble, which implies that the header postamble can be a maximum of 3 bytes long in order to maintain the requirement of 4 bytes max. by which Write Gate must be asserted after the header check field. There is also a condition that Write Gate must be asserted at least 300 ns after Read Gate is de-asserted. This is satisfied by the DDC as seen from *Figure 6.20(b)*, the minimum time of de-assertion being 9 bit times assuming at least a 1 byte postamble. Write Gate is de-asserted at the end of the data postamble, hence, the data postamble must be at least 4 bytes long. *Figure 6.26(c)* gives the timing requirements for a data write operation.

DATA READ

During a read operation (essentially a compare header-read data operation), DDC asserts the Read Gate 3.5 bit times after the index/sector pulse. Hence the post index/sector gap has to be handled in a similar fashion as discussed in the Data Write section. Read Gate is de-asserted by the DDC 2 bit times after the ID check bit field, which is well within the MDD requirement of 8 bits max. If the continuing operation is Read data, then Read Gate is re-asserted by the DDC 11.5 bit times from the data preamble, *Figure 6.20(b)*, which certainly satisfies the requirement of a 1 byte minimum before re-assertion of the Read Gate. Read Gate is de-asserted by the DDC 2 bit times after the data check bit field. *Figure 6.26(d)* gives the timing requirements of the data read operation.

SPECIAL CONSIDERATIONS FOR SMD SOFT SECTORED DRIVES

For a soft sectored ''SMD'' drive, the recommended format is similar to the one in ESDI. The sector starts with the address mark field (three bytes of no flux transitions). The AME signal is available on the bit 5 of the bus with Tag 3 active while the AMF signal is available on bit 5 of the Status bus with both Tag 4 and Tag 5 inactive. The AME/AMF handshake is handled in a similar fashion as discussed in section 6.2.4.

HANDLING THE SEPARATE CLOCKS FOR READ AND WRITE OPERATIONS IN THE SMD DATA CABLE

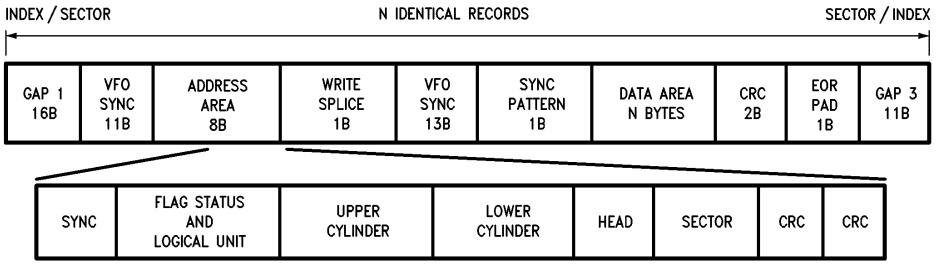
The SMD interface has two reference clocks, one for write (1F WCLK/SERVO CLOCK) and one for read (1F RCLK/READ CLOCK). The DDC requires that the Read/reference clock be provided on the same line (RCLK) and that the switching between the reference clock and the PLL locked frequency clock be such that there are no glitches on the line going to the DDC. To accomplish this the two clock signals, 1F Read Clock/READ CLOCK and 1F Write Clock/Servo Clock need to be multiplexed during read and write operations to switch the appropriate clock signal going to the DDC. It should also be made sure that there are no glitches or short pulses in the process of switching. Since the PLL has a certain finite time for lock, the actual switch of the clocks occurs after a finite lock time from Read Gate assertion. Hence the Read Gate used to mux the two clocks must be delayed by the lock time (worst case) before it is sent to the switching logic. This is shown as the box 'switching logic' in *Figure 6.24* and given in detail in *Figure 6.27*. Besides these other timing requirements are compatible with those of the DDC. This circuitry has also been realized in a PAL.

6.2.6 Miscellaneous ESDI/SMD Considerations

As with most standards, actual devices that follow the standards have their own idiosyncracies. ESDI and SMD standards are no exception. In addition some standards define design constraints which do not necessarily exist with actual devices. Some of these have been discussed earlier (for example de-assertion of Write Gate between ID and Data fields in ESDI). The following subsections describe some additional SMD & ESDI considerations that may be necessary depending on actual drive implementation.

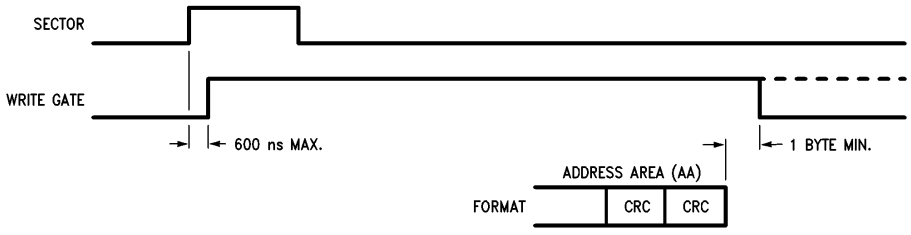
HANDLING THE OPTION OF DE-ASSERTION OF WRITE GATE BETWEEN THE ID AND THE DATA FIELDS, IN THE ESDI FORMAT SPECIFICATION

The option of de-asserting Write Gate between the ID and the Data field is not directly supported by the DDC. The purpose of this may be indication to the encoder, of the start of the data preamble field in case of RLL encoding, so that the encoder can send actual data rather than the encoded data for the preamble pattern. To support this, external logic can be used. This logic would be gated from the trailing edge of the DDC's Serial Data Valid signal, count until the 2 byte pad, (header postamble) has been sent (at the end of the ID segment), then force Write Gate low for the desired time, and then re-enable it. This problem can also be effectively circumvented by first doing a two pass format operation. This first pass does a format operation as above, then a second pass-write data operation is done on all of the sectors. If the Write Gate pulse is used to initiate a drive generated preamble, then by performing a compare header-write data operation the correct data preamble will then be written, and the data field can then be properly read, refer *Figure 6.18* for timing with respect to the DDC.



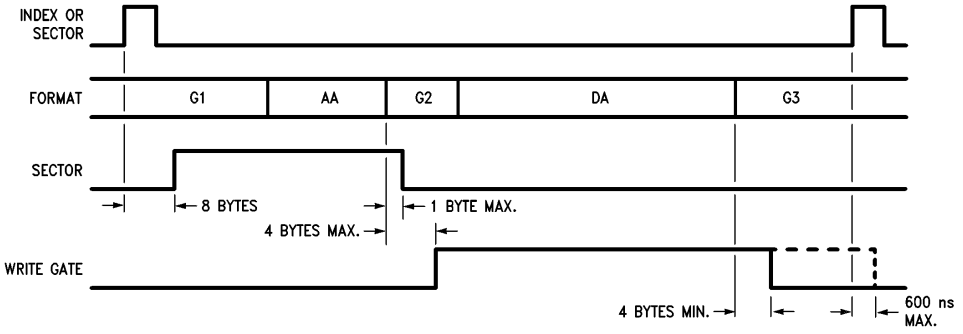
TL/F/8663–B9

(a) Format



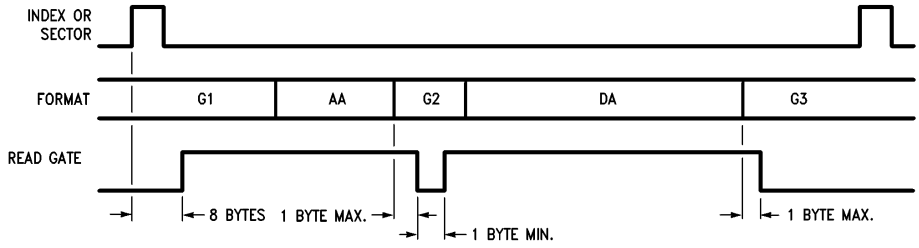
TL/F/8663–C0

(b) Format Write Timing



TL/F/8663–C1

(c) Data Write Timing



TL/F/8663–C2

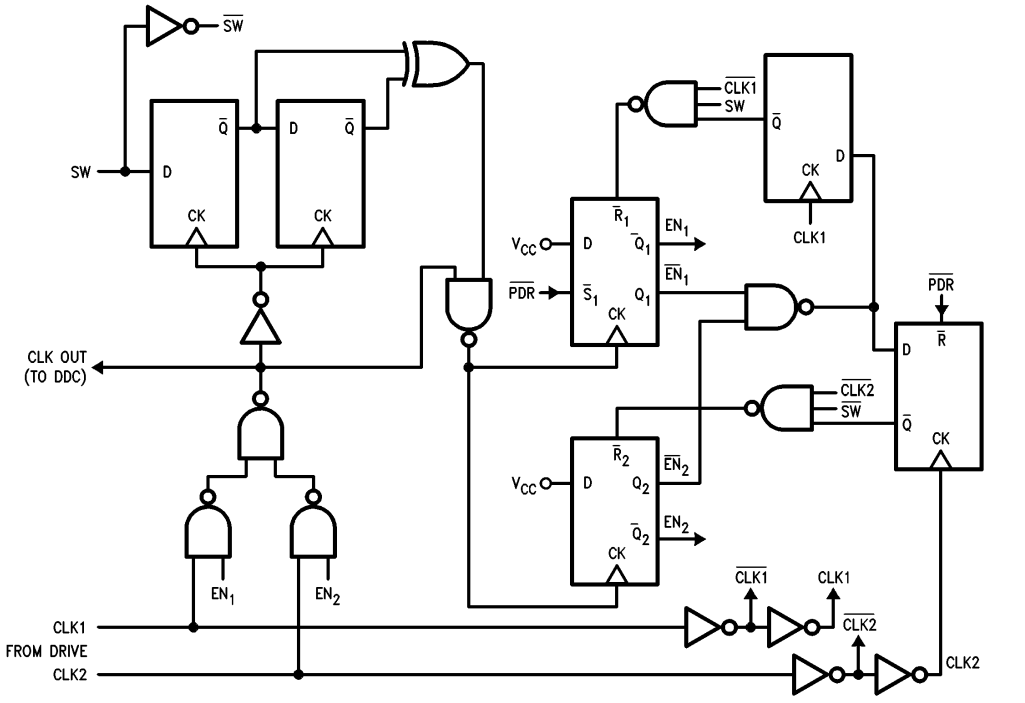
(d) Data Read Timing

FIGURE 6.26. Read/Write Timing (SMD)

A note of caution to be observed for ESDI hard sectored drives during format operation for RLL encoding. Since the leading edge of the write gate may be used by the encoder to trigger the generation of the preamble, this would be a problem as Write Gate is asserted 3.5 bit times after the index pulse and there would be no de-assertion of Write Gate with each sector pulse. The two pass format operation is a good software solution. In hardware the inverted sector pulse could be ANDed with Write Gate to generate the Write Gate to the encoder. Generally after a format write, then read is necessary to determine defective sectors, and initiate some sector substitutions.

HANDLING THE WRITE SPLICE FIELD BETWEEN THE ID AND DATA SEGMENTS IN THE ESDI/SMD FORMAT

The ESDI/SMD format specification recommends a two byte header postamble and a one byte write splice. The DDC format parameters indirectly support a write splice field between the ID and data segments. Consider normal operation of the DDC. The format is programmed to have a 2 byte header postamble and a data preamble one byte longer than the desired length. This byte is taken as the write splice, (a floating byte). During a write operation this floating byte is considered as part of the data preamble, so write gate is asserted 3 bit times into the data preamble i.e., the 'write splice' and data would be written on the media after taking into effect the write propagation delay. In case of a read operaion this byte is taken to be part of the header postamble. Hence as Read gate is asserted after the header postamble, it would never be asserted in the write splice. Normal operation could be achieved with the DDC without physically having a write splice field between the ID and the Data segment, rather creating one by adjusting other field lengths.



TL/F/8663–C3

FIGURE 6.27 (a). MUX and Deglitcher Circuitry to Switch between Read Clock and Reference Clock

Note 1: SW is low at start up, (L selects CLK 2, H selects CLK 1)

Note 2: POR = Power on reset ($EN_2 \rightarrow 1$, $EN_1 \rightarrow 0$)

Note 3: Worst case latency (SW to actual switching) = 1.5 periods of clock switching from +2 periods of clock switching.

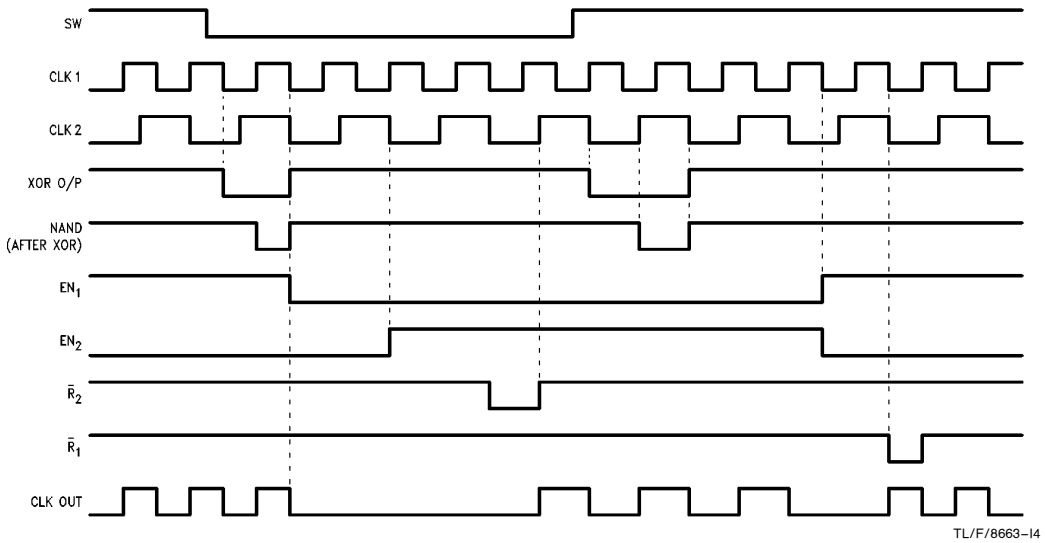


FIGURE 6.27 (b). Timing Diagram for Switching Logic

HANDLING THE POST INDEX/SECTOR GAP FIELD IN THE ESDI/SMD FORMAT SPECIFICATION

In the recommended format of the ESDI/SMD specification there is a gap after the index/sector pulse, referred to as the post index/sector gap. This is necessary mainly to accommodate head transients, read-to-write transients, write-to-read transients, etc. In the DDC, there is no format parameter to implement this field. Hence to implement this, external logic is needed, whereby the index/sector pulse to the DDC is delayed from the index/sector pulse from the drive by the desired gap count using counters. This is done for all format, read and write operations. The gap pattern for the intersector gap is then written for this post index/sector gap field also. Refer *Figure 6.28*.

READ GATE DELAY

As discussed earlier, the separation between assertion of Read Gate and Write Gate at the beginning of the sector is 0.5 to −0.5 bit times. This may not accommodate the write splice associated with the Write Gate due to write driver turn on time etc., which is generally about 8 bit times from write gate assertion. Hence with the existing timing Read Gate may get asserted in the write splice area while reading the header. This might be a problem during the Format operation, in the very first sector of the track, because Write Gate is asserted after the Index pulse and remains asserted through the track till the Index pulse is encountered again after one revolution of the disk. Hence for the very first sector, Read Gate would have to be delayed to avoid the write splice. It would not affect other sectors.

An important point to note here is that if a Write Header operation is done on any sector, then the Read Gate may have to be delayed for that sector also.

6.2.7 Intelligent Disk Interfaces

The overall objective of interfaces in this category, (like SCSI, IPI), was to make it easier for computer systems to talk to disk drives while ensuring minimum overhead for the host system. The Controller would incorporate a drive level interface on the disk side and a well defined interface to the host bus. The controller board has a local microprocessor which essentially controls the disk controller (data path), disk control signals (control path) and communication with the host system. Actually the DDC interfaces directly to the local bus. The microprocessor essentially controls the transfer of data using the DDC's DMA capability, from the local memory to the system memory, etc. *Figure 6.29* shows a block diagram of a high performance mass storage system, incorporating a SCSI peripheral bus.

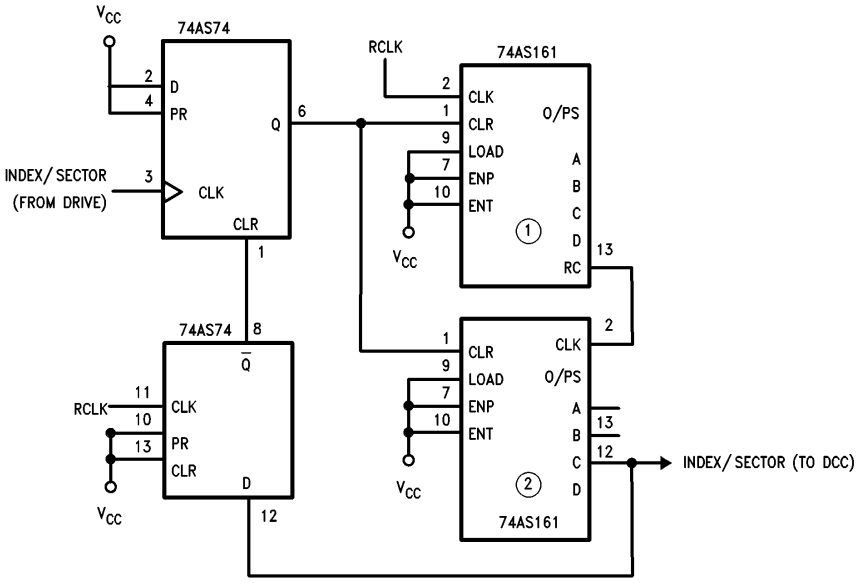


FIGURE 6.28. Logic to Implement Post Index/Sector Gap Field

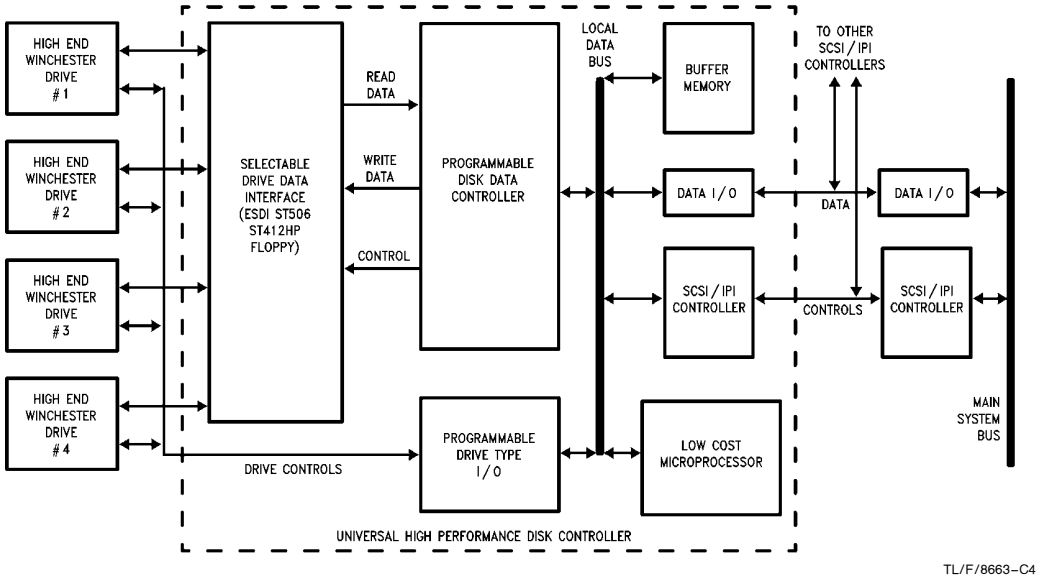


FIGURE 6.29. High Performance Mass Storage System

6.3 CIRCUIT BOARD LAYOUT AND SUPPLY ROUTING

There are several considerations to PC board design that should be followed. These guidelines serve to minimize problems that can occur with any high speed digital device. Since the DDC can operate at speeds approaching 30 MHz on the disk side, and up to 20 MHz on the system side, typical high speed design techniques should be employed to reduce noise, transmission line, and crosstalk effects. These are described below.

6.3.1 General Layout Considerations

The DP8466 has two design areas of routing and loading on its signal lines, namely: disk interface and bus interface signals. For the disk signals, Read Data, Read Clock, Write Data, and Write Clock can be very high speed signals. It is recommended that when interfacing these signals to the interface line drivers/receivers, these lines be kept short as possible. Also the data cable interface devices should be located close to the data connector. It is especially important to minimize noise, propagation delay skews and jitter on these lines when in MFM mode because excessive skew, noise and jitter can lead to increased error rates. A generalized PCB chip layout is shown in *Figure 6.30*.

The bus signals generally should be treated the same as any VLSI's bus interface. The Address/Data bus provides 16 lines capable of 2 mA DC drive, and can drive variable loads up to about 100–120 pF at 20 MHz (larger loads can be driven although not at full speed). If more than 150 pF or 2 mA load needs to be driven, the data bus should be buffered, as shown in *Figure 6.31*. The one ''trick'' is to always connect the address de-multiplexing latches directly on the DDC, this ensures maximum address latch strobe setup time. The buffers can then drive the rest of the system. As with any high speed bus good layout practices should be followed to minimize crosstalk and reflections.

6.3.2 Decoupling, and V_{CC} and Ground Routing Guidelines

Due to the combined high speed and wide data bus of the DDC it is important to follow good power supply layout practices. Any noise generated by toggling of outputs can cause noise on V_{CC} and this in turn can reflect noise back into other inputs or outputs. The result is noise and glitches appearing on other inputs or outputs. This can be especially true when the address/data bus is toggling many lines simultaneously. In this case it is not unusual to have peak current spikes up to 300 mA being generated when toggling 16 or more outputs.

V_{CC} and ground noise problems can be easily minimized by following some simple rules when laying out the PCB. In general, multi-layer printed circuit layouts should include V_{CC} and ground planes. If a simple two-sided board, then wide V_{CC} and ground traces should be used, and an effort to layout V_{CC} and ground planes on the foil and component sides should be made. Most importantly, the DDC should have a decoupling capacitor placed as close to its V_{CC} and ground pins as possible, as shown in *Figure 6.32*. This capacitor should be a low series inductance type ceramic capacitor. Additionally, it may be desirable to add a 3–10 μF tantalum capacitor in parallel with the ceramic to offer further decoupling.

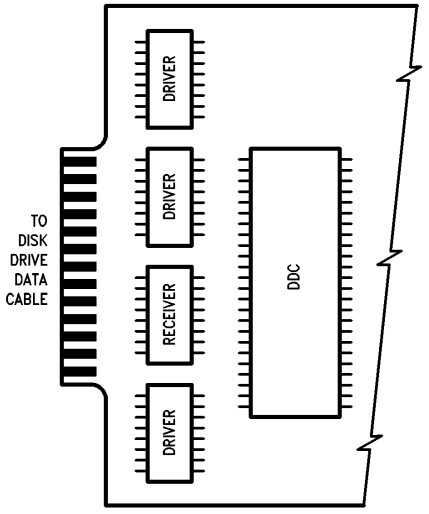
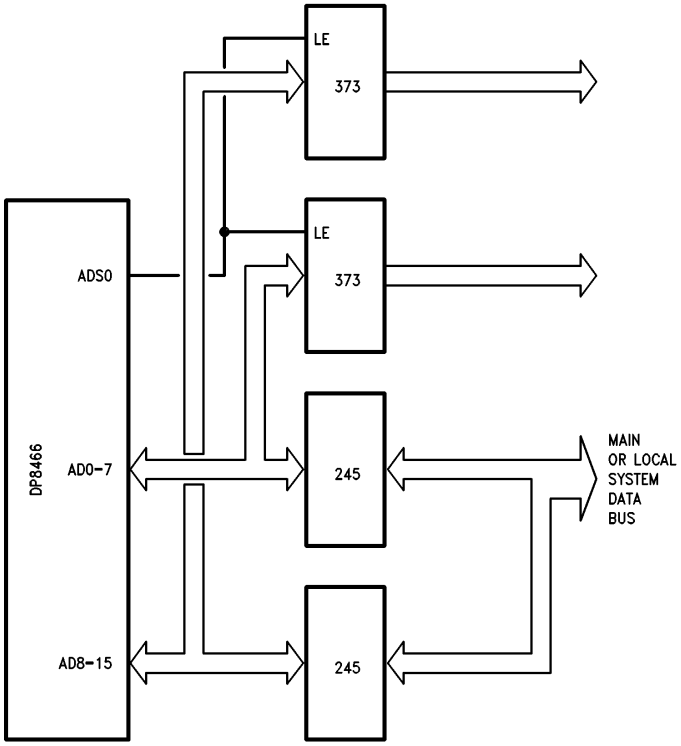
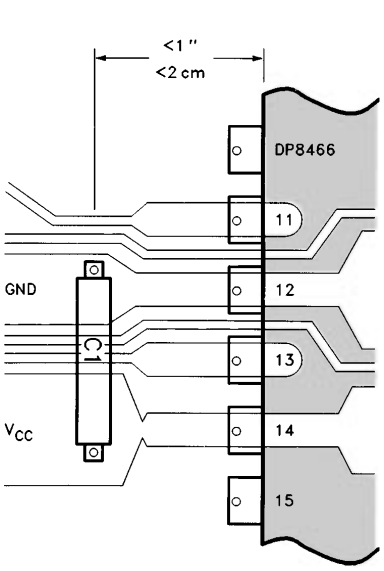


FIGURE 6.30. Conceptual Component Placement to Ensure Short PCB Traces between Connector and DDC



TL/F/8663–I5

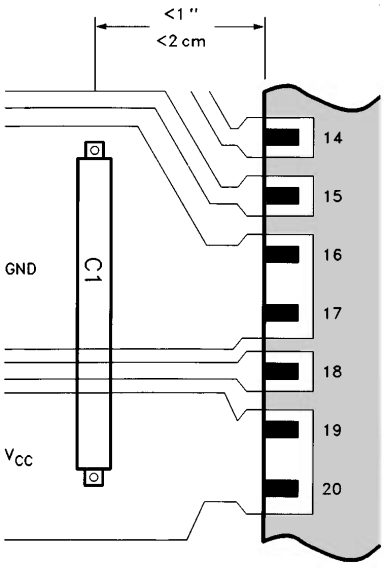
FIGURE 6.31. Proper Buffering of DDC's Address/Data Bus by Latching Address In



TL/F/8663–I6

C1 = 0.1 μF Ceramic

(a) Dual-In-Line Package



TL/F/8663–I7

C1 = 0.1 μF Ceramic

(b) Plastic Chip Carrier

FIGURE 6.32. Typical PC Board Layout for Decoupling the Power Supply (X-Ray View of Foil Side from Component Side)

Chapter 7 DDC Functional Operations

7.0 INTRODUCTION

In this chapter, all the main DDC functional operations such as Disk Formatting, DMA Data transfers, Error detection and Correction, and basic Disk Read/Write are discussed in detail from software point of view. Also, the power-up, chip initialization procedure, and interrupt servicing is described.

7.1 OPERATING MODES OF THE DDC

The DDC can be thought of as operating in one of the four modes; Reset, Command Accept, Command Perform or Error. *Figure 7.1* shows a flow chart for these modes.

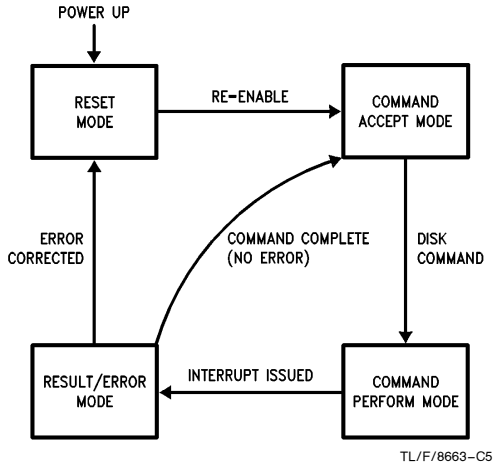


FIGURE 7.1. DDC Operating Modes

RESET MODE:

DDC is put in the reset mode after power-up or prior to starting a new operation if the previous operation was aborted. After the DDC has been reset and the DDC has been re-enabled, it moves to the Command Accept mode.

COMMAND ACCEPT MODE:

The DDC is free and ready to receive a command. Various command and pattern registers and counters may be loaded to perform a disk operation, such as format, read or write etc.

COMMAND PERFORM MODE:

In this mode, the DDC executes the disk command that was loaded into it in the Command Accept mode. It carries out DMA operations. On a successful or unsuccessful completion of the operation, the DDC will generate an interrupt (if the interrupts were enabled, EI bit in OC register), and enters the Result/Error mode.

Note: If interrupts were not enabled, then the Status register should be polled in order to find out the result.

RESULT/ERROR MODE:

In this mode, the Status and Error registers should be read to find out the result of the operation or the type of error that occurred. If the operation was completed successfully, the DDC will go back to Command Accept mode. If an error occurred during the execution of the command, the DDC will abort the operation and the Error register will indicate the type of error that occurred. To perform the operation again or to correct the error, the DDC must be reset and loaded for a particular operation, hence it goes back to the reset mode.

7.2 INITIALIZATION

After the DDC is hooked up in the system, it can be powered up and initialized to perform the desired disk operation. The chip power-up reset, operation initialization, and register programming are discussed in the following paragraphs. A flow chart shown in *Figure 7.2* describes a basic algorithm for a power-up, reset and initialization procedure.

7.2.1 Power-up and Reset

After the chip power-up, the DDC must be held reset for a duration of at least 4 BCLK and 32 RCLK periods (with these clocks active) before it could be assigned a disk format or it could be set for any disk operation. The DDC can be reset by asserting the RESET pin low or by setting the internal RES bit in the OC register high. After the DDC has been reset (for the time indicated above), The external RESET pin must be deasserted and internal RES bit in OC register must be cleared. When the system is powered, the DDC should be reset immediately, even if it will not be used right away. This is because its internal sequencers may be randomly powered on into a state that could draw some excessive I_{CC} currents.

7.2.2 Disk Operation Initialization

After a reset, the DDC must be re-enabled by setting RED bit high in the Drive Command (DC) register before other registers are loaded for a particular operation. Once the DDC is reenabled, it is ready to perform a disk operation such as read, write, or format. Various parameter and command registers and counters can then be loaded depending on the type of operation. In most of the operations, the Drive Command (DC) register is loaded the last except when the DDC is configured in a Non-Tracking DMA mode.

7.2.3 Register Programming

In order to perform an operation, related registers can be loaded in any order keeping the following restrictions in mind.

- The Drive Command (DC) register must be the last register to be loaded for any DDC operation. There is one exception to this. In non-tracking DMA mode, a remote DMA operation may be initiated by loading the operation command (OC) register after a disk command has been started.
- If the on-chip DMA is being used, or if the Remote Data Byte Count registers will be read back, the Local and Remote Transfer registers must be loaded before the Sector Byte Count and Remote Data Byte Count registers are loaded.
- The Number of Sector Operations (NSO) counter must be loaded after an external RESET or internal RESET (in OC register) are both inactive. Other registers can be loaded while reset is active.

- During the execution of an operation (format, read or write), the pattern and count registers **must not** be read. Reading these registers will interfere with the DDC's operation, and could cause some bizarre results. These registers may be written to at anytime. When data is written it takes effect immediately. Thus the only caution is to not change a pattern or count register randomly as the system may not know whether the change will apply to the current sector or the next. The Interlock mode with the Header Complete interrupt should be used to synchronize register updates.

7.3 DISK FORMATTING

The DDC can be programmed to format a disk with any type of sector format. A versatile and flexible sector format with various formatting techniques are incorporated in the DDC. Various formatting features, methods and the DDC sector format options are discussed below.

7.3.1 Key Features

MFM/NRZ Data:

The DDC can be programmed to output MFM or NRZ data while writing to the disk (bit MFM in the Disk Format register). In case of MFM data, some external circuitry will be required, as indicated in section 6.2.

Hard/Soft Sectored Drives:

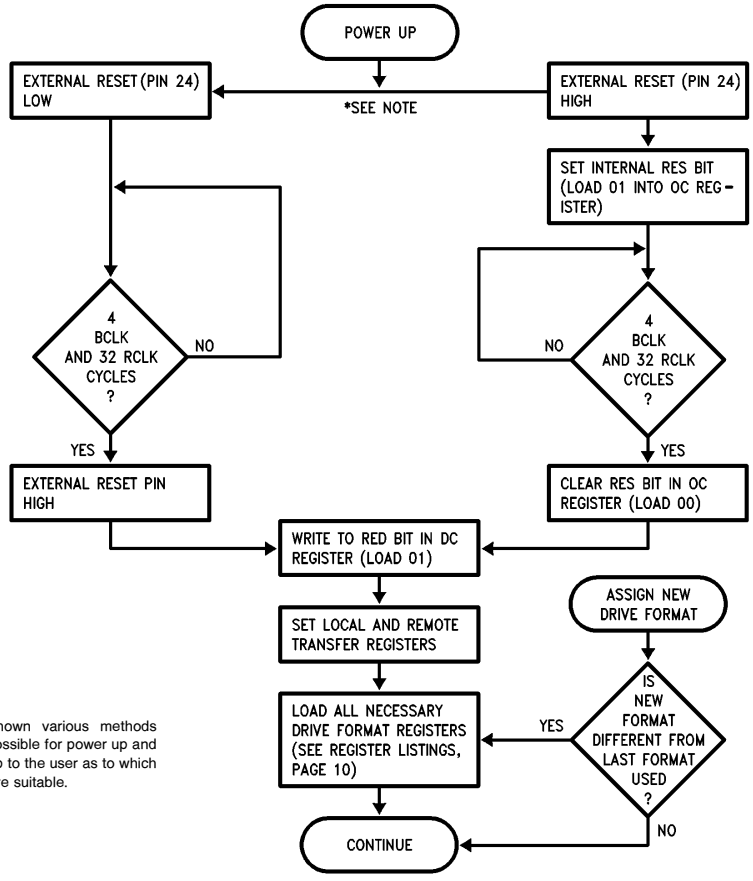
The DDC can be interfaced to both hard and soft sectored drives. This is done through HSS bit in Disk Format register. See section 6.2 and 7.3.3 for implementation of various hard and soft sector formats with the DDC.

7.3.2 Sector Format Options

The DDC offers a versatile sector format which can accommodate most of the currently used disk formats. The DDC sector format options are shown in *Figure 7.3* and the associated registers to program various format fields are shown in *Figure 7.4*. Various ID and Data fields are described below. Discussion on implementation of the popular disk formats using the DDC is given in the next section.

ID And Data Preamble:

There are two fields provided for ID and Data Preambles, which are required in hard and soft sector formats. The ID Preamble field can also be used as an Address Mark (field of no transitions) in case of formats with sector mark (ESDI or SMD type formats). Up to 31 bytes each for ID and Data preambles are allowed. These fields can be programmed using ID and Data Preamble Pattern and Byte Count registers (addresses 31H and 21H for ID, and addresses 3DH and 2DH for the Data).



Note: As shown various methods are possible for power up and it is up to the user as to which is more suitable.

TL/F/8663–C6

FIGURE 7.2. Power-up and Initialization Algorithm

ID And Data Synch #1:

This field will normally be used for writing an ID and Data Address Mark in case of soft sector drives. For hard sector drives this field may either be skipped or be used to extend the ID Preamble or ID Synch #2 fields. Up to 31 bytes can be written in this field for both ID and Data using ID Synch #1 Pattern and Byte Count registers, addresses 32H and 22H for ID, and addresses 3EH and 2EH for data.

ID And Data Synch #2:

The ID and Data Synch #2 fields, used by the DDC for byte alignment, can also have up to 31 bytes each for ID and Data. These fields can be programmed using ID and Data Synch #2 Pattern and Byte Count registers (addresses 33H and 23H for ID and addresses 3FH and 2FH for data).

HEADER BYTES: Header bytes are used to specify the header information of a particular sector such as sector number, cylinder number, track number etc. At least 2 and maximum 6 bytes can be written using 6 Header Byte Pattern and associated control registers (addresses 14H, 15H, 16H, 17H, 18H, 19H and 24H, 25H, 26H, 27H, 28H, 29H respectively). See description of Header Byte Control register in Chapter 5.

ID and Data CRC/ECC:

The DDC can be programmed for a 2 bytes of internal CRC or up to 6 bytes of internal ECC appendage using the Disk Format register, for both ID and Data fields. The CRC appendage is internal to the DDC and no pattern or count

ID FIELD

ID PREAMBLE	ID SYNCH #1 (AM)	ID SYNCH #2	HEADER BYTES	ID CRC/ECC	ID EXT ECC	ID POSTAMBLE
0–31 Bytes	0–31 Bytes	0–31 Bytes	2–6 Bytes	0, 2, 4 or 6 Bytes	0–31 Bytes	0–31 Bytes

DATA FIELD

DATA PREAMBLE	DATA SYNCH #1 (AM)	DATA SYNCH #2	DATA FORMAT PATTERN	DATA CRC/ECC	DATA EXT ECC	DATA POSTAMBLE	GAP 3
0–31 Bytes	0–31 Bytes	0–31 Bytes	1–64K Bytes	0, 2, 4 or 6 Bytes	0–31 Bytes	0–31 Bytes	0–255 Bytes

FIGURE 7.3. Sector Format Options

Pattern Register	Hex Addr	Pattern Source	Control Function	Hex Addr	Control Register
ID Preamble	31	Internal	Repeat 0-31 Bytes	21	ID Preamble Byte Count
ID Synch #1 (AM)	32			22	ID Synch #1 (AM) Byte Count
ID Synch #2	33			23	ID Synch #2 Byte Count
Header Byte 0	14		Define/Control	24	Header Byte 0 Control
Header Byte 1	15			25	Header Byte 1 Control
Header Byte 2	16			26	Header Byte 2 Control
Header Byte 3	17			27	Header Byte 3 Control
Header Byte 4	18			28	Header Byte 4 Control
Header Byte 5	19			29	Header Byte 5 Control
ID CRC/ECC	* *		16-BIT CRC/ 32-BIT / 48-BIT ECC	35	Disk Format (DF) PPB0-5 / PTB0-5
ID External ECC	*	External	0-31 Bytes	2B	ID External ECC Counter
ID Postamble	3C	Internal	Repeat 0-31 Bytes	2C	ID Postamble Byte Count
Data Preamble	3D			2D	Data Preamble Byte Count
Data Synch #1 (AM)	3E			2E	Data Synch #1 (AM) Byte Count
Data Synch #2	3F			2F	Data Synch #2 Byte Count
Data Format	3B		Field Size 1-6k Bytes	38	Sector Byte Count 0
				39	Sector Byte Count 1
Data CRC/ECC	* *		16-BIT/ 32-BIT / 48-BIT ECC	35	Disk Format
Data External ECC	*	External	0-31 Bytes	2A	Data External ECC Counter
Data Postamble	30	Internal	Repeat 0-31 Bytes	20	Data Postamble Byte Count
Gap	3A		Repeat 0-255 Bytes	34	Gap Byte Count

*These are not pattern registers.

**CRC polynomial is built into the DDC and does not require any registers. ECC requires PPB0-5 and PTB0-5 registers.

FIGURE 7.4. Pattern Registers/Counters for Sector Format Fields

register is used besides the disk format register. In case of an ECC appendage, the Polynomial Preset and Tap Byte (0–5) registers (addresses 2H–DH) must be used. Also in case of ID CRC/ECC appendage, if an internal CRC/ECC appendage is not to be used then an external ECC must be used. An external ECC is not necessary if an internal CRC/ECC was not used for a Data CRC/ECC appendage.

ID and Data External ECC:

An external ECC may be appended to encapsulate and internal CRC or ECC, for diagnostic purposes, using an external ECC circuitry. Up to 31 bytes can be appended using ID and Data Ext. Byte Count registers (addresses 2BH and 2AH).

ID and Data Postamble:

Up to 31 bytes can be used for ID and Data postamble using ID and Data Postamble Pattern and Byte Count registers, addresses 3CH and 2CH (for ID) and addresses 30H and 20H (for data).

Data Format Pattern:

Data format is programmed by the Data Pattern register (address 3B) and then can be repeated up to 64K times through the Sector Byte Count registers (addresses 38H and 39H). The number of times data format is repeated depends on the sector size.

Gap 3:

Up to 255 bytes can be written using Gap Pattern and Byte Count registers (address 3AH and 34H). In soft sectored drive operation, the Gap 3 bytes are written for each sector (determined by Gap 3 Byte Count register) except the last sector. For the last sector, gap bytes will be written until an Index pulse is received. In case of hard sectored drives, the Gap 3 Byte Count register is only used while formatting the disk. In normal disk write operation, the DDC writes gap bytes until a Sector pulse is received, ignoring the contents of the Gap byte count register.

Considerations for Pattern and Count Register Programming for Format Operations

- If any Byte Count register is loaded with zero, that field will be excluded and no pattern for the corresponding Pattern register needs to be loaded. Similarly if any of six Header Byte Control registers is set with all bits equal to zero, no pattern for that byte needs to be loaded.
- Maximum two consecutive fields can be excluded from a sector format. This includes the six header bytes which may be thought of as six fields.
- Format operations always start with an index pulse and end with the next index pulse, thus making one track. The DDC can only be programmed to format one track at a time.

7.3.3 Implementing Some Popular Sector Formats

There are three general sector formats which are commonly used in various disk systems; soft sector format (Floppy, ST506 type formats), hard sector format (hard or fixed formats used in ESDI/SMD type drives), and format with sector mark (soft or variable formats used in ESDI/SMD type drives). These three types are shown in *Figure 7.5* and their implementation using the DDC is discussed below.

SOFT SECTOR (FLOPPY/ST506 TYPE) FORMATS

The field shown in *Figure 7.5 (a)* is the most commonly used sector format used in soft sectored drives such as Floppy and ST506/412/419 type winchester drives. A double density floppy format recommended by the IBM and a Seagate's ST506/412/419 type format are shown in *Figure 7.6* as examples. It can be seen from *Figure 7.5* that the fields used in these formats are in direct correspondance with the ones supported by the DDC except the Post Index Gap in floppy format or Gap 1 in ST506 format. The Post Index Gap field is not supported by the DDC and may be eliminated as part of the disk format. If it must be generated external hardware may be added as discussed in section 6.2. The implementation of rest of the format is very straightforward and can be achieved using the registers shown in *Figure 7.4*. Implementation of the ST506/412/419 type format is shown in *Figure 7.7* and Table 7.1.

TABLE 7.1. Implementation of ST506 Formats

ST506			DDC		
Field	**Pattern**	**Byte Count**	**Field**	**Pattern Register Address**	**Byte Count Register Address**
Gap 1	4E	16	*	*	*
SYNC	00	13	ID Preamble	31H	21H
ID AM	A1, FE	2	ID Synch #1	32H	22H
CYL	#	2	Header Byte	14H–5H	24H–5H
HD	#	1	Header Byte	7H	7H
SEC	#	1	Header Byte	19H	29H
CRC	*	2	ID CRC/ECC	**	**
Gap 2	00	3	ID Postamble	3CH	2CH
Gap 2	00	13	Data Preamble	3DH	2DH
Data AM	A1, F8	2	Data Synch #1	3EH	2EH
Data	— —	256	Data Format	3BH	38H, 39H
CRC	****	2	Data CRC/ECC	****	****
Gap 3	00	3	Data Postamble	30H	20H
Gap 3	4E	15	Gap	3AH	34H
Gap 4	4E	352	Gap **	3AH	34H

*Gap 1, Post Index Gap, is not supported by the DDC. See section 6.2 and the hard sector format section in this chapter.
**The CRC is internal to the DDC and Disk Format register is used to select it.
***The DDC only allows 256 bytes for the Gap field. During a format the gap for the last sector of the track will be written until the pulse is received.

TABLE 7.2. Implementation of ESDI Hard Sector Format

ESDI FORMAT			DDC REGISTERS		
Field	Pattern (See Note†)	Byte Count (See Note†)	Field	Pattern Register Address	Count Register Address
Inter-Sector Gap	00	‡	*	*	*
Address PLO Sync	00	‡	ID Preamble	31H	21H
(No Field)		0	ID Sync #1	32H	22H
Byte Sync Pattern	—	≥ 1	ID Sync #2	33H	23H
Cylinder	xxxx	2	Header #1/2	14/15H	24/25H
Head	xx	1	Header #3	16H	26H
Sector	xx	1	Header #4	17H	27H
Flag/Status	xx	1	Header #5	18H	28H
(No Field)		0	Header #6	19H	29H
Address Check Bytes	**	**	ID CRC/ECC	**	**
Address Pad	00	—	ID Postamble	3CH	2CH
Write Splice	00	1	***	***	***
Data PLO Sync	00	—	Data Preamble	3DH	2DH
(No Field)		0	Data Sync #1	3EH	2FH
Byte Sync Pattern	—	≥ 1	Data Sync #2	3FH	2FH
Data	xx	—	Data Pattern@	3BH	38/39H
Data Check Bytes	**	**	Data CRC/ECC	**	**
Data Pad	00	≥ 2	Data Postamble	30H	20H
Format Tol.	00	—	Gap	3AH	34H
Inter Sector Gap	00	‡	Gap	*	*

†Where dashed entries appear in these columns the ESDI standard does not specifically define these fields but leaves this up to the user.

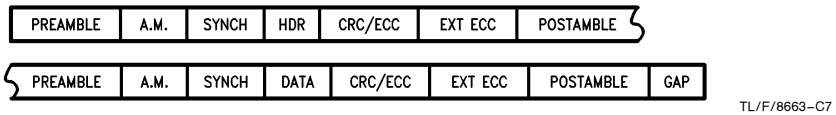
‡Defined by the ESDI specification using a specific formula.

*The standard Inter Sector Gap field is not supported by the DDC. See Hard sector section in this chapter, and section 6.2 for details in generating this field. Chapter 6's hardware actually uses the DDC's gap field to generate the ISG field for the next sector.

**The 16 bit CRC, 32 bit or 48 bit ECC is programmed internally see section 7.6 for details.

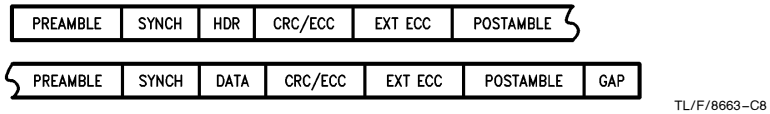
***The write splice field is not supported by the DDC. It should be included as part of the Data preamble field for format and write operations, and part of the ID postamble for read operations.

@Used only to format the data field.



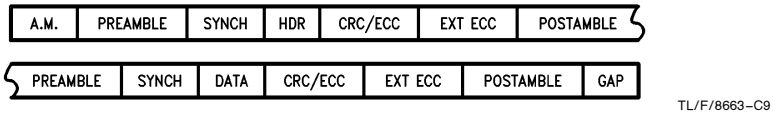
TL/F/8663–C7

(a) Soft Sector Format



TL/F/8663–C8

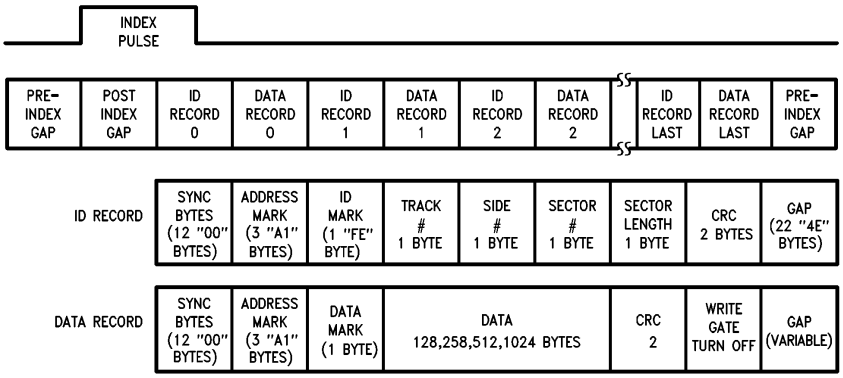
(b) Hard Sector Format



TL/F/8663–C9

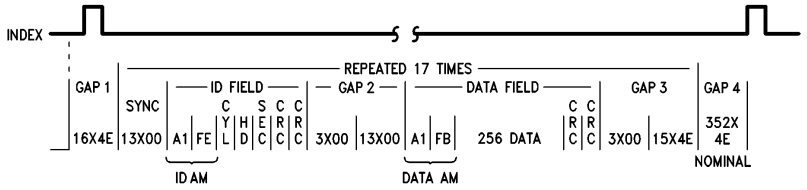
(c) Format with Sector Mark

FIGURE 7.5. Generalized Common Sector Formats



TL/F/8663–D0

(a) Double Density Floppy Format



TL/F/8663–D1

(b) ST506/412/419-type Sector Format Recommended by Seagate

FIGURE 7.6. Soft Sector Formats

HARD SECTOR FORMATS (Setting up Pattern and Count registers for Read/Write and Formatting)

The second sector format shown in *Figure 7.5* is the general hard sector format. In this format the beginning of each sector is marked by a sector pulse. This format is typically used in ESDI and SMD type drives. The format for these types of drives is fairly straight forward, and Table 7.3 shows the DDC registers, patterns and count lengths to perform and ESDI format. Table 7.4 shows Control Data Corp. recommended format for an SMD specification.

In both formats the Inter Sector Gap of ESDI and the Head Scatter Gap, commonly referred to as the Post Index or Post Sector Gap in SMD are not directly supported by the DDC. Additional counters and logic as shown in Section 6.2 previously are needed if these are to be supported. This solution would be appropriate for ESDI or SMD hard sectored drives. The philosophy behind it being that the index/sector pulse from the drive is presented to the DDC delayed by the external logic by the length of the post index/sector gap.

Another field in both SMD and ESDI that is not directly supported is the write splice field. This field is intended for turning off and on the read/write head without interfering with the data preamble. This field is easily generated by including it in the data preamble for formatting. Then during read operations including it in the ID postamble, and for write operations including it in the Data preamble.

For ESDI drives during the format operation the standard calls out for optionally deasserting Write Gate for two bit times during the write splice. This option is useful if the drive is performing some data encoding (such as 2–7). Pulsing Write Gate for two bit times informs the encoder/decoder on the drive to start the data preamble field. The DDC does not directly support this deassertion of Write Gate, but can format the drive easily anyway, by performing a two pass format operation. For the first pass the DDC will be set up to format the whole track, however only the ID fields will actually be formatted due to the encoder. The second pass should be a multi-sector write data operation, which will format the data fields.

FORMATS WITH SECTOR MARKS (Setting up Pattern and Count registers for Read/Write and Formatting)

The third method of formatting a disk is with sector marks. A simplified typical format is shown in *Figure 7.5(c)*. This format is most common with soft sectored ESDI, and a few SMD drives. Basically this format uses a field of no flux transitions and some special drive hardware to enable the drive to generate a sector pulse-like signal called Address Mark. The Address Mark signal thus signifies the start of a new sector.

To implement this using the DDC, requires some manipulation of the format parameter RAM in conjunction with the use of the ESDI control PAL hardware described in section 6.2. Other external hardware designs may require altering the format parameters, however, much of the discussion is still applicable. While formatting, the ID preamble field of the DDC is set with the pattern of the Post index gap. The count length of this field is set to the length of the gap plus length of address mark (usually 3 bytes). The ID synch #1 field contains the pattern of the preamble while its length is increased by one to accommodate the Address Mark pad field. This ID Sync #2 contains the byte synch pattern.

The SAM bit in the Disk Format register is set. Hence when formatting is initiated, the DDC generates AME and the start of its preamble. This is taken by the PAL and external hardware, and delayed by the length of the gap (which is being written), to when the address mark (3 bytes of no transitions) is written. This is then followed normally by the preamble and synch fields.

TABLE 7.3. Implementation of SMD Hard Sector Format

SMD FORMAT			DDC REGISTERS		
Field	**Pattern**	**Byte Count**	**Field**	**Pattern Register Address**	**Count Register Address**
Head Scatter Bytes	00	16	*	*	*
Address PLO Sync	00	11	ID Preamble	31H	21H
(No Field)		0	ID Sync #1	32H	22H
Byte Sync Pattern	—	1	ID Sync #2	33H	23H
Flag/Status	—	1	Header #5	14H	24H
Cylinder	—	2	Header #2/3	15/16H	25/26H
Head	—	1	Header #3	17H	27H
Sector	—	1	Header #4	18H	28H
(No Field)		0	Header #6	19H	29H
Address Check Bytes	CRC/ECC	2–6	ID CRC/ECC	**	**
(No Field)		0	ID Postamble	3CH	2CH
Write Splice	00	1	***	***	***
Data PLO Sync	00	11	Data Preamble	3DH	2DH
(No Field)		0	Data Sync #1	3EH	2FH
Byte Sync Pattern	—	1	Data Sync #2	3FH	2FH
Data	xx	—	Data Pattern@	3BH	38/39H
Data Check Bytes	CRC/ECC	2–6	Data CRC/ECC	**	**
EOR Pad	—	1	Data Postamble	30H	20H
End of Sector	—	10	Gap	3AH	34H

†Where dashed entries appear in these columns the SMD standard does not specifically define these fields but leaves this up to the user.

‡The Data Pattern is used during format operations only.

*The standard Inter Sector Gap (Head Scatter Bytes) field is not supported by the DDC. See Hard sector section in this chapter, and section 6.2 for details in generating this field. The hardware in chapter 6 uses the DDC's gap filed to generate the Head Scatter Bytes for the next sector.

**The 16 bit CRC, 32 bit or 48 bit ECC is programmed internally see section 7.6 for details.

***The write splice field is not supported by the DDC. It should be included as part of the Data Preamble field for format and write operations, and part of the ID postamble for read operations.

@Used only to format the data field.

When reading from the disk the DDC looks on the drive like a hard sectored one. The Address Mark is detected, and the drive asserts AMF which is decoded by the PAL into a sector pulse for the DDC. Hence the DDC format parameter RAM should be changed to the ID preamble field containing the pattern of the actual preamble and count length. The ID synch #1 field count is set to zero so that this field is skipped and the ID synch #2 contains the byte synch. (The read gate is delayed by 8 bits to accommodate the 1 byte write splice field). This ensures a successful read operation. With the DDC working in this mode it would see an index and a sector pulse from the sector 0 AMF separated by the Post Index Gap length. The DDC takes the index to be the sector 0 pulse also. The ESDI control PAL of chapter 6 takes care of delaying the index pulse over the sector 0 AMF pulse and suppresses the sector pulse to the DDC from the sector 0 AMF.

If a Write Header operation is desired at any time then it would be possible to do so only if a regular format operation had been done earlier. The AMF from the drive generates a sector pulse to the DDC to start writing the header. Also the Read Gate assertion is delayed externally by a byte to accommodate the Write Splice associated with Write Gate assertion. The ESDI field names byte values and lengths along with the DDC's equivalent registers are shown in Table 7.5.

The DDC does not directly support the ESDI write splice field as before, however, this is easily remedied by including this byte as part of the postamble during formats and write operations as part of the data preamble during read operations. This ensures that reading of the write splice is avoided which is the purpose of this field.

As in the hard sector format, ESDI has an optional deassertion of the Write Gate in between the ID and Data fields. The DDC does not support this option, but by performing a two pass format as described above a complete format can be accomplished.

MODIFICATIONS TO SECTOR FORMATS

While the previously discussed ''standard'' formats are a useful starting point, they are generally not strictly adhered to when actually formatting a disk. This is due usually to the users desire to optimize the drive for various parameters. These include optimizing data integrity, data separator per-

TABLE 7.4. Implementation of ESDI Sector Mark Format

ESDI FORMAT			DDC REGISTERS		
Field	Pattern (See Note†)	Byte Count	Field	Pattern Register Address	Count Register Address
Inter-Sector Gap	00	‡	*	*	*
Pad	00	1	ID Preamble	31H	21H
Address PLO Sync	00	—			
Address Mark	†††	3	ID Sync #1	32H	22H
Byte Sync Pattern	—	≥ 1	ID Sync #2	33H	23H
Cylinder	xxxx	2	Header #1/2	14/15H	24/25H
Head	xx	1	Header #3	16H	26H
Sector	xx	1	Header #4	17H	27H
Flag/Status	xx	1	Header #5	18H	28H
(No Field)		0	Header #6	19H	29H
Address Check Bytes	**	**	ID CRC/ECC	**	**
Address Pad	00	≥ 2	ID Postamble	3CH	2CH
Write Splice	00	1	***	***	***
Data PLO Sync	00	—	Data Preamble	3DH	2DH
(No Field)		0	Data Sync #1	3EH	2FH
Byte Sync Pattern	—	≥ 1	Data Sync #2	3FH	2FH
Data	xx	—	Data Pattern@	3BH	38/39H
Data Check Bytes	**	**	Data CRC/ECC	**	**
Format Tol.	00	‡	Gap	*	*
ISG	00	‡	Gap	*	*

†Where dashed entries appear in these columns the ESDI standard does not specifically define these fields but leaves this up to the user.

‡Defined by the ESDI specification using a specific formula.

†††For the Address mark the pattern does not matter, as an area of no flux transitions is recorded.

@The Data Pattern register is used only for format operations.

*The standard Inter Sector Gap field is not supported by the DDC. See ESDI control PAL hardware solution in chapter 6.2 for details in generating this field.

**The 16 bit CRC, 32 bit or 48 bit ECC is programmed internally see section 7.6 for details.

***The write splice field is not supported by the DDC. It should be included as part of the data preamble field for format and write operations, and part of the ID postamble for read operations.

formance, access speed, sector defect sparing algorithms, and total storage capacity. Some of these considerations are discussed below.

Error Detection/Correction And Data Field Length

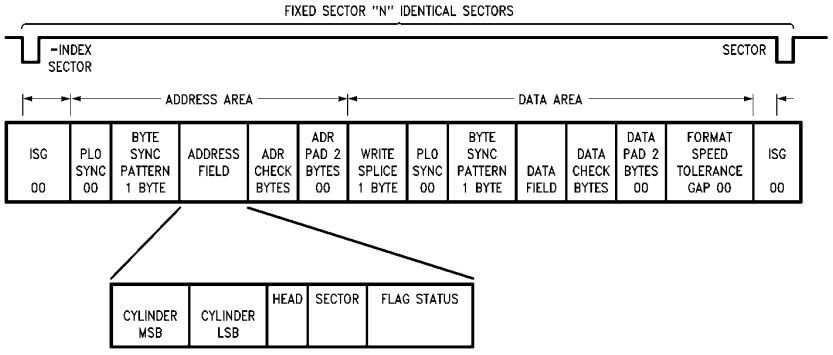
Generally, ST506 type drive recommended formats utilize a 16 bit CRC, however, this generally does not offer the type of data integrity that is needed in the data field. Thus generally ST506 type drives should utilize a 32 ECC for data. The ID field usually can be a CRC check field since the header is so short, and since recovery of the header address can be achieved by reading the previous sector header.

In general the type of ECC field to choose for the data field depends on the length of the data field, the defects on the disk media itself and the tolerance the designer places on his system for detecting errors, and the probability limits for miscorrection.

The choice for the length of the data field is determined in part by the type of ECC chosen (or vice versa), but also by system performance criteria. For a given media, a longer data field, and fewer sectors per track can maximize total storage capacity, but it requires better ECC. In many cases where the disk is to store many small files that leave many partially filled sectors, a larger sector size will waste disk space. However, if a few very large files are stored when large data field maximize storage. Thus usually a good tradeoff is to have sector sizes between 256 to 1024 bytes.

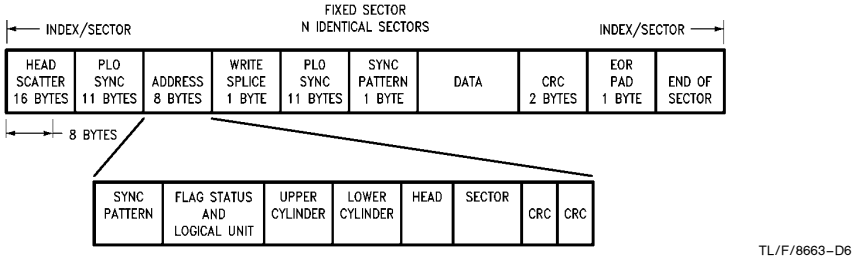
ID And Data Preamble

These field lengths are used to lock the data separator to incoming data during a read operation. The lengths of these fields determine the lock time requirements of the data separator. For individual ESDI and SMD drives, where the data separator is on the drive, the preamble should follow the manufacturers recommendations. For ST506 drives, and drives with imbedded controllers, the data separator is part of the controller and the preamble length should be set based on the separators performance goals.



TL/F/8663–D5

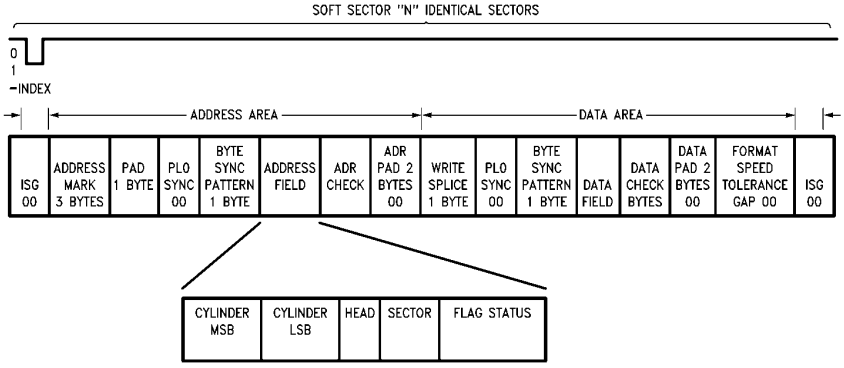
(a) An EDSI Hard Sector Format Recommended by Maxtor



TL/F/8663–D6

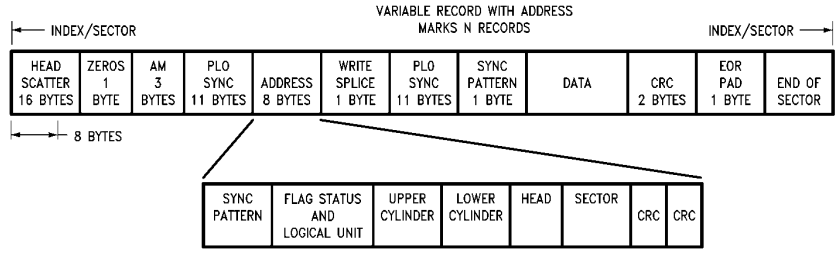
(b) An SMD Hard Sector Format Recommended by CDC

FIGURE 7.7 Hard Sector Formats



TL/F/8663–E0

(a) An ESDI Soft Sector Format Recommended by the Maxtor



TL/F/8663–E1

(b) An SMD Soft Sector Format Recommended by the CDC

FIGURE 7.8. Formats with Sector Mark

7.3.4 Formatting Methods

The disk formatting can be carried out using one of the three methods supported by the DDC; Internal Sequential, FIFO Table and Interlock Mode. These three disk formatting methods are explained in the following paragraphs and also summarized in a flow chart in *Figure 7.9.*

1. INTERNAL SEQUENTIAL METHOD

This method of disk formatting is adopted when sectors are to be physically contiguous. The DDC can be set for a multi-sector operation to format a whole track of sequential sectors. The steps required to perform the Internal Sequential method are explained in the following paragraphs. See *Figure 7.9*.

The DDC in Command Accept Mode

Step 1. All the Pattern and Byte Count registers for various sector format fields are loaded. All the Header byte's Pattern and Control registers (such as the one for cylinder number, head number etc.) are loaded except the one containing sector number information. The Pattern register for this header byte need not be loaded with anything, but the Control register should be loaded with 3H (i.e. SSC = 1 and HB = 1). With SSC = 1, the sector number for each sector will be loaded into this Header Byte Pattern register, automatically, from the Sector Counter.

Step 2. The Sector Counter is loaded with the first sector to be formatted. The contents of Sector Counter are loaded into the Header Byte Pattern register reserved for sector number, written to the disk, and then incremented for the next sector. The Number of Sector Operation (NSO) Counter is loaded with the number of sectors per track.

Step 3. The Disk Format register is loaded with FTF = 0, desired internal CRC/ECC appendage and other information (such as Hard/soft sector, NRZ/MFM data, etc.). The ECC/CRC control register (address 0EH) should also be loaded for desired options, such as inverting the serial data. The ECC polynomial and tap registers should be programmed if ECC is chosen.

Step 4. The Operation Command register is loaded to enable interrupts. Finally the Drive Command register is loaded with the Format Track command (i.e. ACH). Refer to Table 5.6 for various DDC commands.

The DDC in Command Perform Mode

Step 5. The DDC will start the operation when it receives an index pulse indicating the start of a track and will end the operation when it encounters another index pulse.

The DDC in Result/Error Mode

Step 6. An interrupt will be generated after a successful or unsuccessful execution of Format Track command, if the interrupts were enabled. The Status and (or) Error registers will indicate the result or the type of error occurred. In case of successful completion of formatting, the DDC could be initialized to format the next track and steps 1 thru 5 will be repeated. If an error has occurred, the interrupt should be serviced properly and the DDC should be reset. Refer to section 7.7 for interrupt servicing, and section 7.2.2 for resetting.

2. FIFO TABLE METHOD

This method is ideal if sector interleaving is required. The sectors may be written to the disk in any order using this method which offers the minimum amount of microprocessor involvement during the format operation. In this method the header bytes are written on the disk from the memory (via FIFO) instead of the Header Byte registers. This essentially eliminates the need of the microprocessor to update header bytes for each sector as could be done in the interlock mode. All other format pattern and count registers are loaded once by the microprocessor remain valid for the entire operation. The header bytes for each sector (with or without interleaving) are set up contiguously in sets as a table in the memory and then read by the DDC, one set for each sector, from the memory using the local DMA channel.

The steps required to perform the FIFO table method are explained below. Also see *Figure 7.9*.

Step 1. Sets of header bytes (one for each sector) for the entire track are stored contiguously in a memory area accessible to the local DMA. Each header byte set must contain an even number of bytes and start on an even byte boundary. If the header byte set contains an odd number of bytes, an extra dummy byte must be inserted at the end of each set so that each header byte set will start on an even byte boundry, for DMA considerations.

The DDC in Command Accept Mode

Step 2. Address of the first byte of the first header byte set is loaded in the DMA Address Byte (0, 1) registers.

Step 3. Sector Counter (SC), Number of Sector Operations (NSO) counter and Header Byte Count registers are loaded with initial sector number, number of sectors to be operated on, and number of header bytes (2–6 bytes), respectively.

Step 4. All other Format patterns and count registers (such as Preamble, Postamble, ECC/CRC etc.) according to the selected sector format are loaded with appropriate information. See *Figure 7.4*.

Step 5. The Desk Format (DF) register is loaded with the FTF bit set. The DF register is also loaded with MFM/NRZ, hard sector/soft sector, and ID, Data CRC/ECC appendage etc.

Step 6. The Operation Command (OC) register is loaded to enable interrupts. Finally, the Drive Command register is loaded with Format track command (ACH). See Table 5.6 for the DDC commands.

The DDC in Command Perform Mode

Step 7. The DDC starts the operation when it receives the index pulse and ends it on the occurrence of next index pulse. As the header bytes are needed they are DMA'd from memory. After a successful or unsuccessful completion of the operation, the DDC will generate an interrupt.

The DDC in Result/Error Mode

Step 8. The Status and (or) Error registers are read to find out the cause of interrupt. In case of a successful completion of the operation, steps 1 thru 7 may be repeated or the DDC may be initialized for another disk operation. In case of an error interrupt, the interrupt is serviced properly. See section 7.2.2 and 7.7.

3. INTERLOCK METHOD

This method is the most versatile of the three disk formatting methods. But it requires fast microprocessor involvement. This method may be used to format a whole track of interleaved sectors or a single sector. Using this method to reformat a single sector is ideal. Formatting single sectors is useful for remapping bad sectors. This method can also be used for creating tracks with varying sector field lengths.

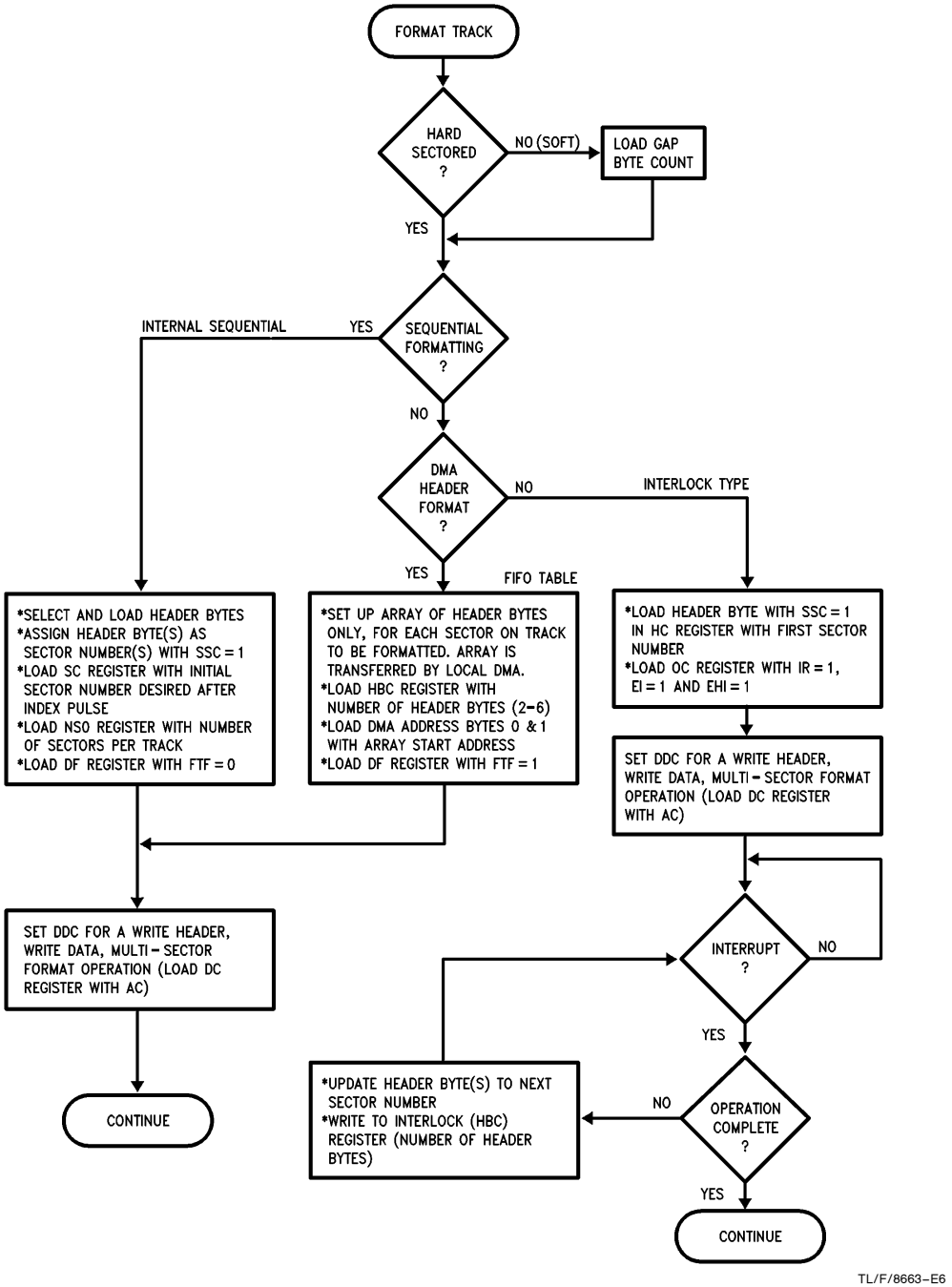


FIGURE 7.9. Track Formatting Methods

The Interlock type disk formatting uses the interlock mode and header complete interrupt to enable the microprocessor to directly update any format parameter bytes. In a write header-write data operation, after the header bytes are written to the disk, the DDC issues the header complete interrupt. With interlock mode set and header complete interrupt issued, the controlling microprocessor has the time (until the preamble field of the next sector) to read status, load the next sector's header bytes, and write the interlock (HBC) register. Writing to the interlock (HBC) register confirms that microprocessor has updated all the required parameters. This must be done following each header match complete interrupt for every sector, including the last sector of the operation. If this is not done, the DDC assumes that the microprocessor did not properly update the parameter registers in time for the next sector and therefore generates an interrupt indicating the Late Interlock error when a subsequent command is loaded in the DC register. A 'step by step' procedure of the Interlock type formatting is explained below. Also see *Figure 7.9*.

DDC in Command Accept Mode

Step 1. All the format registers (shown in *Figure 7.4*) are loaded with respective pattern and count values. The header byte control register for the sector number is loaded with SSC = 1. This is done only for the first sector. Later on, depending upon the interleave factor, the header byte reserved byte for sector number may be loaded with a new value and the associated control register with SSC = 0.

Step 2. The Sector Counter is loaded with the sector number to start with and NSO (number or sector operations) Counter with number of sectors to be formatted.

Step 3. In Disk Format register, FTF is set to zero and other relevent bits such as HSS, MFM, IH's and ID's are also set or reset.

Step 4. The DDC is set to be in Interlock Mode by setting IR = 1 in the Operation Command register. Also header complete and other interrupts are enabled by setting EHI = 1 and EI = 1 in the same register.

Step 5. Finally, the Drive Command register is loaded with the Format Track command, ACH. See Table 5.6 for various DDC commands.

DDC in Command Perform Mode

Step 6. The format operation starts when the DDC receives an index pulse. An interrupt is expected at the completion of the Write Header operation. When this interrupt is received it is tested for Command Complete Error or Header Complete Status Bits set. If it's a Header Complete, then parameter, count and control registers are updated. The Interlock register is written to. This is repeated until the Command Complete or Error Interrupt occurs.

DDC in Result/Error Mode

Step 7. On the occurrence of an interrupt, the Status register is read. If the interrupt was an operation complete interrupt then steps 1 through 5 are repeated for the rest of the sectors to be formatted without changing the contents of the NSO counter. In case of an error, the interrupt is serviced properly. See section 7.7 on interrupts.

7.4 READ AND WRITE OPERATIONS

Once the disk has been formatted, various disk read and write operations can be performed. These commands are listed in Table 7.5 and are discussed briefly in section 5.2.7.

Generally, the read operation is taken as reading data from the disk and can therefore be performed by executing the Read Sector (single or multi-sector) and Read Track commands. Similarly, write operation is considered as writing data to the disk and hence could be achieved by executing the Write Sector (single or multi-sector) and Write Track commands. Other read and write commands imply reading or writing ID with or without data from (to) the disk. The DDC programming procedure for all the read and write commands basically is the same.

A general register programming procedure to perform read and write operations is given below.

Figure 7.10 shows a generalized flow chart for performing a read operation. To generalize both multi-sector and single sector operations, the continue block would go to the next sector operation if multi-sector, and would go to other μP tasks if single sector. Refer also to *Table 7.5* which shows the command codes for the various operations.

Command Name		Op Code	
Read Single Sector	RDSS	11010010	D2H
Read Sector ID	RDID	01110010	72H
Read Multi-Sector Logical	RDMS	11010110	D6H
Read Track	RDTK	11010100	D4H*
Read Track Blind	RDTB	11000100	C4H*
Read ID Multi-Sector	RDIM	01110100	74H
Read Track Data/ID	RDDI	11110100	F4H*
Write Single Sector	WRSS	10010010	92H
Write Multi-Sector Logical	WRMS	10010110	96H
Write Track	WRTK	10000100	84H*
Format Track	FMTK	10101100	ACH
Format Track No Gap	FMNG	10100100	A4H
Find ID	FNID	01010010	52H
Find ID Multi-Sector	FNMS	01010110	54H
Recover Header	RCID	01100010	62H
Re-Enable Controller	RENB	00000001	01H
No Operation	NOP	00000000	00H

*__Note:__ For an entire track operation, the Number of Sector Operations Counter should be set to the number of sectors per track.

Table 7.5. Common Configurations of the Command Bits

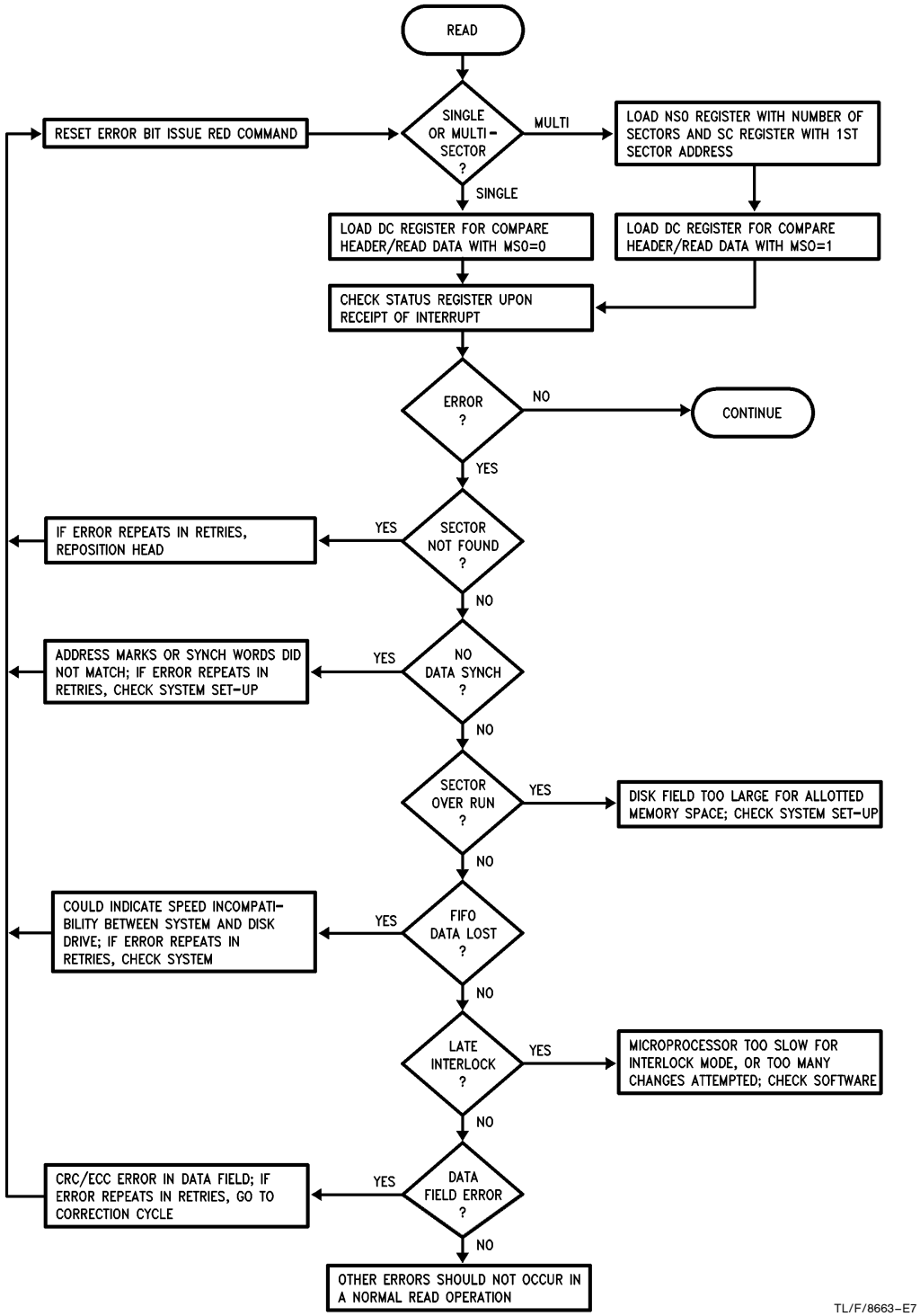
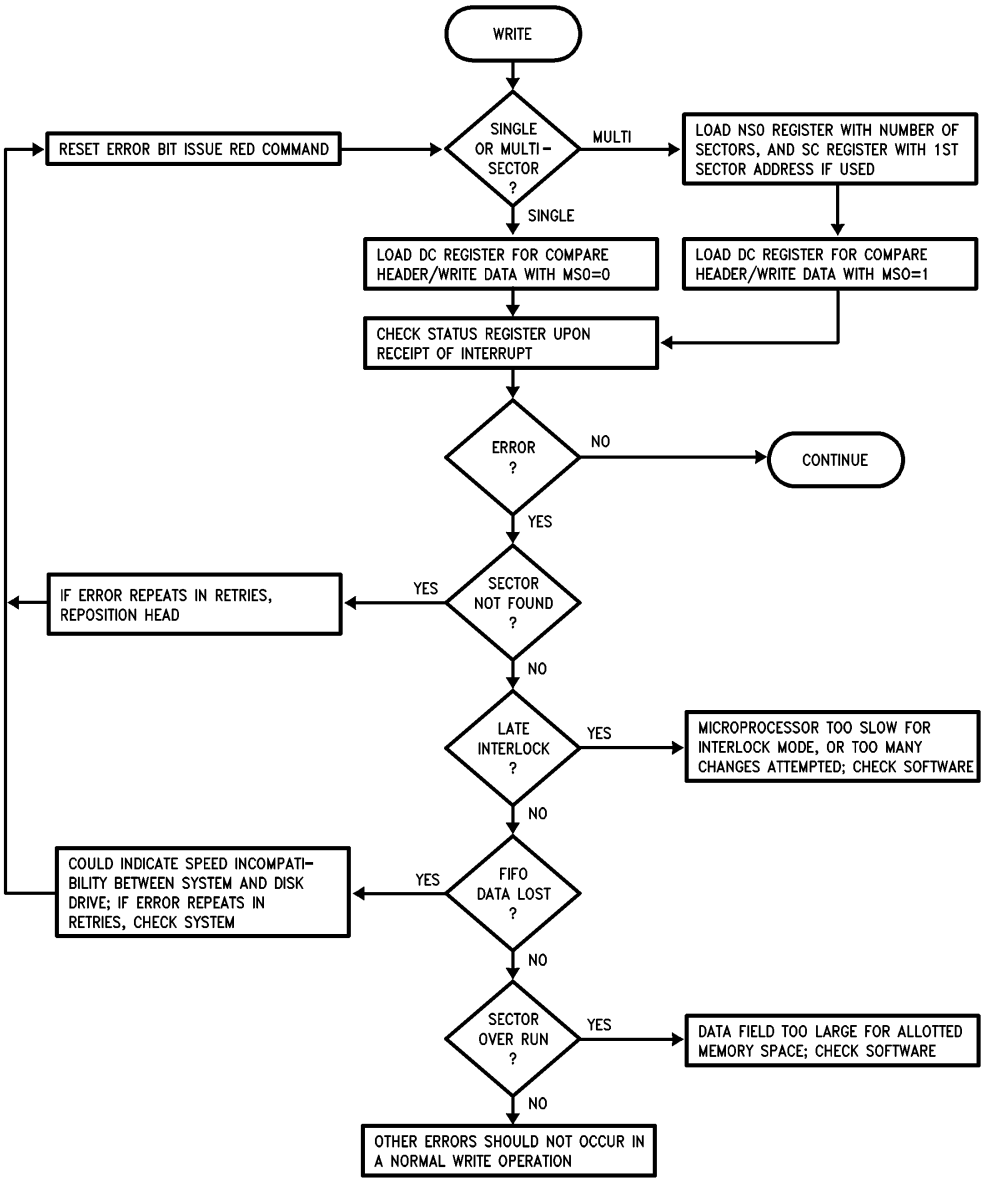


FIGURE 7.10. A Flow Chart for Microprocessor Initiation and Servicing of a Read Operation



TL/F/8663–E8

FIGURE 7.11. A Flow Chart for Microprocessor Initiation and Servicing of a Write Operation

SINGLE SECTOR READ AND WRITE OPERATIONS

This operation involves setting the Disk Command register to perform a Compare Head/Read Data Operation. Only one sector is transferred.

DDC in Command Accept Mode

Step 1. The Parameter and Count registers must be loaded with the proper format information (if not already loaded with the correct information). The Header byte pattern registers must be loaded with the exact header information for the sector to be transferred.

Step 2. The Local DMA channel must be initialized. The Local transfer register should be configured to the desired DMA mode. The DMA address registers are loaded with the data transfer address. If the Remote DMA is used in coordination with the local transfer, especially in tracking mode, the Remote Transfer register should be initialized, as should its address, and length information. Interrupts should be enabled, in order to enable determination of operation completion.

Step 3. Finally the Drive Command register is loaded with the desired Read or Write Command. If a remote DMA operation is to be performed, the Operation Command register should be loaded to start the remote DMA.

DDC in Command Perform Mode

Step 4. The DDC will perform the operation by looking for the correct sector header, then acquiring the system bus, and transferring the appropriate data.

DDC in Result/Error Mode

Step 5. At the end of the command an interrupt is generated. If the interrupt is the Operation Complete interrupt the command terminated properly, and the μP can read this from the Status and Error registers and can proceed to the next command. If an error occurred the microprocessor should service this by either retrying the operation, or correcting the error condition.

MULTI SECTOR LOGICAL READ AND WRITE OPERATIONS

Multi sector logical operations transfer sectors sequentially based on their sector number, whether these sectors are scattered around the disk or not. Multi sector operations use the Number of Sector Operations Counter to determine the number of sectors to transfer. Logical operations are most typically done by using the Sector Counter.

DDC in Command Accept Mode

Step 1. The Parameter and Count registers must be loaded with the proper format information (if not already loaded with the correct information). The Header byte pattern registers must be loaded with the exact header information for the sector to be transferred. The header byte that contains the sector number must have the Sector Counter substituted for it. The Sector Counter is set to the number of the first sector to be transferred. The Number of Sector Operations Counter should be loaded with the number of sectors to be transferred.

Step 2. The Local DMA channel must be initialized. The Local transfer register should be configured to the desired DMA mode. The DMA address registers are loaded with the data transfer address. If the Remote DMA is used in coordination with the local transfer, especially in tracking mode, the Remote Transfer register should be initialized, as should its address, and length information. Interrupts should be enabled, in order to enable determination of operation completion.

Step 3. Finally the Drive Command register is loaded with the desired Logical Read or Write Command. If a remote DMA operation is to be performed, the Operation Command register should be loaded to start the remote DMA.

DDC in Command Perform Mode

Step 4. The DDC will perform the operation locating the first sector, acquiring the system bus, and transferring that sector's data. The Number of Sector Operations Counter is decremented, and the Sector Counter is incremented. Step 4 is repeated until the Number of Sector Operations Counter reaches zero. The command may be terminated early if the DDC could not find one of the correct sectors being sought.

DDC in Result/Error Mode

Step 5. At the end of the command when the NSO counter equals zero, an interrupt is generated. If the interrupt is the Operation Complete interrupt the command terminated properly, and the μP can read this from the Status and Error registers and can proceed to the next command. If an error occurred the microprocessor should service this by either retrying the operation, or correcting it.

MULTI SECTOR PHYSICAL READ AND WRITE OPERATIONS

These operations are very similar to the logical read and write commands, except that rather than looking for the sectors in numerical order, they are read from the disk in the exact order that they pass under the read/write head. Only the differences between these and the logical commands are described.

DDC in Command Accept Mode

The only difference in setting up for the command is that since the basic command will ignore the header bytes for comparison (Ignore Header-Read/Write Data) the Header pattern registers do not need to be updated. If doing a track operation, the command should start on an index pulse by setting the SAIS bit, and the Number of Sector Operations Counter should be loaded with the number of sectors on the track.

DDC in Command Perform Mode

This is the same as previous operation.

DDC in Result/Error Mode

This is the same as the previous operation.

OTHER MULTI-SECTOR PHYSICAL OPERATIONS—INTERLOCK MODE

The host microprocessor can perform many related but different operations on physically consecutive sectors, by using the Interlock mode. Several possible command sequences are briefly outlined:

1. Header Re-writing—If an ID is detected to have an error in its header bytes a sequence of commands can be executed to re-write the header. The rewritten ID may attempt to fix the error or mark the sector as bad. This involves doing two commands in sequence:

a) First find the header of the sector located physically prior to the Bad ID field.

b) Second, do a write header operation to re-write the ID field.

2. Recovery of Data—This is like one above except no attempt is made to correct the ID field, just obtain the data field information.

a) First find the header of the sector located physically prior to the defective sector.

b) Second, do an ignore header operation to The data field is to be recovered, so a read data operation should be executed, and the data can be read to memory.

3. Sparing a defective sector—This is a more complicated version of 1 and 2 above, and is performed with the following commands. Steps a and b and steps c and d should be performed in the interlock mode, whereas between steps b and c interlock mode may not be desirable since ECC correction may be necessary.

a) First find the header of the sector located physically prior to the defective sector.

b) Second, do a write header operation to re-write the ID field, indicating a bad sector. If the data field is to be recovered the write header operation could be accompanied by a read data operation, and the data can be read to memory.

c) Third the DDC does a compare header operation to find the sector prior to the spare sector (if located on the same track). (If the spare has a known header a Compare Header is all that is necessary.)

d) Finally the spare sector ID is rewritten and the old sector's data is written to the spare sector.

In general, Interlocked mode operation is required only when unique operations on physically adjacent sectors is necessary. General multi-sector operations not performed on adjacent sectors can be cascaded without using the interlock mode.

DDC in Command Accept Mode

Step 1. The Parameter and Count registers must be loaded with the proper format information (if not already loaded with the correct information). The Header byte pattern registers must be loaded with the exact header information for the sector to be that will be operated on. Many Interlock mode command sequences may not need the Number of Sector Operations Counter loaded, but if needed it should be loaded with the number of sectors to be transferred.

Step 2. The Local DMA channel must be initialized. The Local transfer register should be configured to the desired DMA mode. The DMA address registers are loaded with the data transfer address. If the Remote DMA is used in coordination with the local transfer, especially in tracking mode, the Remote Transfer register should be initialized, as should its address, and length information. The header complete and command complete interrupts should be enabled, in order to enable determination of when to load the subsequent sector's information and command. This is done by setting the EI, EIH, IR bits in the Operation Command Register.

Step 3. Finally the Drive Command register is loaded with the desired first Command. If a remote DMA oeration is to be performed, the Operation Command register should be loaded to start the remote DMA.

DDC in Command Perform Mode

Step 4. The DDC will perform the operation locating the first sector, acquiring the system bus, and transferring that sector's data. As soon as the first sector's header has been located, then the DDC issues a header complete interrupt, and the host must update all desired registers, and finally loading the Drive Command register with the new information. (Note: if the interlocked operation was a multi-sector (NSO not equal 0) operation then the Drive command is not updated, since the operation is a continuation of the multi-sector operation.) Finally the Interlock Register is written to, and step 4 is repeated until the microprocessor is done writing commands. During the last operation, the Interlock register must be written to avoid a late interlock error.

The command may be terminated early if the DDC could not find one of the correct sectors being sought, or the Interlock register is not written to prior to the beginning of the next sector.

DDC in Result/Error Mode

Step 5. At the end of each header field an interrupt is issued. The operation has completed when the Header Interrupt and Operation Complete is received after the last command. If the command terminated properly, and the μP can read this from the Status and Error registers and can proceed to the next command. If an error occurred the microprocessor should service this by either retrying the operation, or correcting the problem.

GENERAL CHAINING OF DISK COMMANDS

Various commands can be executed one right after the other, like the interlock mode described above, except without the Interlock timing constraints. This is done by enabling the Header Complete Interrupt executing a disk command, and when the interrupt is received by the CPU, it checks to make sure it is the Header Complete interrupt, and that the Next Disk Command bit is set. If it is set the CPU can then execute a new operation, while the old operation is completing. In this way fast access to the data on the disk track can be achieved.

7.5 DMA OPERATIONS

In this section, the DDC's data transfer operations using on-chip or external DMA will be discussed in depth. Discussion of DMA as part of the above disk operations has been omitted in favor of a separate discussion. It is important for the designer to keep in mind that while the type of DMA operation won't affect the individual commands, it can be a very important factor in overall system through-put, and bus utilization. In general, once a disk sector buffering scheme and method of transferring data to/from the system has been designed, the DMA mode selection is obvious, and usually remains fixed. The DDC-system interface connections for different system applications are discussed in chapter 6.

7.5.1 Data Transfer Features

All DMA operations are supported by the following four features. These features are valid for all types of DMA modes described in section 7.5.2 including the Slave mode (external DMA).

PROGRAMMABLE BURST LENGTHS

The data transfer from/to the DDC to/from the system is fully programmable. In single bus systems, the data from/to the FIFO to/from the memory, can either be transferred in 32-byte bursts or in smaller bursts of 2, 8, 16 or 24 bytes (or 1, 4, 8 or 12 words). In dual bus systems, data can be transferred either up to 64 Kbytes in a single operation or in smaller operations. The programmable burst lengths feature accommodates the variations in bus latency time usually present in all systems (see Chapter 6).

The DDC is programmed for the desired data transfer mode through LTEB, LBL1, and LBL2 bits in the Local Transfer Register and the RTEB, RBL1, RBL2 bits in the Remote Transfer Register. For Remote transfers, the DMA Byte Count registers are also used.

8-BIT OR 16-BIT WIDE TRANSFERS

Data can be transferred either byte wide or word wide. This is achieved through LWDT and RWDT bits in the Local and Remote Transfer Registers, respectively. The DMA address counters are incremented by one for byte wide and by two for word wide transfers.

SLOW READ/WRITE

For slow memory or other devices, the normal DMA memory read/write cycle of four periods can be extended to five cycles for all DMA modes (including external DMA), using bit LSRW and bit RSRW in the Local and Remote Transfer Registers respectively. The read/write cycles can also be extended to an infinite length by using the External Status input (pin 17) of the DDC in conjunction with EEW bit in the Remote Transfer Register.

REVERSE BYTE ORDER

This option is only valid for 16-bit wide transfers using the Local DMA channel. It enables the two bytes being transferred to be mapped with the high order byte to AD0–7 and the low order byte to AD8–15, or vice-versa. This could be achieved through RBO bit in the Local Transfer Register.

(Note: This option is still functional in 8 bit mode, however it performs no useful function. When reading, the first byte DMA'd was the second byte read, the second DMA'd byte the first read, the third DMA'd byte the fourth, the fourth DMA'd byte the third, and so on. Similar order occurs for a write.)

7.5.2 DMA Modes

Various data transfers are carried out by configuring the DDC in one of the three main DMA modes: single channel, dual channel or external DMA. Some of this has been discussed in Chapter 6.

SINGLE CHANNEL (LOCAL DMA) MODE

In the local DMA mode, only three DMA registers/counters are used; the Local Transfer Register, DMA Address Byte 0 and 1. The Sector Byte Count 0 and 1 registers determines the sector size. A local transfer operation can be carried out following the steps below:

DDC in Command Accept Mode

Step 1. The DMA Address Bytes (0 and 1) Counters are initialized with local (or main) memory address to/from where the data is to be transferred from/to the FIFO.

Step 2. The Local Transfer Register is set for enabling the local DMA channel (bit SLD), 8- or 16-bit transfer (bit LDWT), reverse-byte order (optional in 16-bit data transfer mode, using bit RB0). Slow Read/Write cycles (bit LSRW), Long Address (bit LA), and the burst length (bits LTEB, LBL1, LBL2).

Step 3. The Operation Command Register is loaded for enabling interrupts, if desired (bit EI).

Step 4. Finally, the Drive Command (DC) Register is loaded with the desired DDC command. See Table 7.5. The DDC enters the command perform mode immediately after the DC register is loaded. (This step is the same as Step 3 in previous read/write operations discussions.)

DDC in Command Perform Mode

Step 5. The DDC should be granted bus control on the occurrence of an LRQ. The DDC will generate an interrupt after the completion of the operation or on the occurrence of an error. (Same as previous read/write operations Step 4.)

DDC In Result/Error Mode

Step 6. On the occurrence of an interrupt, the Status Register is read. In case of an operation complete (the NDC bit), steps 1 through 6 may be repeated, or the DDC may be initialized for a new operation. If an error was occurred, appropriate actions should be taken, as discussed in section 7.7.

DUAL CHANNEL TRACKING (LOCAL AND REMOTE) MODE

In the dual DMA mode, all the DMA registers are used (see Chapter 5). The DDC can further be set for either a Tracking Dual DMA or a NON-Tracking Dual DMA mode.

In Tracking mode, data is transferred from the on-chip FIFO to the system I/O port through the local buffer memory, and vice versa. The entire DMA operation is controlled by the DDC and external arbitration logic that synchronizes the DDC's remote operation with the external DMA for the system, after initialized by the microprocessor. Basically, local and remote transfers are dependent on each other and the DDC keeps track of both transfers in order to avoid any possible data overlapping in the local buffer memory.

This mode effectively turns the buffer memory into a large FIFO. This is accomplished through the use of DMA Sector Counter (DSC), which keeps track of the difference between sectors read/written from/to the disk and sectors transferred to/from the host system. Each time the source transfers a sector or data into buffer memory (length is determined by the Sector Byte Count Register pair), the DSC register is incremented. It is decremented each time the destination has transferred a sector of data. Whenever the DSC register contents become zero, destination transfers are inhibited. Note that in a Disk Read operation, local DMA (or the FIFO) is the source and remote DMA (or system I/O port) is the destination. Similarly, in a Disk Write operation, the remote DMA (or the system I/O port) is the source and local DMA (or the FIFO) is the destination. A detailed step-by-step disk read/write operation, in the tracking dual DMA mode, is given below and the flow chart is given in *Figure 7.12*.

DDC in Command Accept Mode

Step 1. The DMA Address Bytes (0, 1, 2, 3) registers are loaded with the same local and remote start address in the local buffer memory.

Step 2. Bits SLD and SRD in the Local and Remote Transfer Registers are set to enable both local and remote DMA channels. The DDC is configured for Tracking Mode by setting the TM bit in the Remote Transfer Register. Other options such as 8-/16-bit transfer, slow read/write cycles, and

burst lengths for both FIFO and local buffer memory are selected through LWDT, LSRW, LTEB, LBL1 and LBL2 bits in the Local Transfer Register and RWDT, RSRW or EEW, RTEB, RBL1 and RBL2 bits in the Remote Transfer Register. Bit LA in the Local Transfer Register must be reset for 16-bit address mode.

Step 3. The Number of Sector Operations (NSO) counter and Sector Counter (SC) are loaded for multi-sector operation. Only SC should be loaded for a single sector operation.

Step 4. Interrupts are enabled using EI bit in the Operation Command (OC) Register.

Step 5. Finally, the desired Read/Write command (see Table 7.6). (This is the same as Step 3 in the previous Read/Write command descriptions.)

DDC in Command Perform Mode

Step 6. The DDC will start performing the desired operation after the DC register is loaded. An LRQ, RRQ or Interrupt should be expected. The DDC should be given the bus control when LRQ or RRQ occurs. (Same as Step 4 in previous Read/Write command descriptions.)

DDC in Result/Error Mode

Step 7. If an interrupt is generated, it should be serviced properly (see section 7.7 for interrupt servicing). If the operation was completed successfully, steps 1 through 6 may be repeated for a new operation. (Same as Step 5 in previous Read/Write command descriptions.)

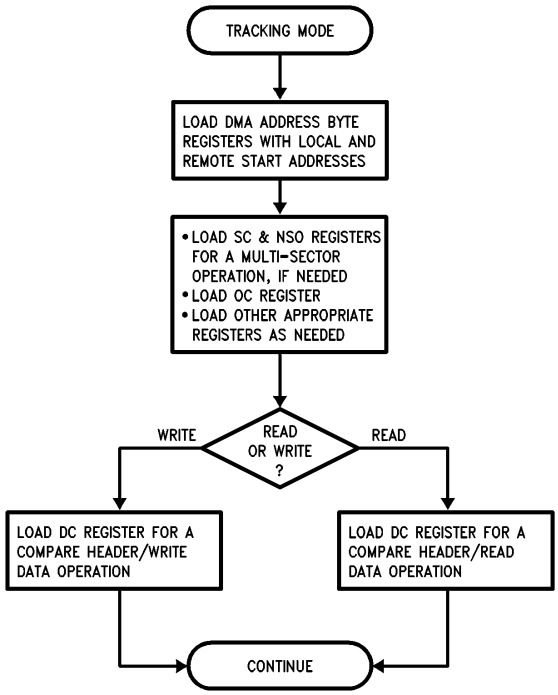
DUAL DMA NON-TRACKING (LOCAL AND REMOTE) MODE

In the non-tracking dual channel DMA mode, the Local and Remote transfers are independent of each other. The controlling microprocessor has to keep track of both transfers to avoid any possible data overlapping in the local buffer memory. The DMA Address (bytes 0–3) Registers are set up independently for Local and Remote transfers. All the necessary steps needed to perform a data transfer between the FIFO and system I/O port with the DDC in this mode are explained below and also shown in a flow chart in *Figure 7.16*.

DDC in Command Accept Mode

Step 1. The DMA Address (byte 0–3) are set up for the desired Local and Remote addresses. Any address within 64k memory space could be loaded.

Step 2. The DDC is configured in the non-tracking mode, first by enabling Local and Remote DMA channels through SLD and SRD bits in the Local and Remote transfer Registers. The LA bit is also set to zero for 16-bit address for both DMA channels. Other DMA options such as 8-/16-bit data transfer slow read/write, reverse byte ordering, and burst length, are selected through LWDT and RWDT; LSRW, RSRW and EEW; RBO; and LTEB, LBL1, LBL2, RTEB, RBL1, and RBL2 bits in the Local and Remote Transfer Registers.



TL/F/8663–E9

FIGURE 7.12. Dual DMA Tracking Mode μP Programming Flow Chart

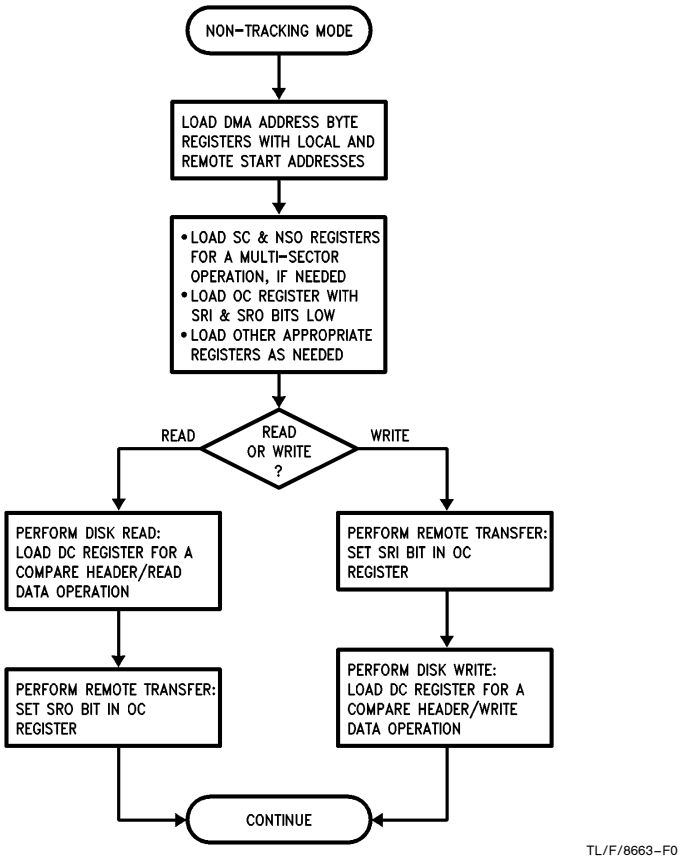


FIGURE 7.13. Dual DMA Non-tracking Mode μP Programming Flow Chart

Step 3. The Number of Sector Operation (NSO) Counter and Sector Counter (SC) are loaded for multi-sector operation. Only SC is loaded for a single sector operation.

Step 4. Write Operation: Before having the DDC to perform a write operation, data is transferred from the system I/O port to the buffer memory by enabling the Remote channel through SRI bit in the Operation Command Register. Interrupts are also enabled using the EI bit.

Read Operation: The Drive Command (DC) Register is loaded with the desired Read command (refer to Table 5.6). If a multisector operation is selected, the SAIS bit may be set to zero for the operation to start at the Index pulse.

DDC Performs the Remote Transfer

Write Operation: The DDC will transfer the remote data to local memory and will issue an operation complete interrupt. Remote transfer operations could be repeated to fill the local memory before performing a disk operation. The DDC now should be initialized for the actual disk write operation.

Read Operation: The DDC will complete the disk read just like a normal operation. See Step 7. Now the data may be transferred to the system I/O or any remote locations.

Step 5. Finally, the Operation Command Register is loaded to enable data transfer from the local buffer memory to the system I/O, with SRO bit set. The interrupts may also be enabled with the EI bit, if required.

The DDC will start the operation when the OC Register is loaded. The RRQ must be acknowledged.

Step 6. The Drive Command register is loaded with the desired write command. If a multi-sector operation is desired, the SAIS bit may be reset for the operation to start at the Index pulse. (This is the same as Step 3 for Read/Write operations discussed previously.)

DDC in Command Perform Mode

Step 7. The DDC will start performing the desired write operation immediately and will issue a local request, LRQ. Upon receiving an LACK, it completes the write operation. (This is the same as Step 4 for previously discussed Read/Write operations.)

DDC in Result/Error Mode

Step 8. The DDC will issue an interrupt which should be serviced properly (refer section 7.7 for interrupt servicing). In case of an operation complete interrupt, steps 1 through 5 may be repeated for a new operation.

IMPORTANT NOTES

1. By setting both SRI and SRO simultaneously, any non-tracking DMA operation will stop. The current remote address and remote data byte count will be retained, and the local DMA will be unaffected. Loading the original OC instruction (input or output) will restart the original instruction from the last remote DMA address.

2. In either Tracking or Non-tracking mode, if either channel is loaded with an odd byte transfer count, the DDC will transfer the next higher even byte. For example, if 511 was loaded in Remote Data Byte Count Registers, 512 bytes would be transferred, with the valid data only in the first 511 bytes.

3. In the Tracking mode the DDC keeps track of the data in the buffer memory. The Remote Transfer follows the Local transfer by a sector length and the DDC makes sure that the correct data is transferred to the system memory. However in the Non-tracking mode the remote channel is independent of the disk operation and hence the remote channel can follow the local channel as closely as possible. The microprocessor is responsible of preventing overlap of data.

4. Even though normally the remote channel would be used for transfers from system to buffer memory (and vice versa) and the local channel for transfers from the buffer memory to the FIFO (and vice versa), the remote channel could also be used for some other purpose that is independent of the DDC's other operations.

EXTERNAL DMA

In external DMA mode, the data transfer between the on-chip FIFO and external memory (local buffer or system) is controlled by the external DMA. The DDC is programmed to perform a disk read/write operation without the internal DMA. Whenever the FIFO needs any data transfer, the DDC asserts LRQ. At this point, external DMA takes control and completes that particular data transfer. The following steps illustrate the necessary actions to perform a disk read/write operation using external DMA i.e. DDC in the slave mode.

DDC in Command Accept Mode

Step 1. The registers are initialized.

Step 2. Interrupts are enabled using EI bit in the Operation Command Register.

Step 3. Finally, the Drive Command Register is loaded for the desired Read/Write operation.

DDC in Command Perform Mode

Step 4. The DDC will start performing the operation and LRQ will be asserted when the FIFO requires the data transfer. The LRQ must be acknowledged by the external DMA in order to complete the operation.

DDC in Result/Error Mode

Step 5. If an interrupt is issued, it must be serviced, (refer section 7.7). In case of an operation complete interrupt, steps 1 through 3 may be repeated for a new operation.

7.6 ERROR DETECTION AND CORRECTION

The Disk Data Controller, DDC has comprehensive and versatile error detecting and correcting capabilities. It features a fully programmable ECC;

- Programmable Preset Pattern
- Programmable Polynomial Taps
- Programmable Correction Spans
- Programmable Assignment of CRC/ECC on Header or Data

There are essentially two internal codes available; a fixed Cyclic Redundancy Checking (CRC) code for detecting errors only, which uses a CRC-CCITT polynomial that provides 16 generated check bits for appending to the Header fields and/or Data fields. The other type is the ECC code which may be a Fire code or a Computer generated code with 32 or 48 generated check bits that may be appended to the Header field and Data field. National Semiconductor recommends a computer generated polynomial called the Glover 140A0443 code with a correction span of 5-bits for MFM encoded drives. The designation represents the hexadecimal equivalent of the forward polynomial and it requires a preset of all 1's. The code has two polynomials; the forward one for checkbit generation and checking and the reverse one for error location. The error detection span is 32-bits while the correction span is 5-bits. The number of bytes in the sector determines the integrity of the code. The maximum sector length the code can handle is 1024 bytes of data and 4 bytes of ECC, which is within the limits of most disk formats. The completely programmable feature of the DDC with respect to ECC offers a lot of flexibility to the user. In case a user prefers to use his own high integrity code, the DDC can be configured to interface easily with external ECC circuitry and the DDC can be programmed to operate in the external ECC mode, as discussed in chapter 4. There are essentially three kinds of operations associated with the ECC circuitry: 1) checkbit generation, 2) checkbit verification, and 3) error location.

7.6.1 Error Detection

Internal Checkbit Verification—Write

This operation occurs when the controller is performing a write operation. The ECC shift register is downloaded with the preset value stored in the Preset Register. The code length is selected independently for Header and Data appendage. Bits being shifted out of the SERDES (serializer/deserializer) to the disk, are also shifted into the ECC Shift Register. When the last bit of the Header or Data field has been transmitted out of the DDC, the generated check bits in the ECC shift register are directly shifted out and onto the disk, starting with the MSB and ending with bit 0. After the ECC bits have been appended, the DDC switches to the next field.

Internal Checkbit Verification—Read

This function will occur concurrently with a read data operation from the disk. The ECC Shift Register is first preset from the Preset Registers. The incoming Header or Data field is serially fed into the same ECC Shift Register that is used to generate checkbits. When all the Header or Data field bits and all the generated checkbits have entered the ECC Shift Register, the status of the bits in it is checked for an all zeroes condition. If it is true then the field contains no errors, else if any of the ECC Shift Register's bits are high, the field contains an error. In the case of a Header field error, the Header Fault bit (SO), in the Status Register is set, while in case of a Data field error, the Data Field Error bit (E1), of the Error Register is set.

System Alternatives on Error Detection

Once an error has been detected by the ECC logic, the Re-enable (REN) bit must be reset via the Drive Command Register, before proceeding. If a Header field error is detected, the DDC will react differently depending on the Header operation involved. The various options are discussed under the Drive Command Register description. If an error is detected in the Data field, a re-read is initiated (by the system), to overrule a soft error. If the data is still not corrected after several re-reads, it implies the detection of a hard error. By re-reading the sector in question and comparing the syndromes to previous retries, a certain level of confidence can be reached, that the error is media induced and ECC correction can be attempted. The syndrome bytes in the ECC Shift Register will contain the bit error information, although the bytes in error have been transferred to memory.

7.6.2 Error Correction

The DDC has a maximum correction span of 15 bits, i.e. it can correct up to 15 contiguous bits in error, or a span of errors 15 bits or less. Of course correction can only be attempted if internal ECC checkbits were appended to the data field when written to the disk. The first step in the correction process is to load the Data Count Register with the data count, (sector byte count) plus the number of bytes of checkbits, i.e. sector byte count registers must be initialized to sector length plus 4 or 6 for 32 bit mode or 48 bit mode ECC respectively. Then the correction cycle is initiated by setting the Start Correction Cycle bit of the Operation Command Register. This should be done before any further Drive Command operation is issued to the DDC. This prevents the destruction of the stored syndromes in the ECC Shift Register. Also while the correction cycle is in progress,

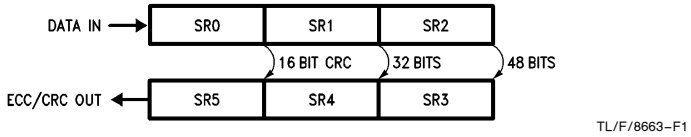


FIGURE 7.14(a). Hardware Configuration of ECC Shift Register

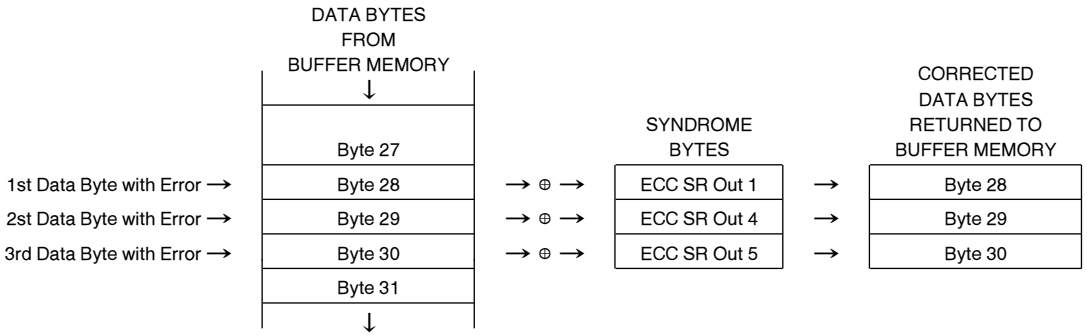


FIGURE 7.14(b). 32-Bit ECC Correction Process

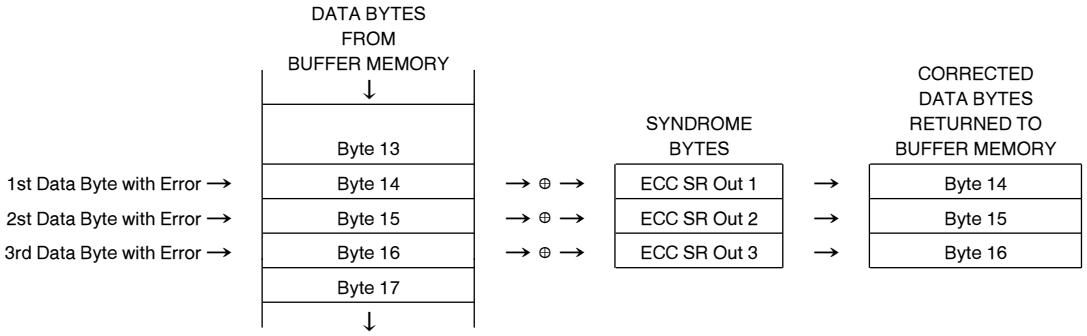


FIGURE 7.14(c). 48-Bit ECC Correction Process

MP Operations after Error Located in DDC

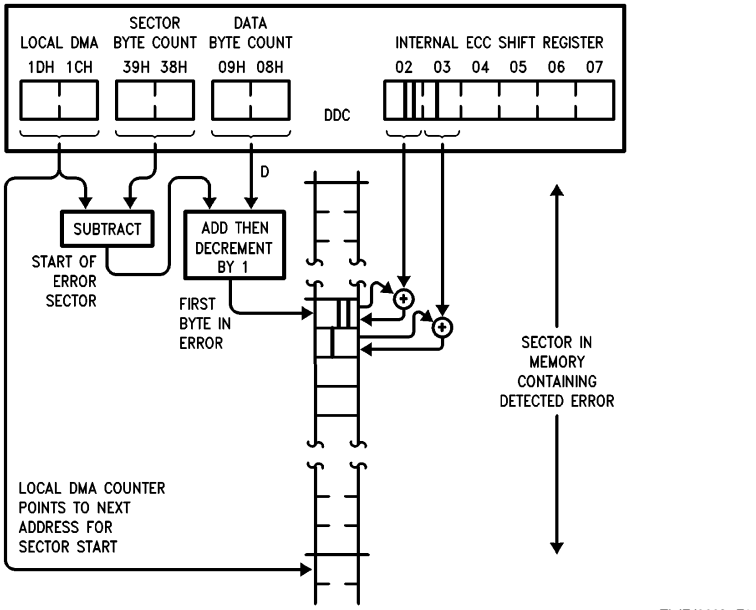


FIGURE 7.15. Location of Bytes in Error (in Memory) for the Correction Process

the DDC ignores any drive command loaded into the Drive Command Register. On initiation of the Correction Cycle, the Correction Cycle Active flag, (bit 6 of the Status register), will go high.

The ECC Shift Register contains encoded information with regards to both the location of the bytes in error and the error pattern. The ECC Shift Register's contents are transposed which sets up a reverse shift without actually reversing the direction of shift in the shift register. The advantage of reverse shifting is that a non-correctable error is determined much quicker than if forward shifting is used. It also guarantees the completion of the correction cycle within the time it takes to read one sector of the disk. The ECC logic begins shifting, looking for a zero detect, i.e. detection of all zeroes in the upper (32-C) or (48-C) bits of the ECC Shift Register, where C is the correction span selected. After 8 shifts, the Data Count Register begins decrementing, with one down count for every 8 shifts of the ECC Shift Register. When the zero detect condition occurs, the control logic will stop decrementing the Data Count Register and its state indicates the byte that is in error. If the Data Byte Counter decrements to zero before the selected most significant bits of the ECC Shift Register are all zeroes, the error is non-correctable. In case of this condition or the zero detect condition of the ECC Shift Register, an interrupt is issued to indicate to the host microprocessor that the correction cycle has finished, indicated by the CCA flag (bit 6 of the Status Register being reset).

During the correction cycle other operations like completion of remote DMA etc., may issue an interrupt which should be serviced to enable recognition of the interrupt on completion of the correction cycle. The Error Register bit CF is examined, which if set signifies a non correctable error. If the bit is not set, then the error is correctable and must be either in the data field or the checkbits of the ECC field or overlapping both fields.

At the instant when the 'zero detect' condition occurs in the ECC Shift Register, the status of the Data Count Register indicates the byte in error. For example—if the data count register shows 515, then the 515th byte of the data field is in error. If there were only 512 bytes in the data field, then 515 means that the 3rd byte of the checkbit field is in error. The syndrome bytes in the ECC Shift register should be aligned so that the Most Significant Bit of the syndrome field align with the Most Significant Bit of the byte 515. However, if the syndrome spans a field of two bytes, then it will align with byte 515 and 516. When the data byte in error is located, the ECC logic makes sure that the syndrome bits are aligned properly on a bit by bit basis with that byte in error. Therefore, it will continue to shift until this has happened. To facilitate the speed restraints of the process, the syndrome

will get shifted one full byte beyond where one would expect to see it in the ECC Shift Register. The syndrome bits will be located starting at the second byte of the ECC Shift Register. *Figure 7.14 (a)* shows the orientation of the ECC Shift Register for various sizes of polynomials selected.

Errors are corrected by XOR'ing syndrome bytes (ECC SR 0–5) with the bytes in the data record in memory that contain the error. The address of the first byte of the data field, in error is computed as follows: [current value of DMA address bytes 0 and 1] − [sector byte count] + [data byte count] − 1. For performing a correction with 32 bit ECC, ECC SR1, ECC SR4, and ECC SR5 contain the syndrome pattern in that sequence. ECC SR2 and ECC SR3 are not used in 32-bit mode and will contain 0's if read. ECC SR0 will contain all 0's if the error is correctable, and may contain some set bits if not. The bytes in error (in the memory) are located as shown in *Figure 7.15* while the correction process is shown in *Figure 7.14 (b)* and *(c)*. To perform a 48 bit correction ECC SR1, ECC SR2 and ECC SR3 should be read sequentially for the syndrome bits. ECC SR0, ECC SR4 and ECC SR5 are not used and will contain 0's for a correctable error. *Figure 7.16* shows an example of the correction process. *Figure 7.17* gives a flow chart of the correction cycle operations.

EXAMPLE OF A 32-BIT CORRECTION

Shown in *Figure 7.18* is a record with several bits read in error from the disk. Bits D4, D11, D13 and D14, now located in memory were read incorrectly and need to be corrected. As can be seen, the correction pattern provided in ECC SR1 and ECC SR2 can be used to correct bits D4, D11, D13 and D14. The CPU reads the Data Byte Count and computes the address of the first data byte in error, read from the disk. This byte is XOR'ed with ECC SR1 and is written back to memory. The second byte read form the disk is XOR'ed with ECC SR4 and then written back. ECC SR5 need not be used since it contains all 0's.

7.6.3 Programming the ECC

There are two sets of six registers used to program the ECC. One set of six is used to program the polynomial taps, while the other set is used to establish a preset pattern. Bits contained in the ECC Control Register are used to control the correction span. The Data Format Register contains bits for choosing the desired type of appendage: either 32 or 48 bit programmable ECC polynomials, or the 16 bit CCITT CRC polynomial.

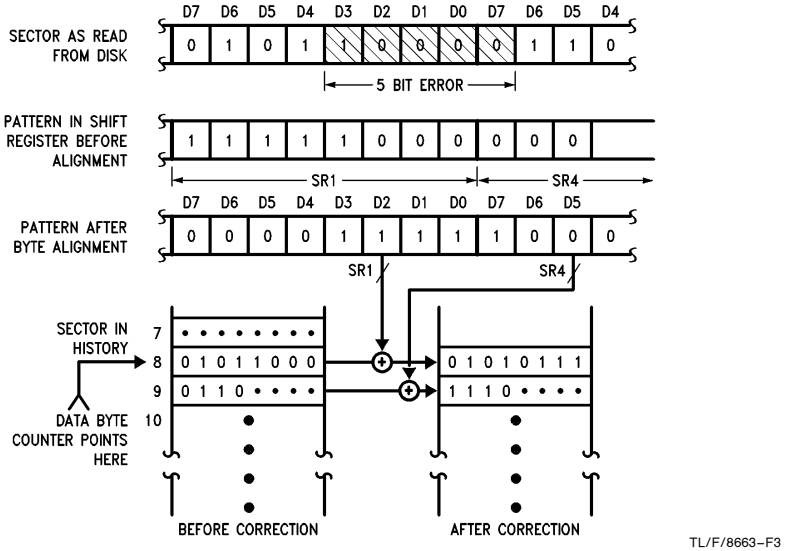
PROGRAMMING POLYNOMIAL TAPS

To program a polynomial into the shift register, each tap position used in the code must be set to 0, and all unused taps should be set to 1. The bit assignment for these registers in 48 and 32-bit modes is shown in the tables, *Figure 7.21 (a).* It is important to note that for 32-bit codes, PTB2 and PTB3 must be set to all 1's. Failure to do so will result in improper operation. Also x^{48} for 48-bit and x^{32} for 32-bit ECC are implied and so is x^0, even though this bit is accessible.

PROGRAMMING PRESET PATTERN

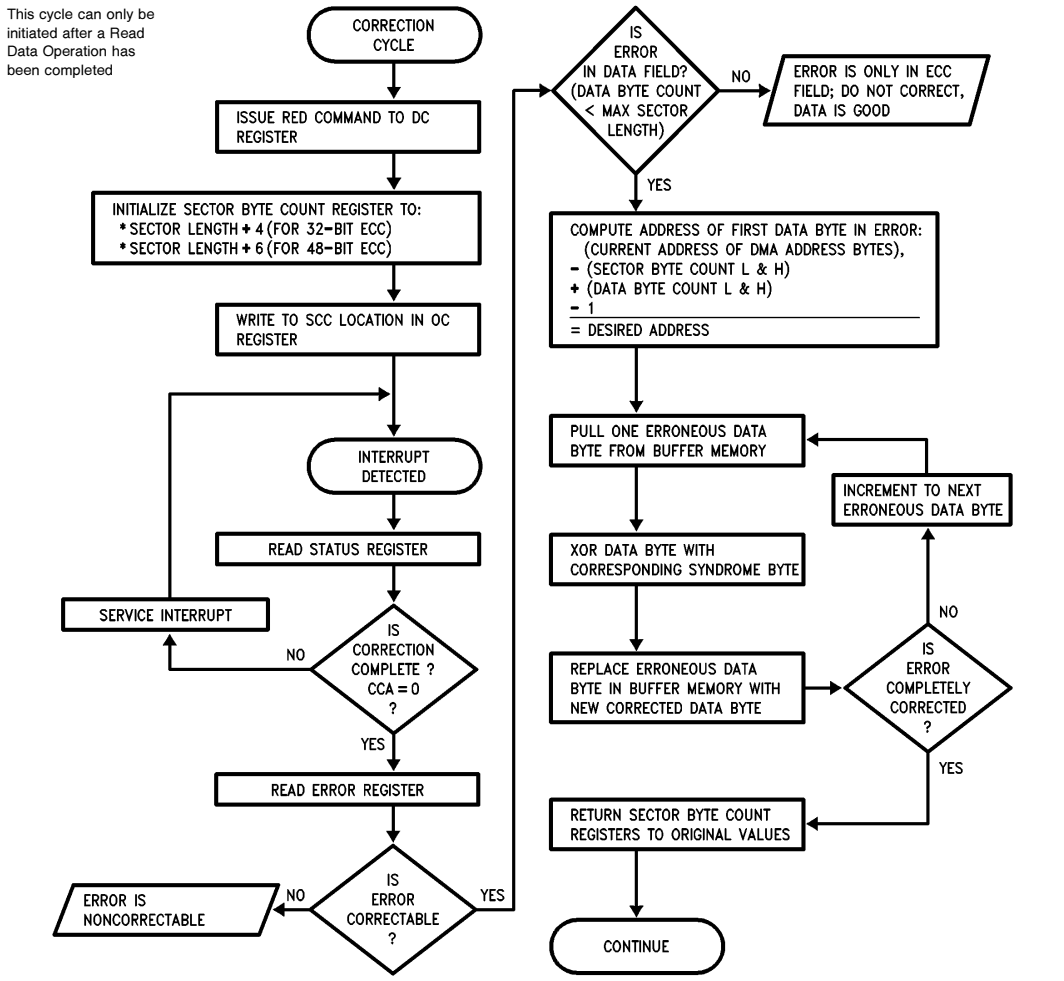
PPB0–PPB5 must be initialized to program the preset pattern that the shift registers will be preset to. As in the polynomial taps, x^{48}, x^{32}, and x^0 are implied. The assignment of the bits for 48 and 32-bit modes is shown in *Figure 7.21 (b)*. The value programmed into each register will be the preset pattern for the eight bits of the corresponding shift register. For typical operation, these will be programmed to all 1's. All unused presets should be set to 0. In 32-bit mode, PPB2 and PPB3 must be set to all 0's. Failure to do so will result in improper operation.

An Example of ECC in Action



TL/F/8663–F3

FIGURE 7.16. How Error Correction Works



TL/F/8663–F4

FIGURE 7.17. Flow Chart of the Correction Cycle Operation

ECC CONTROL REGISTER

The ECC Control Register controls a number of functions. The correction span can be programmed using four bits of this register. Errors longer than the correction span are treated as non-correctable. The allowable correction span is 3–15 bits. If a span outside this range is loaded, then the DDC defaults to a span of three bits. There is a bit (HEN) to indicate whether Header address mark and/or synch fields are encapsulated in the CRC/ECC calculation. There is also a bit (DEN) for indicating whether data address mark and/or synch fields are encapsulated in the CRC/ECC calculation. Facility for inverting data entering and leaving the ECC Shift Register is also provided. For selecting the internal 16-bit CRC polynomial, the appropriate bits in the Disk Format register are set, the ECC Control Register is programmed as desired.

Syndrome Pattern

Register		Bit Number 7	6	5	4	3	2	1	0
ECC	SR1	0	0	0	1	0	0	0	0
ECC	SR4	0	1	1	0	1	0	0	0
ECC	SR5	0	0	0	0	0	0	0	0

Buffer Memory

Corresponding Buffer Data Bit Pattern							
D7	D6	D5	*	D3	D2	D1	D0
D15	*	*	D12	*	D10	D9	D8
D23	D22	D21	D20	D19	D18	D17	D16

* = location of bits in error

Figure 7.18 Example of Correction Syndrome Bits relating to Data Bit Patterns

Tap Assignment 32-Bit Mode

REG #	ADDR	Bit Number DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PTB0	08	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
PTB1	09	X^{15}	X^{14}	X^{13}	X^{12}	X^{11}	X^{10}	X^9	X^8
PTB2	0A	1	1	1	1	1	1	1	1
PTB3	0B	1	1	1	1	1	1	1	1
PTB4	0C	X^{23}	X^{22}	X^{21}	X^{20}	X^{19}	X^{18}	X^{17}	X^{16}
PTB5	0D	X^{31}	X^{30}	X^{29}	X^{28}	X^{27}	X^{26}	X^{25}	X^{24}

Preset Bit Assignment 32-Bit Mode

REG #	ADDR	Bit Number DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PPB0	02	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
PPB1	03	X^{15}	X^{14}	X^{13}	X^{12}	X^{11}	X^{10}	X^9	X^8
PPB2	04	0	0	0	0	0	0	0	0
PPB3	05	0	0	0	0	0	0	0	0
PPB4	06	X^{23}	X^{22}	X^{21}	X^{20}	X^{19}	X^{18}	X^{17}	X^{16}
PPB5	07	X^{31}	X^{30}	X^{29}	X^{28}	X^{27}	X^{26}	X^{25}	X^{24}

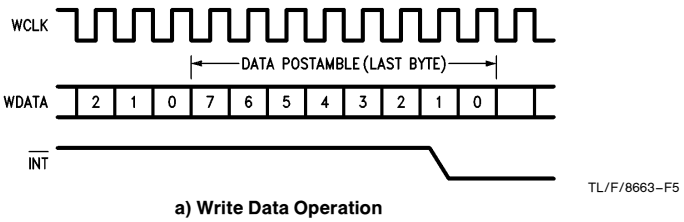
Tap Assignment 48-Bit Mode

REG #	ADDR	Bit Number DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PTB0	08	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
PTB1	09	X^{15}	X^{14}	X^{13}	X^{12}	X^{11}	X^{10}	X^9	X^8
PTB2	0A	X^{23}	X^{22}	X^{21}	X^{20}	X^{19}	X^{18}	X^{17}	X^{16}
PTB3	0B	X^{31}	X^{30}	X^{29}	X^{28}	X^{27}	X^{26}	X^{25}	X^{24}
PTB4	0C	X^{39}	X^{38}	X^{37}	X^{36}	X^{35}	X^{34}	X^{33}	X^{32}
PTB5	0D	X^{47}	X^{46}	X^{45}	X^{44}	X^{43}	X^{42}	X^{41}	X^{40}

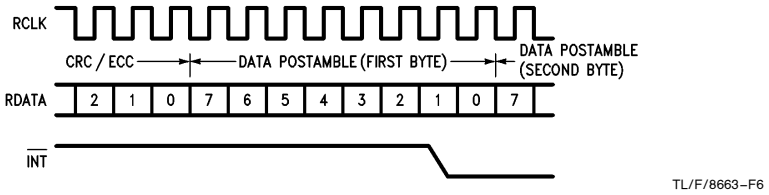
Preset Bit Assignment 48-Bit Mode

REG #	ADDR	Bit Number DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PPB0	02	X^7	X^6	X^5	X^4	X^3	X^2	X^1	X^0
PPB1	03	X^{15}	X^{14}	X^{13}	X^{12}	X^{11}	X^{10}	X^9	X^8
PPB2	04	X^{23}	X^{22}	X^{21}	X^{20}	X^{19}	X^{18}	X^{17}	X^{16}
PPB3	05	X^{31}	X^{30}	X^{29}	X^{28}	X^{27}	X^{26}	X^{25}	X^{24}
PPB4	06	X^{39}	X^{38}	X^{37}	X^{36}	X^{35}	X^{34}	X^{33}	X^{32}
PPB5	07	X^{47}	X^{46}	X^{45}	X^{44}	X^{43}	X^{42}	X^{41}	X^{40}

FIGURE 7-19. Programming the Presets of Taps; the Tap and Preset Register Configurations

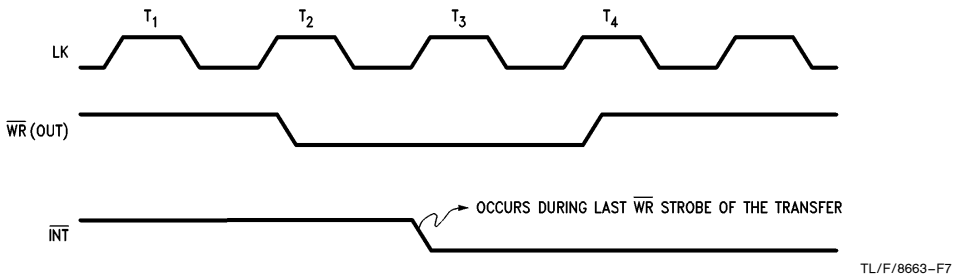


a) Write Data Operation

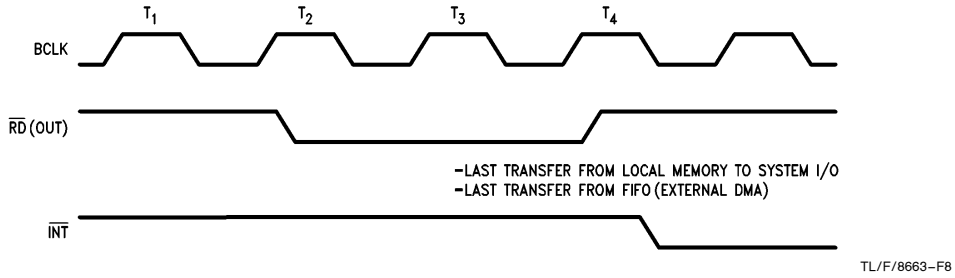


b) Check Data Operation

FIGURE 7.20. Operation Complete Interrupt Timing



c) Read Data, in Single or Dual Channel (Non-tracking) Move



d) Read Data, in Dual Channel (Tracking) or in an External DMA Mode

FIGURE 7.20. Operation Complete Interrupt (Continued)

Example of programming the ECC registers

Objective: To program the 32-bit polynomial of the form; (This is National Semiconductor's recommended polynomial)

$x^{32} + x^{28} + x^{26} + x^{19} + x^{17} + x^{10} + x^6 + x^2 + x^0$

with a preset of all 1's, a correction span of 5-bits with no header/data encapsulation. The registers would be programmed as given below. Note that as defined earlier, PTB2 and PTB3 must be all 1's and PPB2 and PPB3 must be all 0's.

Polynomial Taps Registers

REG#	7	6	5	4	3	2	1	0
PTB0	1	0	1	1	1	0	1	0
PTB1	1	1	1	1	1	0	1	1
PTB2	1	1	1	1	1	1	1	1
PTB3	1	1	1	1	1	1	1	1
PTB4	1	1	1	1	0	1	0	1
PTB5	1	1	1	0	1	0	1	1

Polynomial Preset Registers

REG#	7	6	5	4	3	2	1	0
PPB0	1	1	1	1	1	1	1	1
PPB1	1	1	1	1	1	1	1	1
PPB2	0	0	0	0	0	0	0	0
PPB3	0	0	0	0	0	0	0	0
PPB4	1	1	1	1	1	1	1	1
PPB5	1	1	1	1	1	1	1	1

ECC Control Register

BIT #	D7	D6	D5	D4	D3	D2	D1	D0
	1	0	0	1	0	1	0	1

7.6.4 Internal ECC Diagnostics

The DDC has a diagnostic capability for validating the internal ECC function. By loading the Data Byte Count Register with the number of bytes in the sector plus the number of bytes of ECC appendage for the Data field. The internal CRC/ECC appendage for the Data field is set to zero so that no CRC/ECC will append the data field. Next the microprocessor sets up a data pattern in memory of all zeroes for the nominal sector length, except for bit positions where simulated errors are desired. Also, the microprocessor appends to this data the ECC appendage for an all zeroes data field by setting the Drive Command Register to perform a Compare Header-Write Data operation. In this way the DDC executes a diagnostic write function. In this mode, the data field from memory is written as in a normal write operation to the data field of the selected sector. Then the 32-bits or 48-bits of ECC check are also issued, where these check bits are falsely generated as if from an all zeroes data field. The selected sector now contains an all zeroes data field with simulated error bits followed by an ECC appendage representing checkbits generated from an all zeroes data field. The Data Byte Count is now re-loaded with the normal sector length and the correct ECC appendage length selected. A subsequent Read Data operation should produce an error indication. A correction cycle can then be implemented and the syndromes can be examined along with the Data Byte Counter contents. The microprocessor can then compare these syndromes with the positions of the simulated error bits previously written in the data field. This offers the user a diagnostics capability that simulates errors easily, merely by writing the data field with all zeroes except where the simulated error locations are desired.

7.6.5 Encapsulation of Internal ECC with External ECC

The external ECC field may be used to encapsulate the internal ECC/CRC field as a confirmation of error detection. The advantages of this scheme are that both external and internal ECC must agree on 1) the existence of the error, 2) location of the error, and 3) the error pattern. If an error is detected either internally or externally, the DDC will operate as if an internal error were detected.

7.7 INTERRUPTS

The DDC will interrupt the microprocessor only if the Interrupt Enable bit (EI) in the Operation Command register is set high. If it is not set, the INTERRUPT output is always forced high.

7.7.1 Types of Interrupt

The interrupts generated by the DDC can be divided into four categories:

1) Operation Complete Interrupt
2) Header Complete Interrupt
3) Error Interrupt
4) Correction Cycle Complete Interrupt

Each of the above mentioned types is explained in the following.

OPERATION COMPLETE INTERRUPT

The DDC will interrupt the microprocessor when it completes any one of the legal header-data disk operations listed in Table 5.13. The interrupt will also indicate that the DDC is ready to execute a new command. Some interrupt generation situations are explained below.

1) An interrupt will occur when the remote transfer is completed during a disk read operation in Tracking mode.

2) An interrupt will occur when the local transfer is completed during a disk read operation in Non-tracking mode.

3) In Non-tracking mode, if remote DMA channel is enabled, an interrupt will occur after the remote transfer is completed independent of the disk operation or the local transfer.

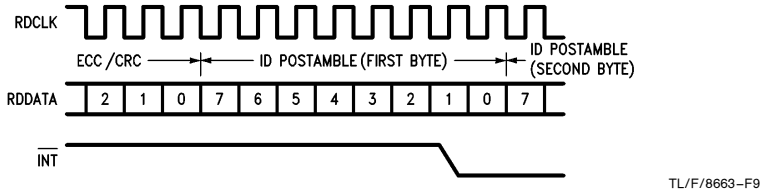
4) If the operation was a multi-sector operation, an interrupt will only occur on the completion of the last operation.

The Operation Complete interrupt generation for various Header and Data operation is shown in *Figure 7.20*. In disk write operations, the operation complete interrupt is generated when last byte of data postamble is being output by the DDC. In Header-Check data operation, operation complete interrupt occurs when first byte of data postamble enters the DDC. In disk read operations when the DDC is using only its local DMA channel (single channel DMA and non-tracking DMA modes), the operation complete interrupt is generated when the last byte (or word) is transferred to the memory. Basically it is coincident with the last WR\ strobe. When the DDC is in dual channel DMA mode, the operation complete interrupt is issued during the last RD\ strobe i.e. when last byte (or word) is transferred from local memory to system I/O. Similarly, when an external DMA is used, the operation complete interrupt is generated during the last RD\ strobe i.e. when last byte (or word) is transferred from the DDC to external memory.

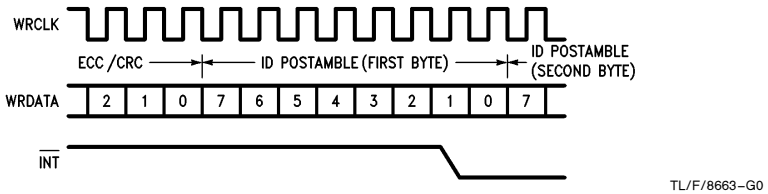
HEADER COMPLETE INTERRUPT

In all legal DDC operations listed in Table 5.13, an interrupt will be generated after a header operation only if the Enable Header Complete Interrupt bit (EHI) in the operation command register is set high. In case of multi-sector operation, this interrupt will be generated after each header of a sector has been operated on. The header complete interrupt feature is commonly used when the DDC is in Interlock Mode (Refer to section 5.2.5). On interrupt, the ID and Data fields for the next sector can be changed, if desired, before the next sector operation starts.

The header complete interrupt is coincident with the Next Disk Command bit (NDC) being set in the Status register. Thus, the controlling microprocessor can be notified to load the DDC with the next disk command. In other words, the DDC could be run continuously for any length of time by loading a new disk command whenever next disk command flag is set. The generation of header complete interrupt is shown in *Figure 7.21*. In Compare, Read and Ignore Header



a) Compare, Read or Ignore Header Operations



b) Write Header Operation

FIGURE 7.21. Header Complete Interrupt Generation

operations, the interrupt is generated when first byte of ID Postamble is being read by the DDC whereas in Write Header operation, interrupt is generated when first byte of ID Postamble is being written to the disk by the DDC.

ERROR INTERRUPT

An interrupt will be generated if any bit in the Error register is set, which in turn sets the Error Detected bit (ED) in the Status register. Refer to description of Error and Status registers in section 5.1.1. Also an error interrupt will be issued.

CORRECTION CYCLE COMPLETE INTERRUPT

An interrupt will occur at the end of an internal correction cycle independent of the result of the correction cycle. If the error was not correctable, another interrupt will not be generated, only the Correction Failed flag (CF) in Error register will be set.

7.7.2 Interrupt Servicing

As explained earlier, the DDC issues an interrupt on, an operation complete, the header operation complete, the occurrence of an error and the completion of a correction cycle. Whenever an interrupt is generated, the Status and Error register should be read in order to find out which one of the four situations has happened. In the status register, flag NDC indicates the completion of an operation, flag HMC indicates the completion of a header operation, and flag ED indicates the occurrence of an error. Only when the ED flag is set, the Error register is read to find out the type of error that caused an interrupt. Also, the CF flag in the Error register indicates result of the correction cycle. The interrupt servicing for various interrupts is described below. *Figure 7.22* shows a flow chart for servicing interrupts.

Operation Complete Interrupt: In case of operation complete interrupt, the NDC flag in the Status register gets set indicating that the DDC is ready for next command. The DDC is brought to the Command Accept mode and the desired command is loaded with all other related registers initialized. Refer to sections 7.1 through 7.6.

Header Complete Interrupt: The HMC flag in the Status register, gets set in case of header complete interrupt. This basically indicates that the header operation (ignore, compare, read or write) has completed. The information in the DDC's registers can be changed before the start of next header operation i.e. during the time when the data operation for the current sector is in progress. If the DDC is in Interlock mode, the HBC/interlock register is also written to during this time. See sections 5.2.5 and 7.2 (interlock format method).

Error Interrupt: In case of an Error interrupt, the ED flag in the Status register gets set. The Error register should be read next, to find out the error that caused the interrupt. For the description of various error flags, refer to Error register description in chapter 5. The HFASM function is explained in detail in section 7.8. Also see description of Header Byte control register in chapter 5. The DFE (data field error) is caused by an ECC/CRC error in the data field. Generally, retrying the operation takes care of this error. If this error does repeat on retries, a correction cycle should be performed. See section 7.6. The SNF (sector not found) error could also be resolved by retrying the operation. If it does repeat on retries, then the head should be repositioned. The SO (sector over run) occurs while reading or writing more data than what has been allotted on the disk and could be taken care of by checking system software. The NDS (no data synch) occurs because of a mismatch in address mark or synch fields and could be resolved by retries or system check-up. The FDL (FIFO data lost) could occur due to speed incompatibility between system and the disk drive and could be resolved by retrying or checking the system. The CF (correction cycle failed) can be taken care of by retrying the correction cycle again, if still not resolved, then that means the error is not correctable. See section 7.7 on ECC/CRC. The LI (late interlock) could also be resolved by retrying. See section 5.2.5 for details on interlock operation.

Correction Cycle Complete: The Error register is read. If CF flag indicates that the correction cycle failed then it can be performed again. After retry if it still fails, then the error is not correctable. See section 7.6 for details on correction cycle.

7.7.3 Interrupt Clearing

The INT pin will be forced inactive high any time the status register is read. If an interrupt condition arises during a status read, an interrupt will be generated as soon as the status read is finished. INT pin will also be deactivated by setting the internal Reset bit (RES) or asserting the external RESET pin. Clearing the RED bit in Drive Command register will not deactivate an interrupt.

7.8 ADDITIONAL OPERATIONS

7.8.1 Data Recovery Using the Interlock Feature

The potential use of the interlock feature is in recovering data from a sector with an unreadable header or ID field. It is assumed that the number of the sector physically preceding the bad sector on the disk is known. A single-sector operation will be performed on these sectors, and the Drive Command Register will be changed in between them. The following steps will recover the data.

Step 1. The header bytes of the physical sector preceding the desired sector are loaded into the relevant header byte pattern registers.

Step 2. The OC Register must be loaded with the EI, EHI, IR bits set. This enables the Header Complete interrupt as well as the interlock feature.

Step 3. The DC register is loaded for a single-sector, compare header/check data operation.

Step 4. After the header complete interrupt, the DC register must be loaded with an Ignore Header/Read data operation, and the Interlock Register (HBC) written to. If the controlling microprocessor fails to write to the HBC register before the end of the data field of the first sector, a Late Interlock error (LI bit in the Error Register) will be flagged, and the operation will be terminated with an interrupt.

Step 5. When the HMC interrupt occurs on the second sector, the Interlock (HBC) Register must be written to again to avoid a LI error.

Step 6. The operation will terminate normally when the data from the badly labeled sector has been read.

Figure 7.23 shows the data recovery algorithm.

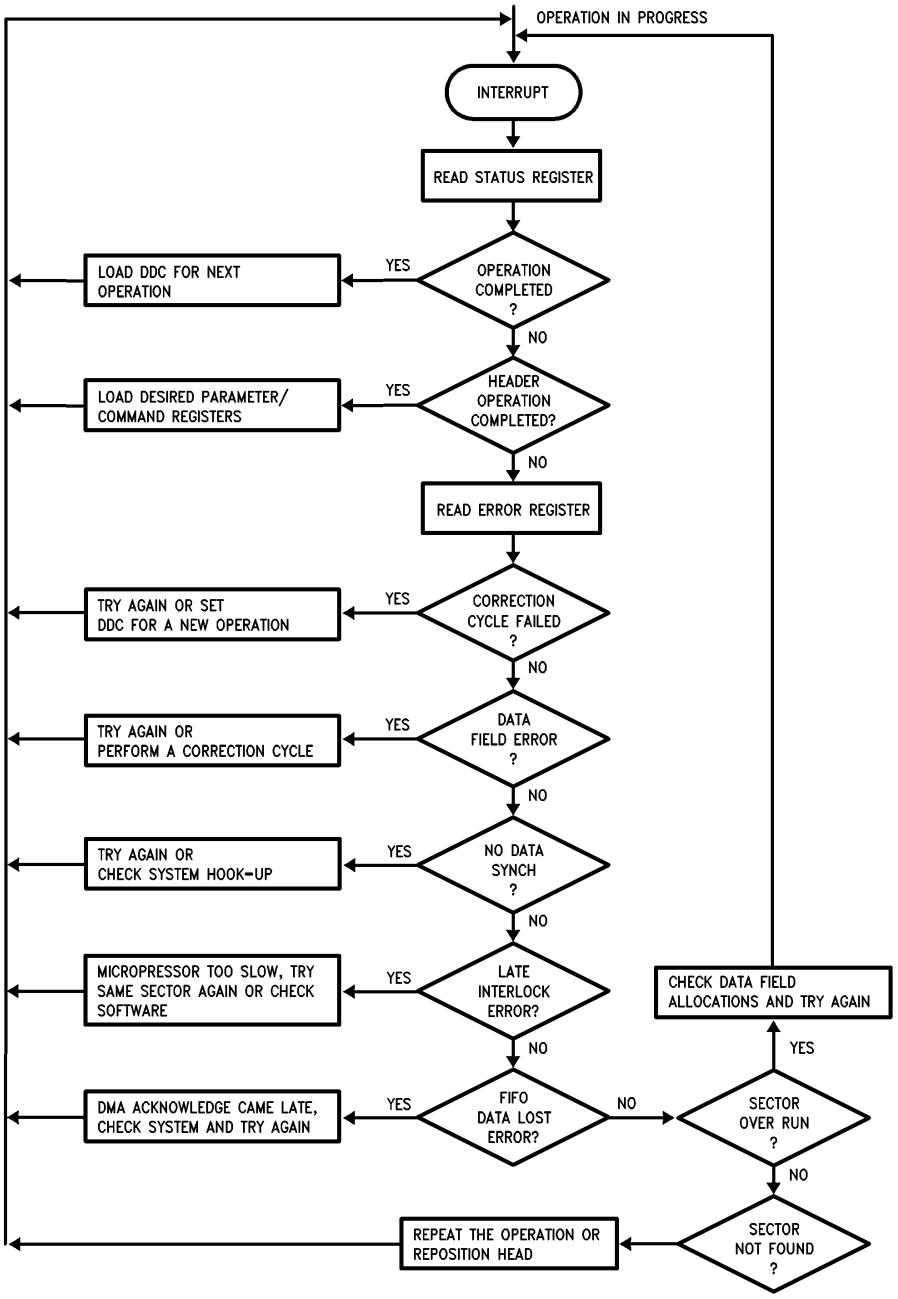


FIGURE 7.22 Interrupt Servicing

7.8.2 The Header Failed Although Sector Matched (HFASM) Function

The Header Failed Although Sector Matched, HFASM function can be used to perform some disk maintenance and diagnostic functions. The HFASM function (pronounced H-fazzzm) has been described very basically in section 5.1.1 and 5.1.3. In this section, an attempt is made to provide a more detailed description and some example uses.

The HFASM function is essentially an Error that can be enabled by setting the EHF bit in any one or more Header Control Registers. When this bit for any header byte is enabled, and a Compare Header Operation is performed, the HFASM Error will be generated if certain conditions are met. The Error is generated if a header byte pattern register matches with its disk header byte, and that header byte had its EHF bit, and any other byte in the header fails to match. If multiple header bytes have been enabled only one need to match, while *any* other header byte does not match in order to generate an HFASM error. The other header bytes may or may not have their EHF bit set. Thus, this error can tell the system when a particular type of header has been found, even though the exact header did not match.

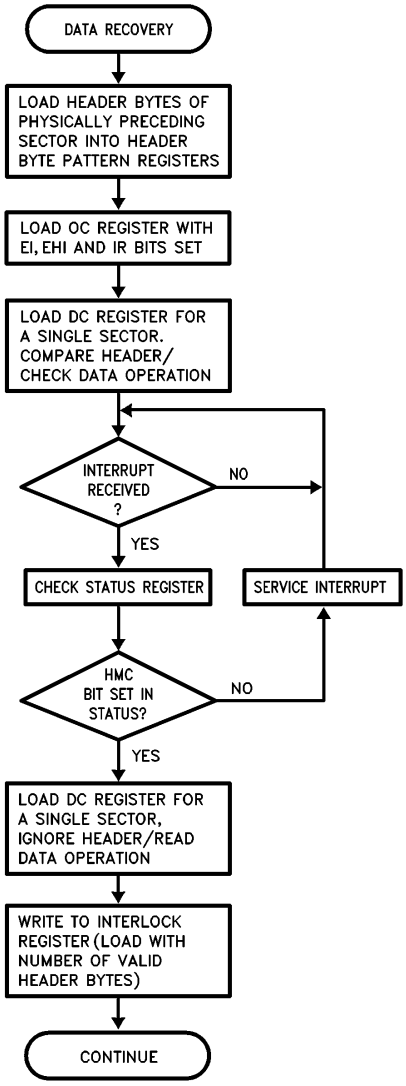
The HFASM error is generated only when execute a command that has a Compare header operation. Write Header, Read Header and Ignore Header operations will not generate an HFASM error. If a Compare Header Operation, and a Check Data Operation form the command is executed and an HFASM error is generated, no data is transferred to the system, but the DDC will load the Header into the FIFO. If a multi-sector operation was in progress the HFASM Error being set will terminate the operation. If the HASM Error is set, but the sector has a CRC error as well, the DDC will terminate the command with both bits set.

This command can be used for various tasks. For example, if the sector's sector number byte has the EHF bit set for all read and write operations, no HFASM error should occur. If one does occur then the system knows that the correct sector number was found, and the reason the correct sector was not found was because of a seek error (head on wrong track); the header was marked bad (and the DDC is looking only for good headers); the wrong head was selected; or some other header parameter was incorrect.

Another example suppose that the header byte that is designated as the sector's sector number has its EHF bit set, as before. The system wants to find sector one, but does not know the other header information. If a Compare Header-Check Data Command is executed, an HFASM error the FIFO will be loaded with the actual header that has a sector number of one. The system can then determine what the status/flag information is (is it a bad sector or a good one etc?) or which cylinder the head is on.

A third possibility, would be to find specific sectors that might have their flag header bytes indicating a bad sector. The EHF bit of the header byte designated for the flag should have its EHF bit set. In this case if a Compare header-Check data is performed using the interlock mode and starting on the sector, all the bad sectors can be identified.

In general the HFASM function is a subtle but powerful tool to enable some diagnostics, and provides to a limited degree the ability for a user definable error condition.



TL/F/8663–G2

FIGURE 7.23. Data Recovery Using Interlock Feature

APPENDIX: DISK DATA CONTROLLER COMMAND FLOW CHARTS

The design guide has covered the general ways that the DDC can execute the various operations. However, it is impossible to account for all possible design situations in the guide. To attempt to provide information which the designer can utilize to determine how the controller might behave in various situations, the figures in this section outline the command flow for the disk operations. All header and data operations are covered, and DMA operations are excluded except to the extent that information must be loaded/unloaded from the FIFO to ensure no overflow or underflow occurs. Other than this DMA operation occurs independently (and concurrently) to the actual disk operations.

The command flow that the DDC executes is divided into header and data operations. In addition to this there are several operations that are performed concurrently, these are shown in the flow charts as special sections and are labelled as concurrent operations. *Figures A.1* to *A.13* cover the header operation and *Figures A.14* to *A.18* cover the data operations. These are described below.

A.1 START OF COMMAND

After a reset, the DDC is in the standby mode waiting for a command. In this mode the DDC is waiting for a command, and once a command is loaded the controller starts its operation. The first major task is to start the DMA transfer to the FIFO if a write data operation is desired. This will start a concurrent task for the DMA controller that will be executed throughout the write data operation. Note that this is shown in *Figure A.2* as a separate dotted outlined block. Otherwise the rest of *Figure A.1* sets up for the header operation, including when to start this operation.

A.2 WRITE HEADER OPERATION

In *Figure A.3* a Write Header operation is assumed. The first test is whether the data output is to be MFM or NRZ. If MFM, the precompensation may be enabled if NRZ, *Figure A.4* is executed. The first steps taken by the DDC are to write the preamble, address mark bytes, and the sync bytes if desired. Also, if the CRC/ECC is to encapsulate both header and sync fields than it is turned on, otherwise it will be turned on later.

Figure A.4, A.5, and *A.6* complete the write header operation. First the CRC/ECC is enabled, then the header bytes are written. There are various options on how to do this and they are shown in this figure. Finally the CRC/ECC fields are written to the disk. This includes the internal one, and/or an external field if desired. The last task is to write the header postamble field.

At entry point D, a standard end to the header operation is shown. This is used by other header operations. Here the header match bit is set and if interrupts are enabled, the Header completion interrupt is issued. If the operation is a multi-sector operation, then the Number of Sector Operations counter and the Start Sector register are updated. If the last operation then the DDC is ready for the next command and will start the data operation.

A.3 NON-WRITE HEADER OPERATION

For a read or ignore header operation, the entry point in *Figure A.7* is H. First the operation waits to start based on the internally programmed mode and the receipt of an index or sector pulse. Then Read Gate is asserted, and the DDC searches for the sync or address mark fields. Once all the sync and address mark fields match, the DDC can perform the Header operation. *Figure A.5*. If encapsulation of the CRC/ECC was enabled then CRC/ECC calculation will begin at the address mark field.

If not already active, the CRC/ECC is activated, and the DDC starts operating on the header information. If the data operation is a check data or the header operation is a read header (the later is shown in *Figure A.11*), then the header bytes are loaded into the FIFO. If the operation is a compare header, the header bytes are compared to the registers or start sector register. Next the CRC/ECC is checked. If it is in error then the header fault flag is set, however the operation is not aborted. If there is a CRC/ECC error or the header did not match, the DDC then looks for the next sector. If the correct header is found then, the DDC will deassert read gate to avoid the write splice, and will proceed to execute a data operation.

A.4 WRITE DATA OPERATION

For a write data operation, the write gate is asserted, and an algorithm similar to the write header operation is executed. MFM or NRZ data output is configured, and the start with address mark mode is checked. Address mark, preamble and sync fields are written *Figure A.14*. If needed the CRC/ECC generator is enabled when the Address Mark is written.

Just prior to writing data, the CRC/ECC will be enabled, if it is not already. Then data is written to the disk from the FIFO or the Format pattern register. Once the data is written, the CRC/ECC is written, followed by the data postamble. If the operation is not a format operation then write gate is deasserted, and interrupts may be generated. If the operation is a format operation, the post sector gap is written, as shown in *Figure A.16*.

The operation then is completed and the DDC will check to see if the command was a multisector one or if a new command has been entered. If so then a new command is started immediately, and this is shown by jumping to entry point V in *Figure A.1*.

A.5 NON-WRITE DATA OPERATION

The flow chart for a read data or check data operation is shown in *Figure A.17*. After the header postamble, read gate is asserted. The DDC first configures when the CRC/ECC calculation should begin. It then begins checking the address mark and sync fields. If an error occurs the operation is aborted. Otherwise if the operation is a read data, the data is sent to the FIFO, and the CRC/ECC is checked.

If the operation is a check data command, the data field is not transferred or checked, and the DDC just counts this field and checks the CRC/ECC. At exit point U on *Figure A.18* the end of the data operation jumps to *Figure A.16*, where the interrupts may be generated if enabled and the DDC checks for another command.

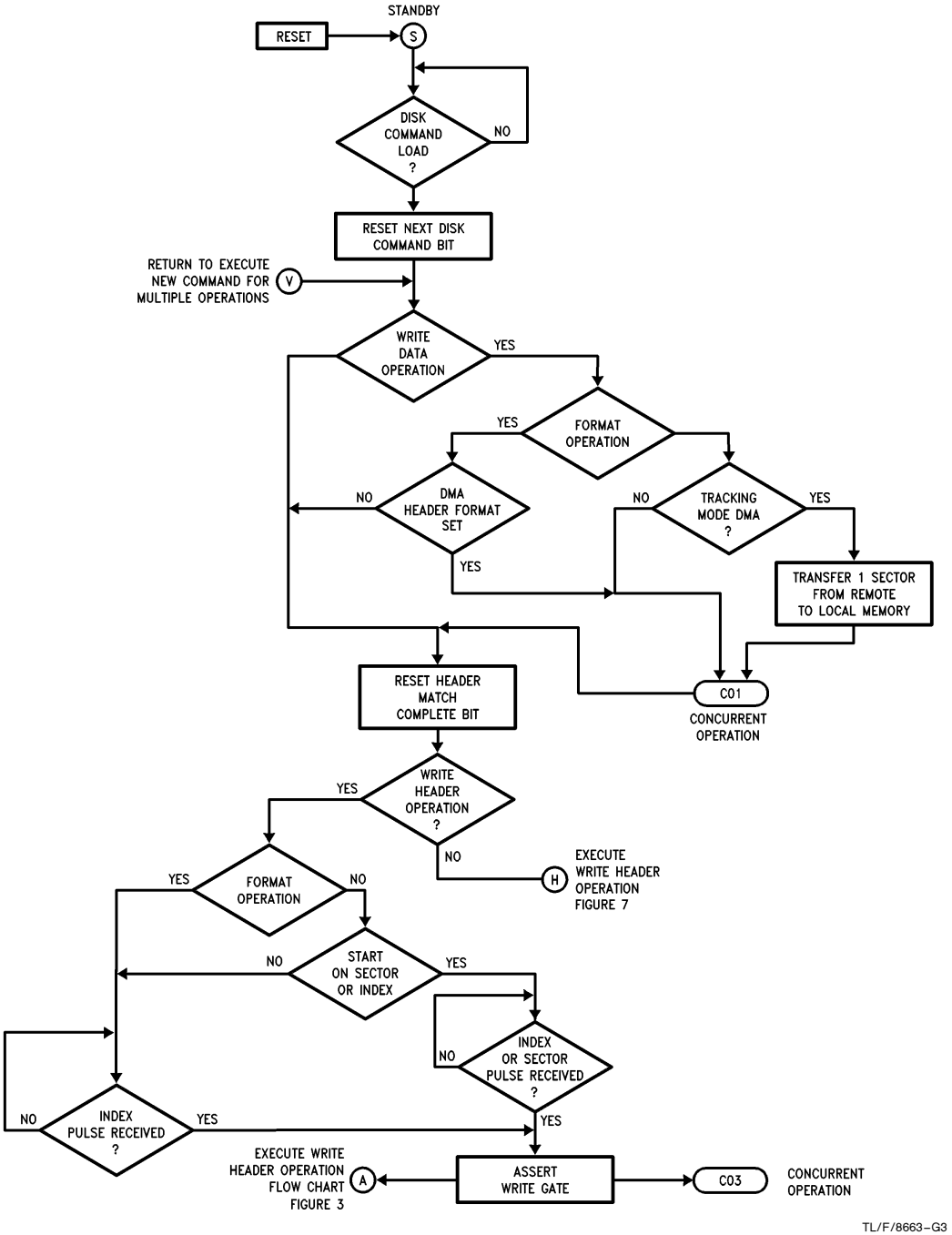
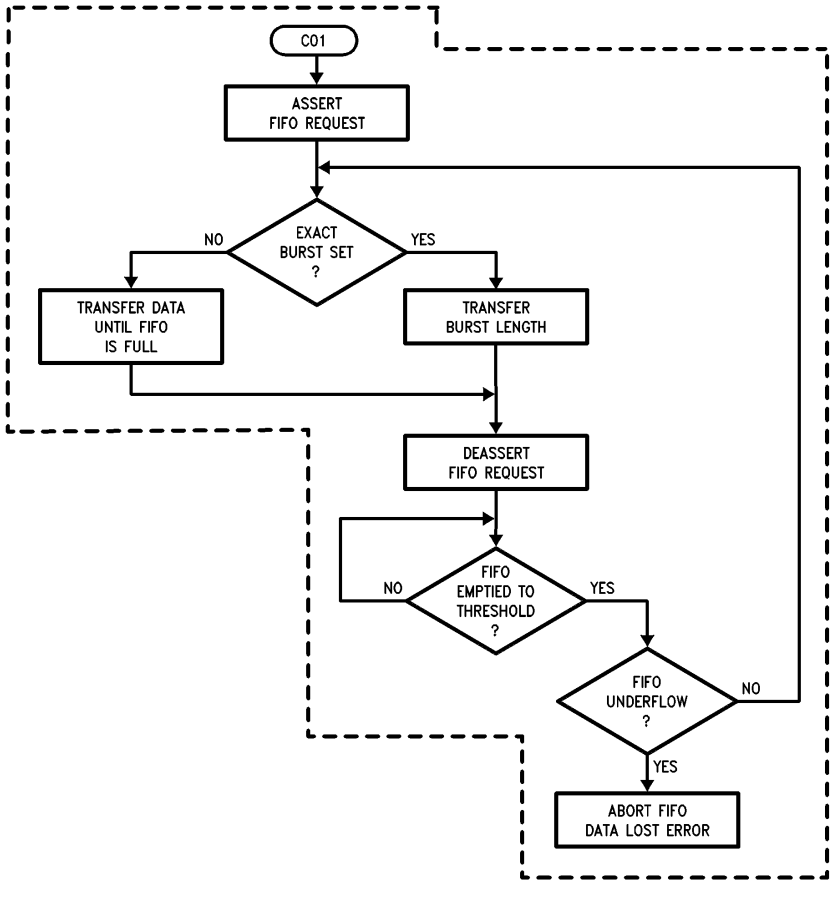
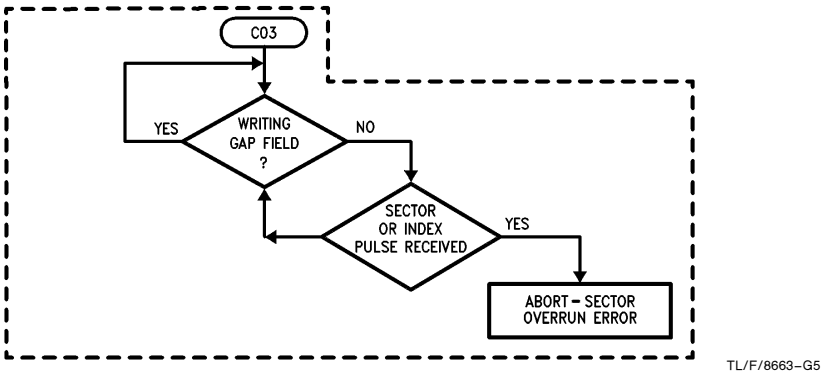


FIGURE A.1. Flow Chart for Start of a Command

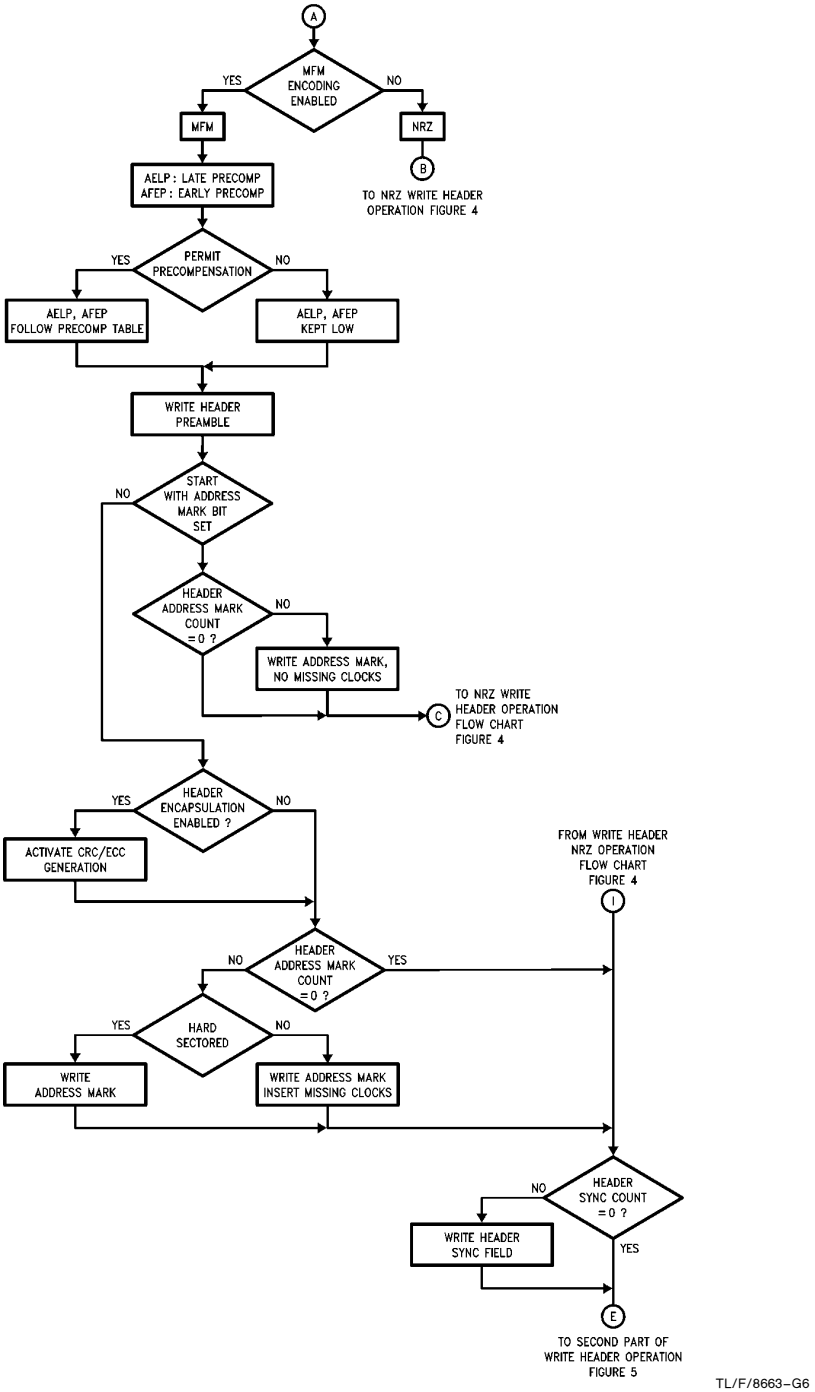


(a) Concurrent Operation 1



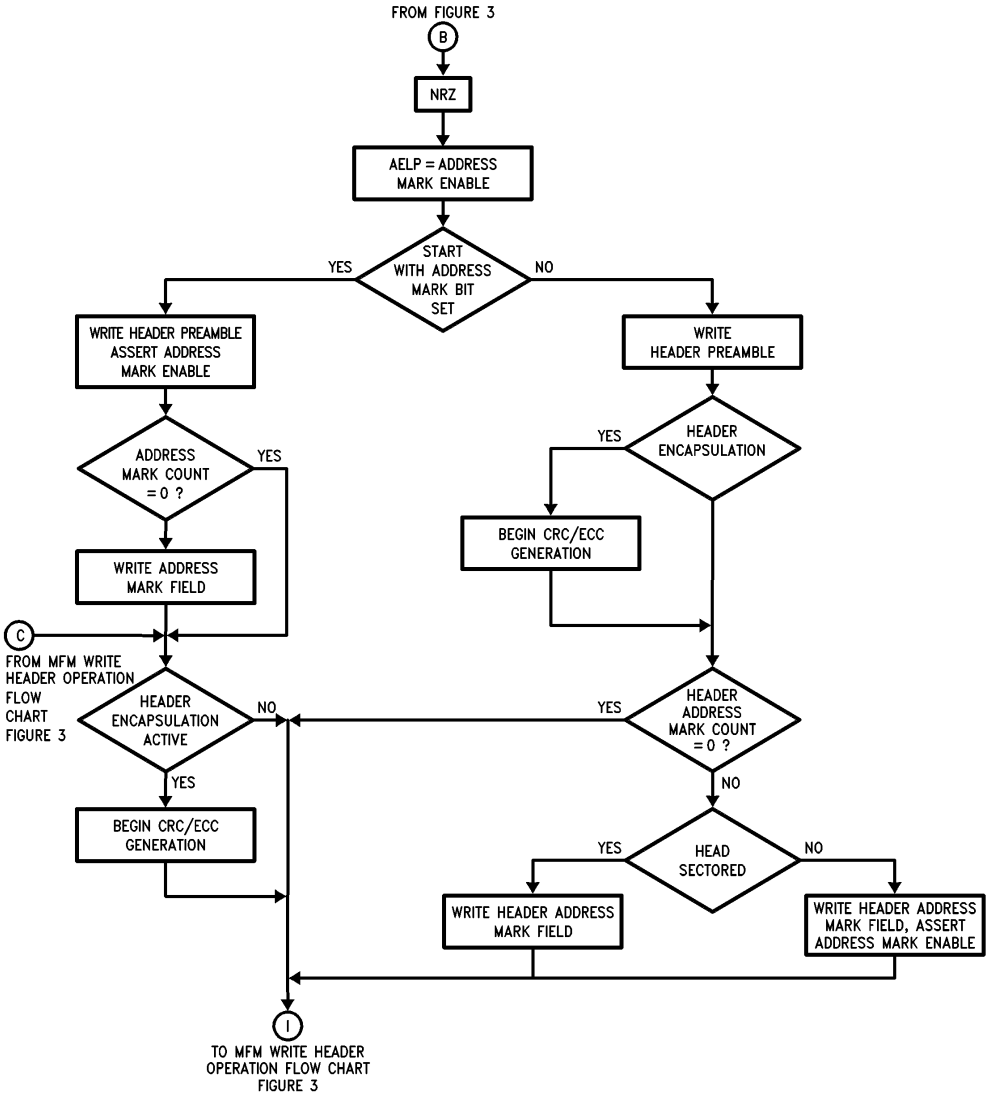
(b) Concurrent Operation 3

FIGURE A.2. Concurrent Operations for the Start of a Command



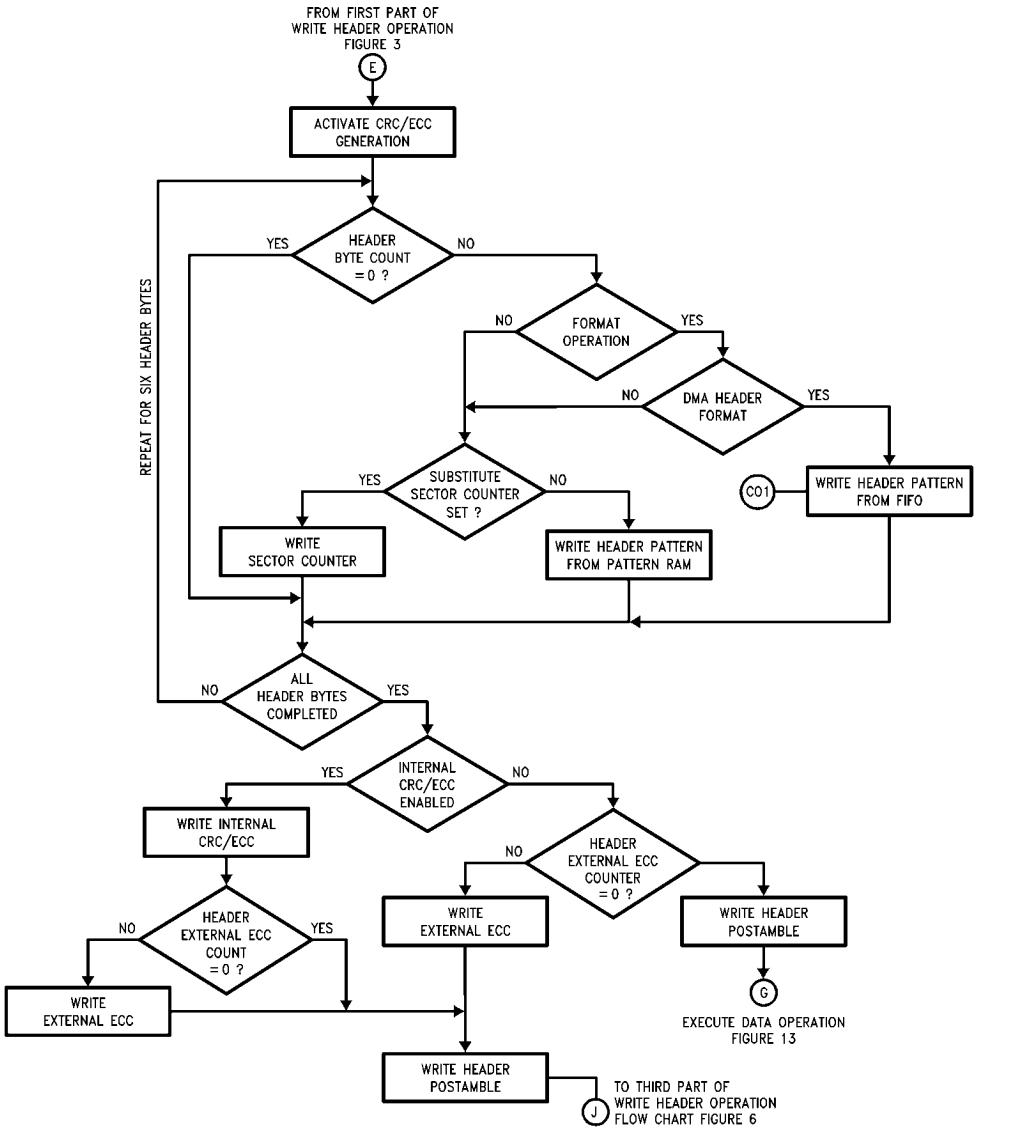
TL/F/8663–G6

FIGURE A.3. Flow Chart for First Half of Write Header Operation



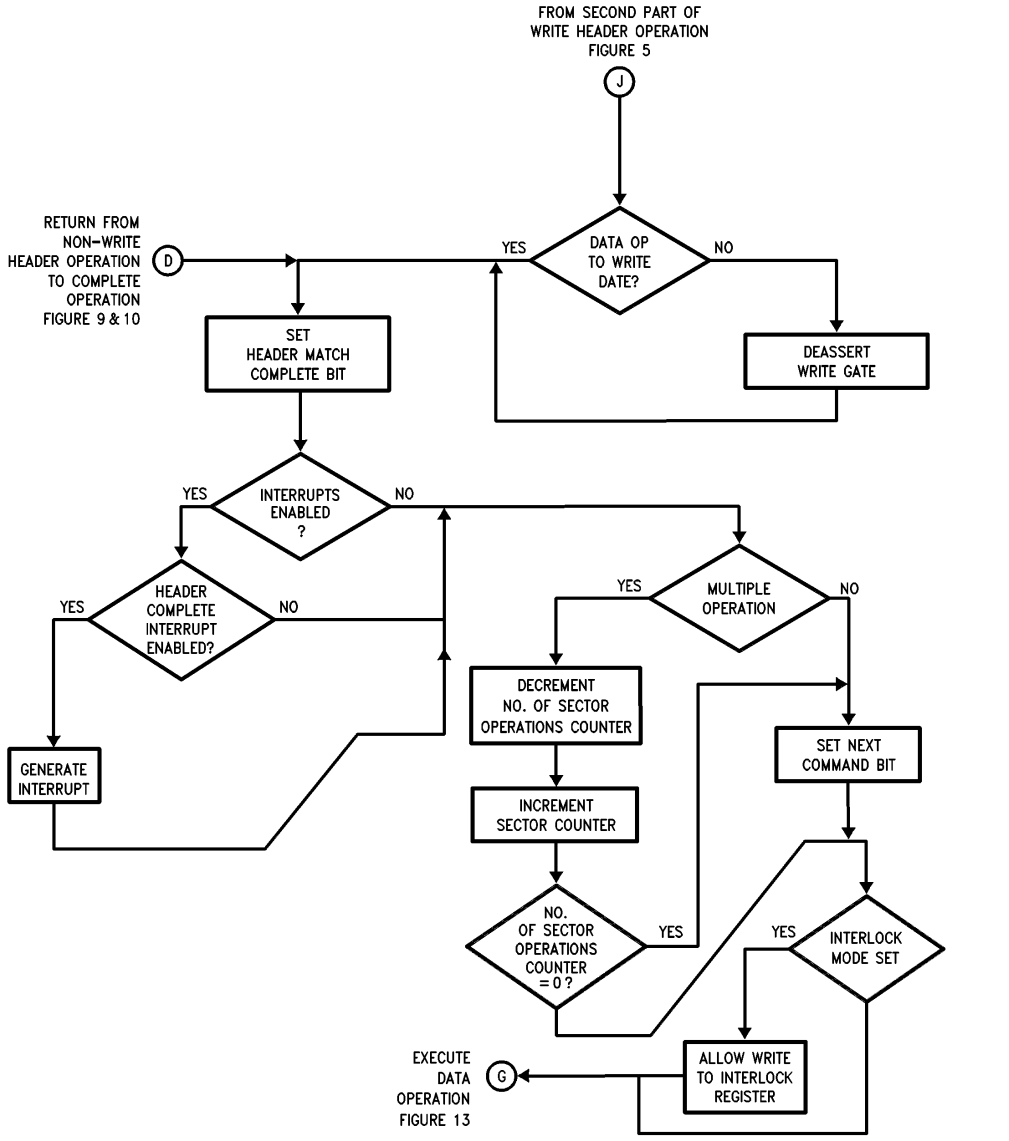
TL/F/8663–G7

FIGURE A.4. Flow Chart for First Half of Write Header NRZ Operation



TL/F/8663–G8

FIGURE A.5. Flow Chart for Second Part of a Write Header Operation



TL/F/8663–G9

FIGURE A.6. Flow Chart for Third Part of a Write Header Operation and Ending Sequence for Other Header Operations

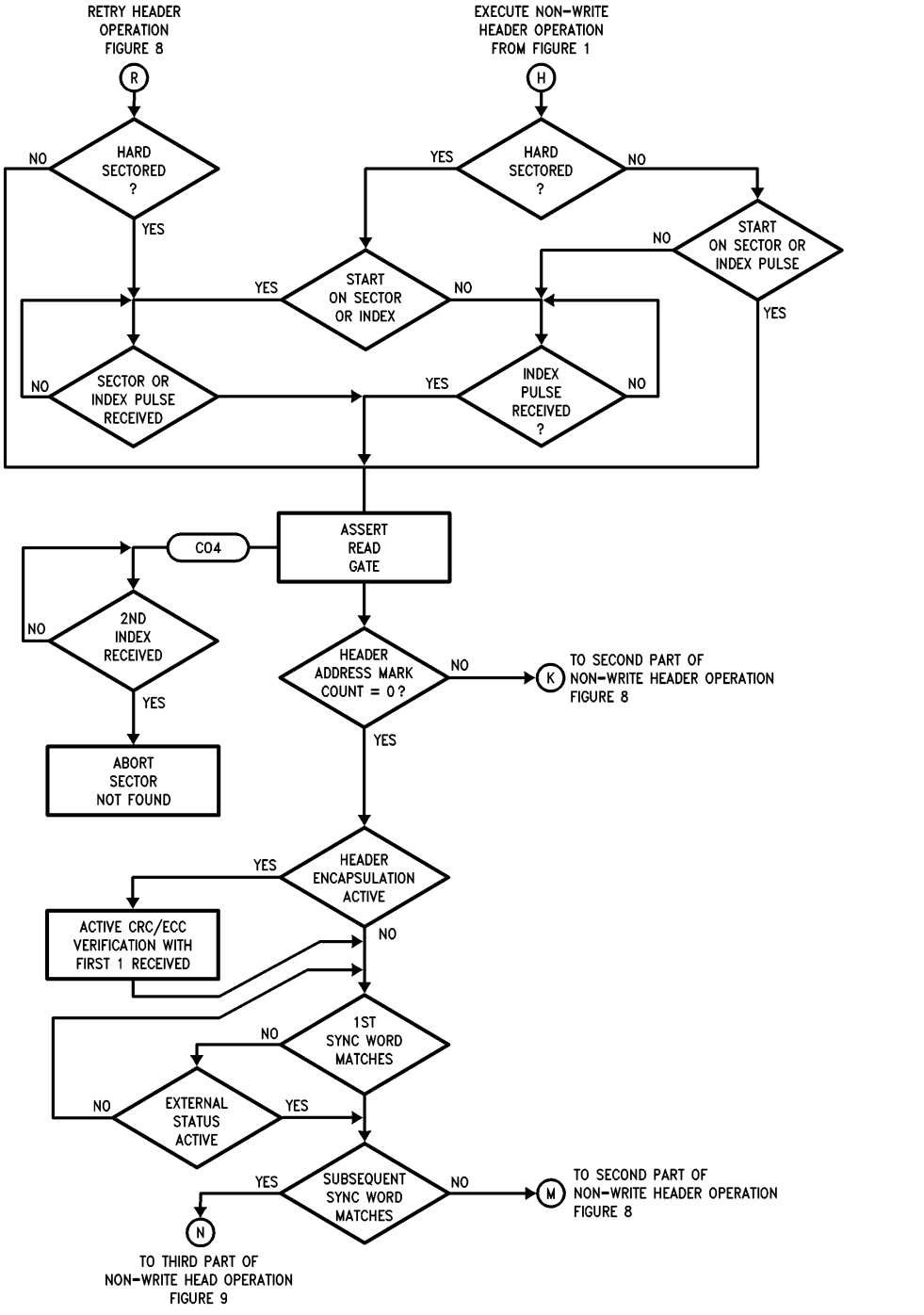


FIGURE A.7. Flow Chart for First Part of Non-Write Header Operation and Retry Header

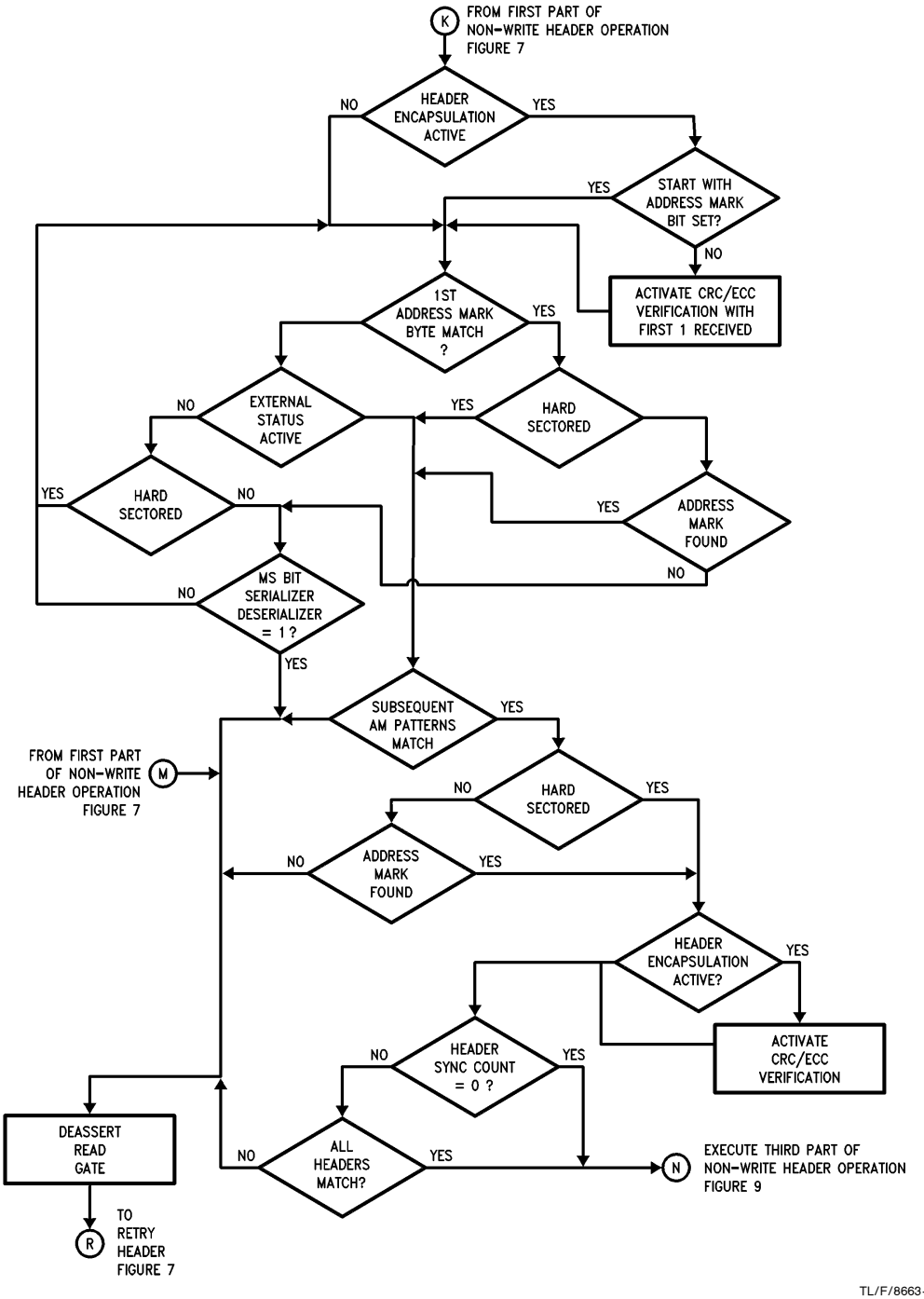


FIGURE A.8. Flow Chart for Second Part of Non-Write Header Operation

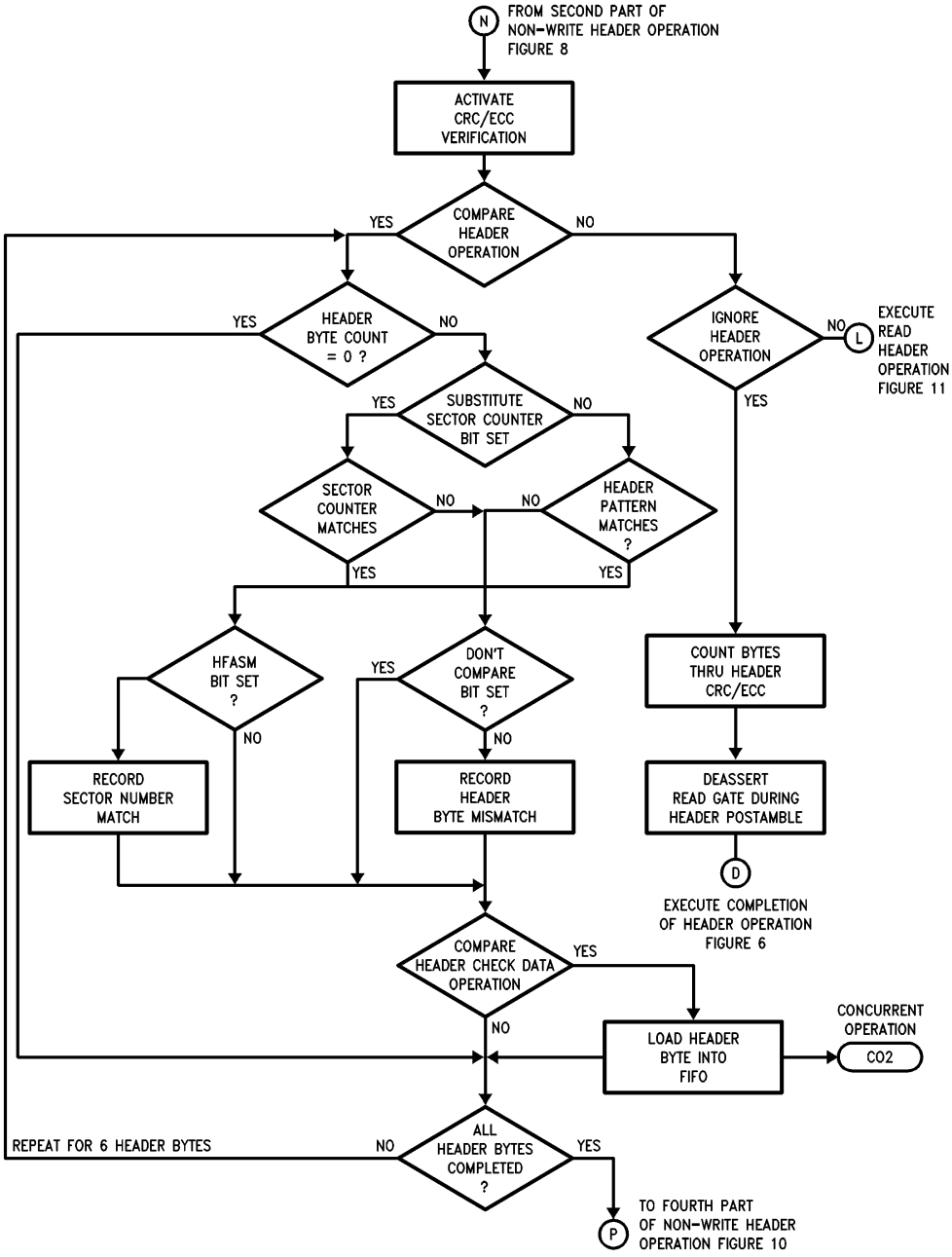


FIGURE A.9. Flow Chart for Third Part of Non-Write Header Operation (Except Compare Header)

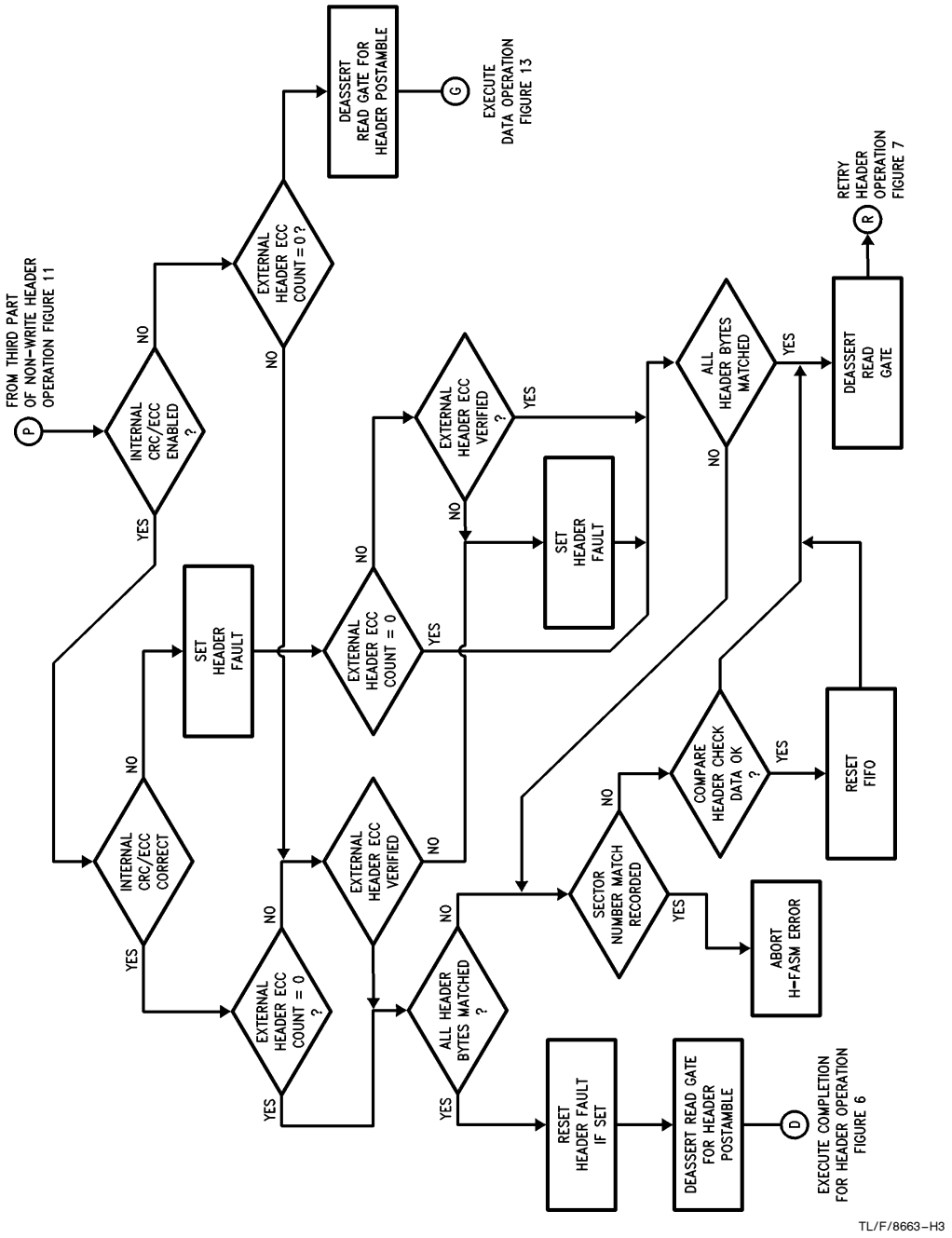
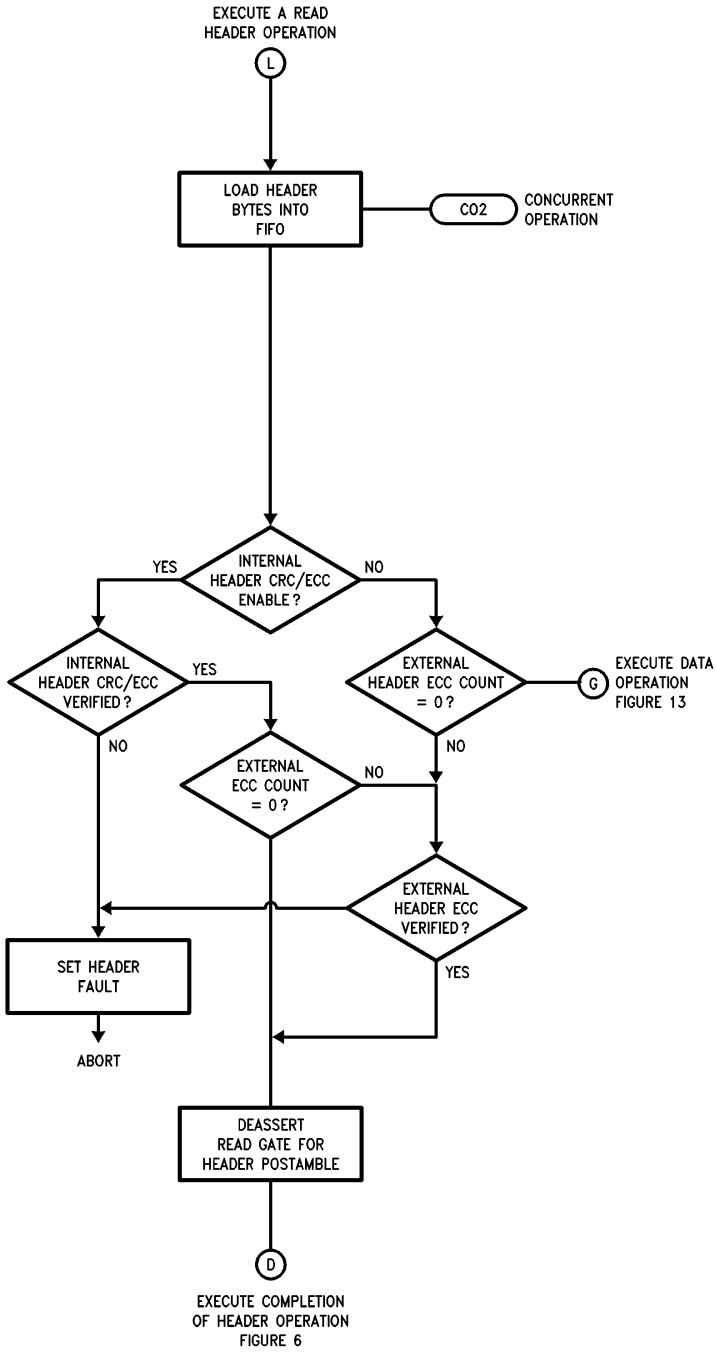
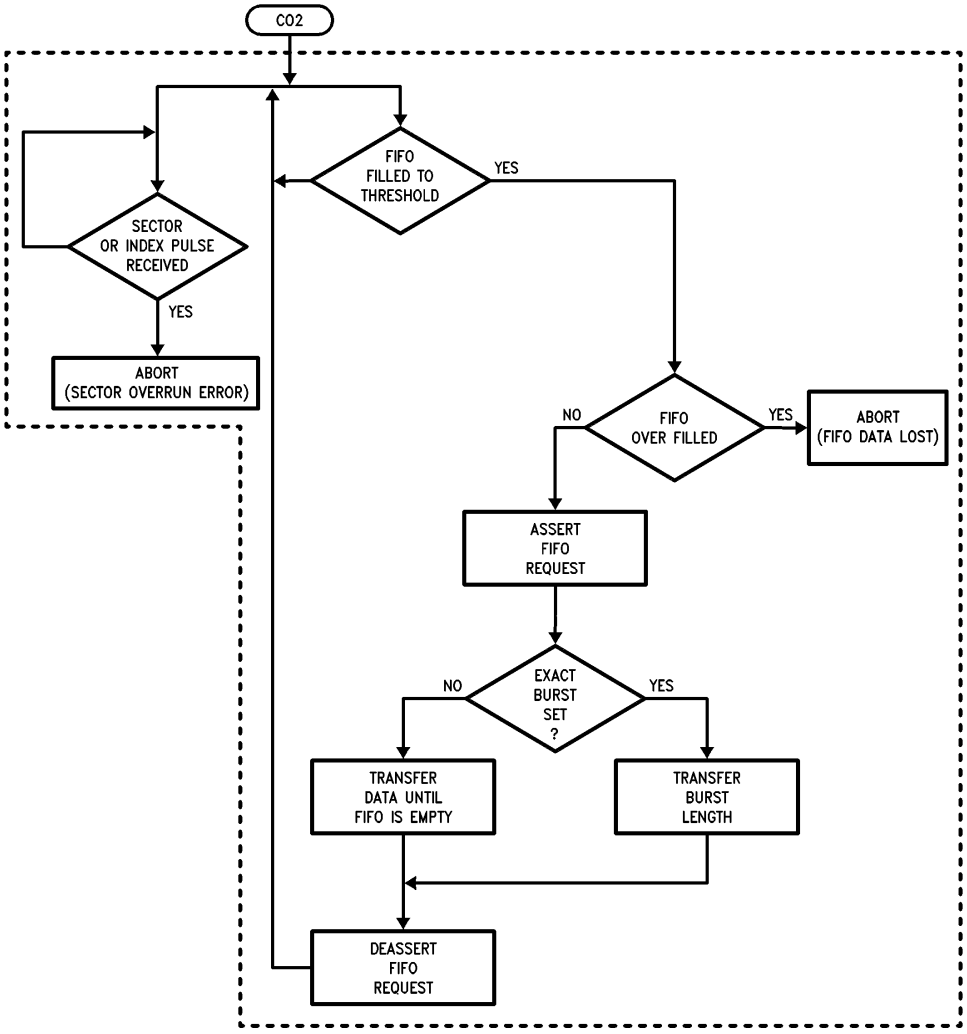


FIGURE A.10. Flow Chart for Fourth Part of Non-Write Header Operation (Except Compare Header)



TL/F/8663–H4

FIGURE A.11. Flow Chart for Compare Header Operation



TL/F/8663–H5

FIGURE A.12. Flow Chart for Compare Header Concurrent Operation

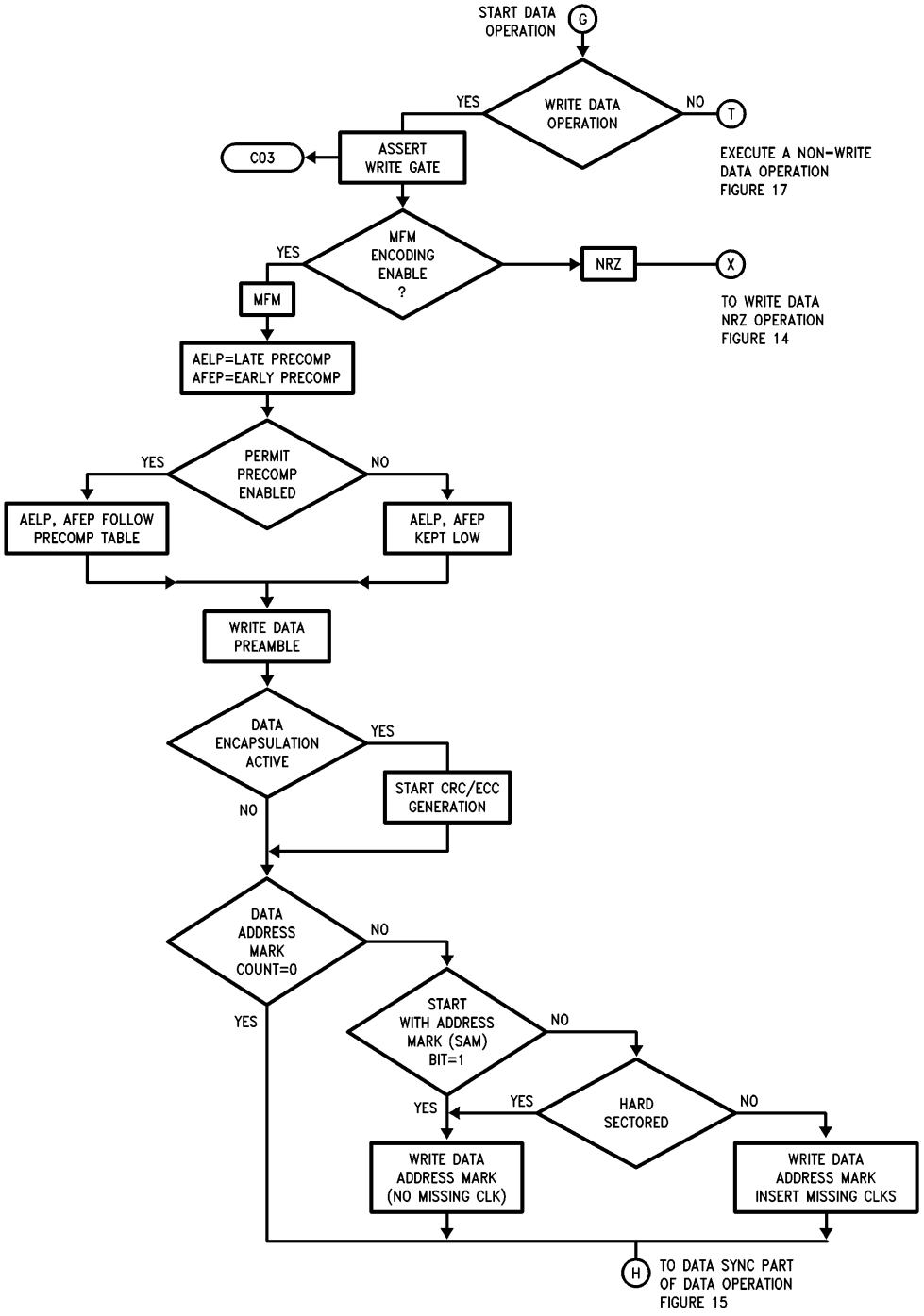


FIGURE A.13. First Part of Data Operation Flow Chart, MFM Mode (Up to Data Sync Field)

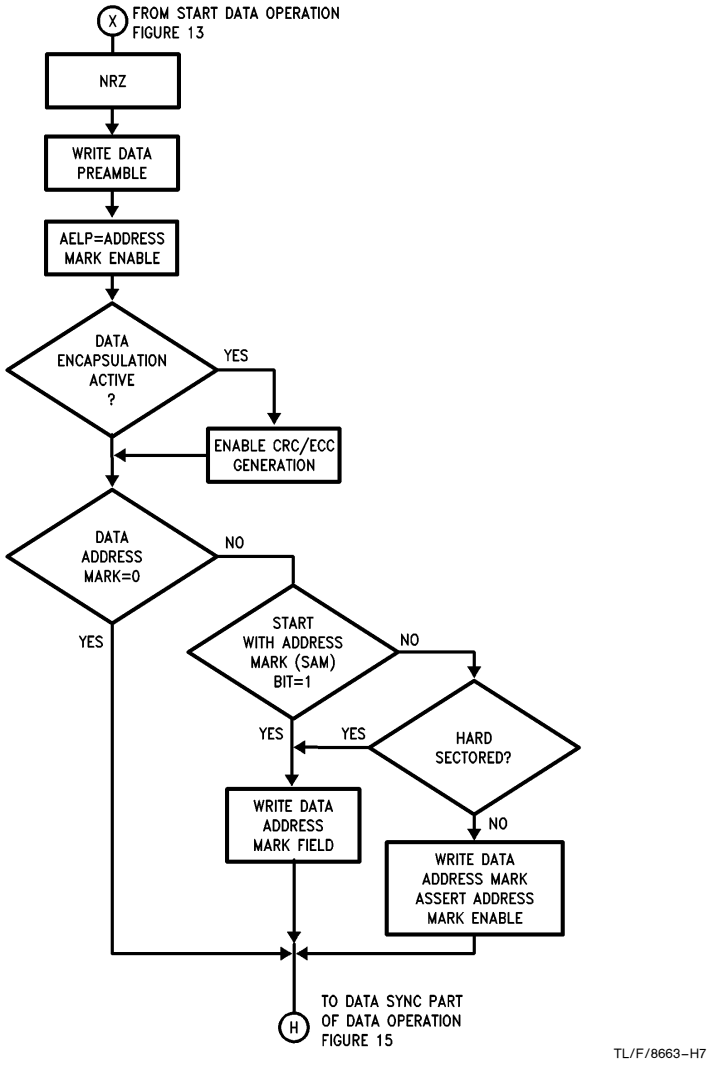


FIGURE A.14. First Part of Data Operation Flowchart, NRZ Mode (Up to Data Sync Field)

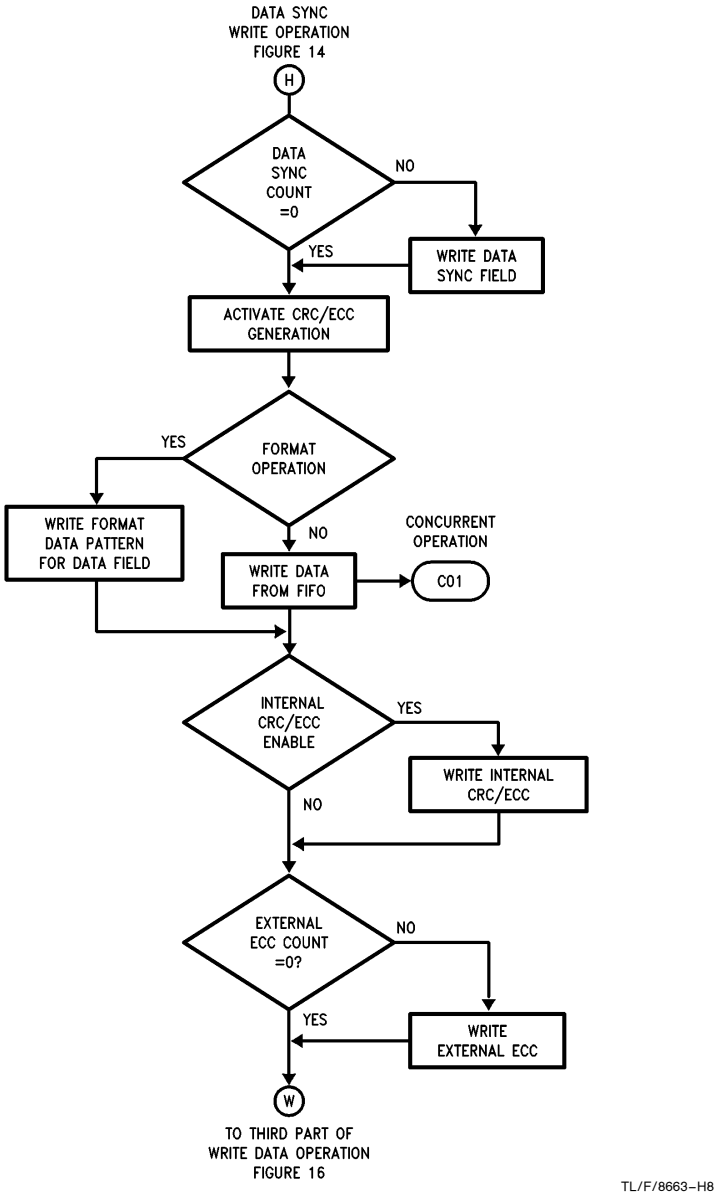


FIGURE A.15. Second Part of Data Operation (for Write Data Operation)

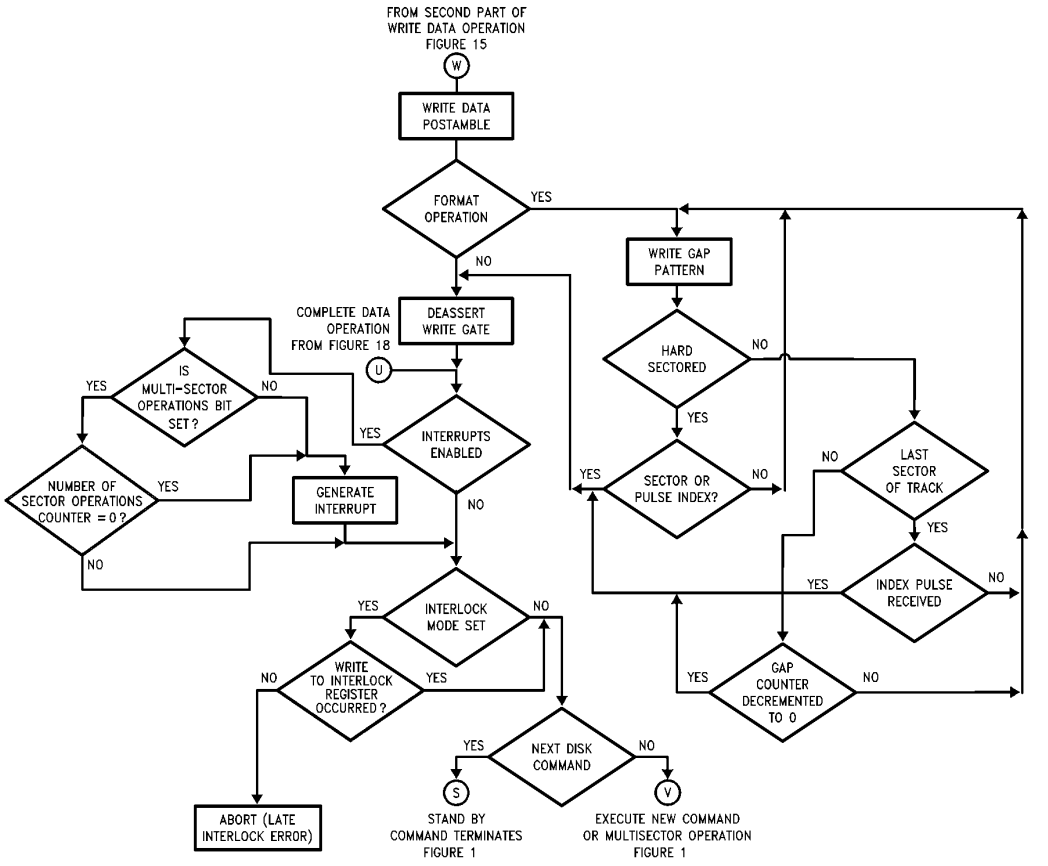


FIGURE A.16. Flow Chart for Third Part of Write Data Operation

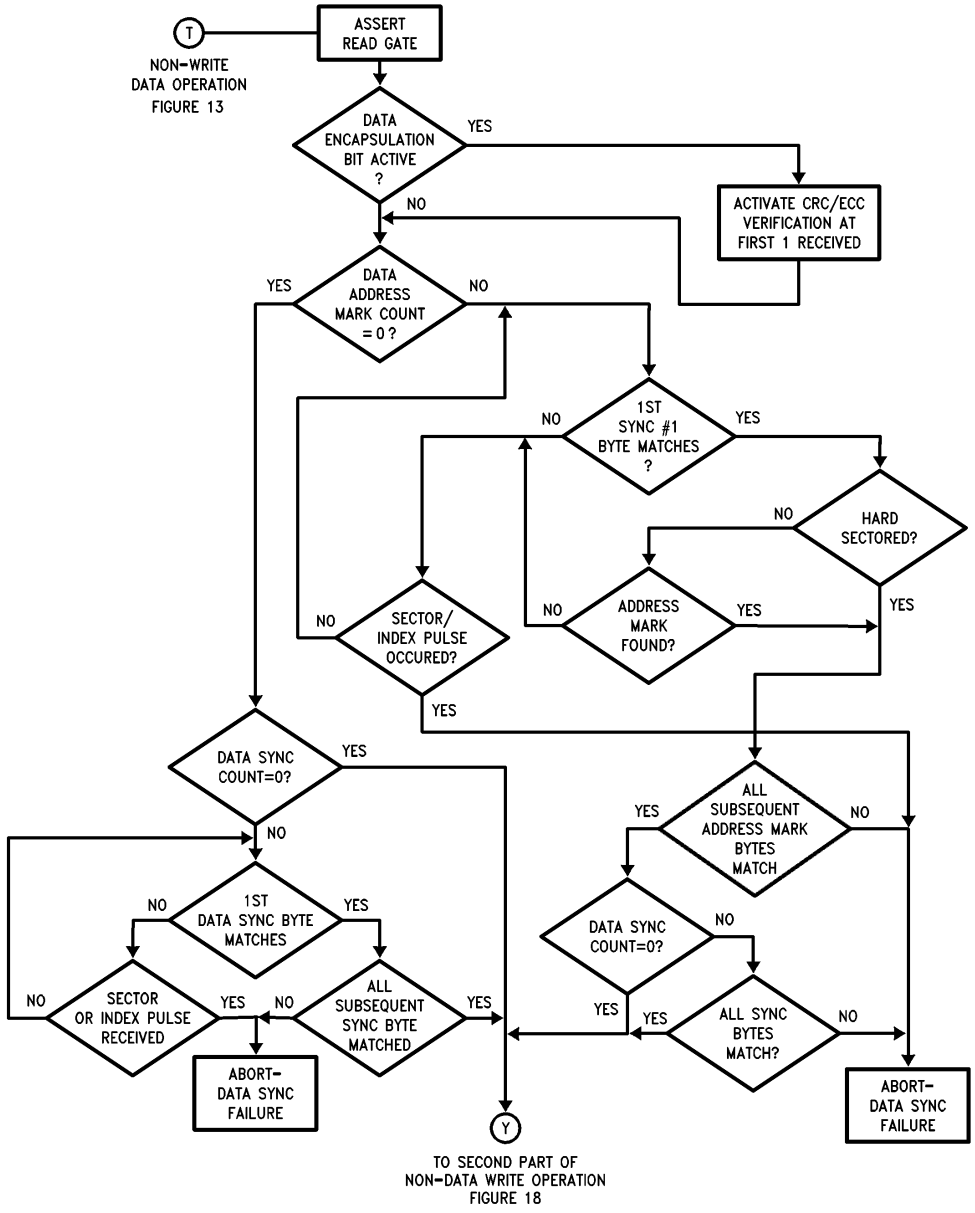
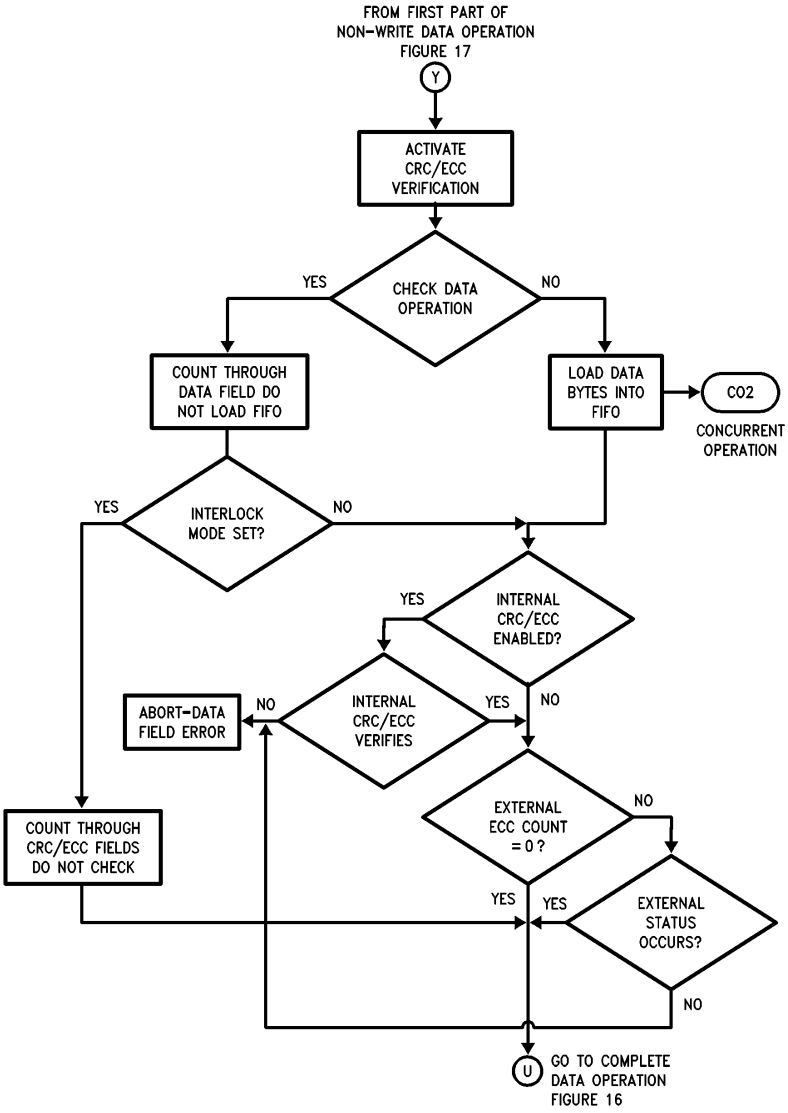


FIGURE A.17. First Part of Data Operation Flow Chart for Non-Write Operation



TL/F/8663–I1

FIGURE A.18. Second Part of Data Operation Flowchart for Non-Write Operation

Table of Contents

Table of Contents (Continued)

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265