ISM ASIC SPECIFICATIONS

Revision 4.1
July 2, 1987

Apple Computer, Inc.

TABLE OF CONTENTS

I. INTRODUCTION

The ISM is an integrated disk controller chip which is designed to read and write MFM and Apple GCR format disks. The ISM also has provisions for reading and writing other formats. The ISM contains the following features:

- **Supports Standard MFM Format**
- **Supports Apple GCR Format**
- **Write Pre-Compensation**
- **Read Post-Compensation**
- **Asymmetry and Speed Error Compensation**
- **Programmable Parameters for using both Multi and Fixed speed drives**
- **Two Byte Data FIFO**
- **Motor Time Out**
- **Programmable Phase Lines**

The ISM uses a programmable parameter scheme which makes it possible to Read and Write 3 1/2 inch variable and fixed speed drives, as well as standard 5 1/4 inch drives.

The ISM makes it possible to Read and Write both MFM and Apple GCR formats on the same disk drive, and also makes it possible to Write MFM format on a 3 1/2 inch variable speed drive in such a way that it can be read back on fixed speed 3 1/2 inch drive.

The ISM provides the ability to do Write Pre-Compensation to correct for Peak Shift effects which occur in magnetically stored media.

The ISM also provides a very sophisticated, and rarely used, form of Read Post-Compensation which corrects for Peak Shift effects on disks with insufficient Pre-Compensation.

The ISM contains a two byte Read and Write FIFO to provide more Software flexibility.

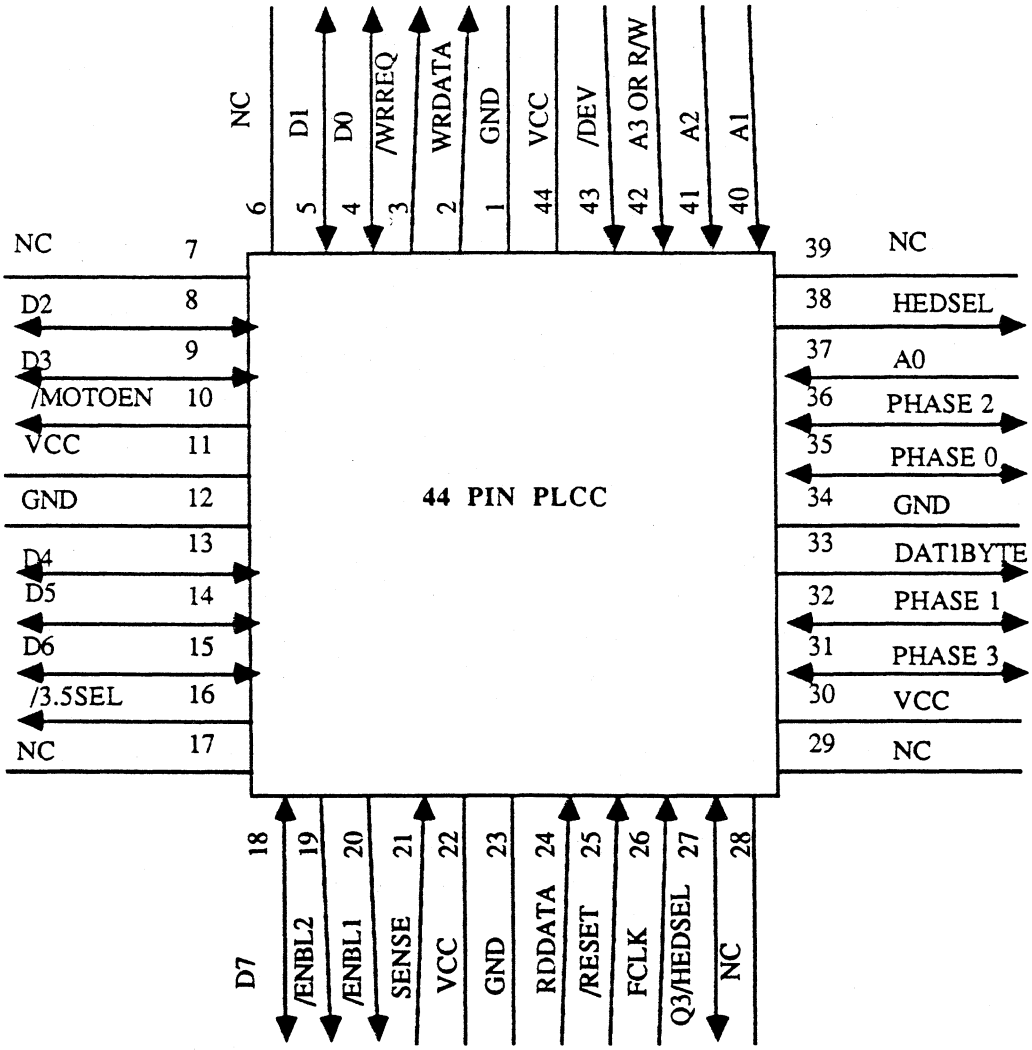
A Motor Time Out is included which will keep the drive enabled for 1/2 s to 1 s to provide time for Software to begin another Read or Write operation without bringing the drive back up to speed.

The ISM makes it possible to program the Phase Lines as either inputs or outputs which make it possible to interface with a wide variety of drives.

II. PIN DESCRIPTION

The ISM will be packaged in a 44 pin PLCC package. The figure below shows a package with the signal names used and the required pinouts.

ALL INPUTS TTL.



PHASE 0 - PHASE 3 - These lines can be programmed to be either inputs or outputs, and are used for communication with the disk drive.

SENSE - This line is used to read the write protect status from the drive.

HEDSEL - This output is used to select the side to write to or read from if it is a double sided drive.

WRDATA - This line contains the data to be written on the disk.

/WRREQ - This line is used to tell the drive to start accepting data from the Wrdata line.

/ENBL1 - This line is used to enable drive number 1.

/ENBL2 - This line is used to enable drive number 2.

/MOTORENABLE - This line is used to indicate that either the motoron bit (bit 7 of the mode register) is set or the timer is timing out.

RDDATA - This line contains the data being read from the drive.

FCLK - This is the clock input to the chip.

D0 - D7 - These lines contain the data which is read from or written to the chip by a processor.

A0-A2 - These lines are used by a processor to tell the chip which register to read or write data from.

A3 OR ARW - This line is used to tell the chip whether the processor is reading or writing information to the chip.

/DEV - This line is used to select the chip for either reading or writing data from the processor.

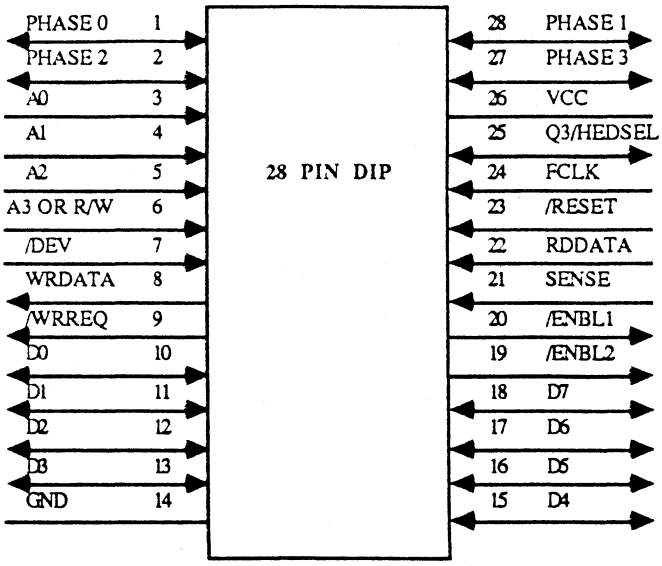
/Q3/HEDSEL - On reset this signal will appear to be a Q3 input and is used to latch the data. This is necessary on systems in which the data is not valid at the rising edge of /DEV. If this signal is not needed for latching the data it can be used as HEDSEL pin by setting bit 0 of the Setup register to a 1.

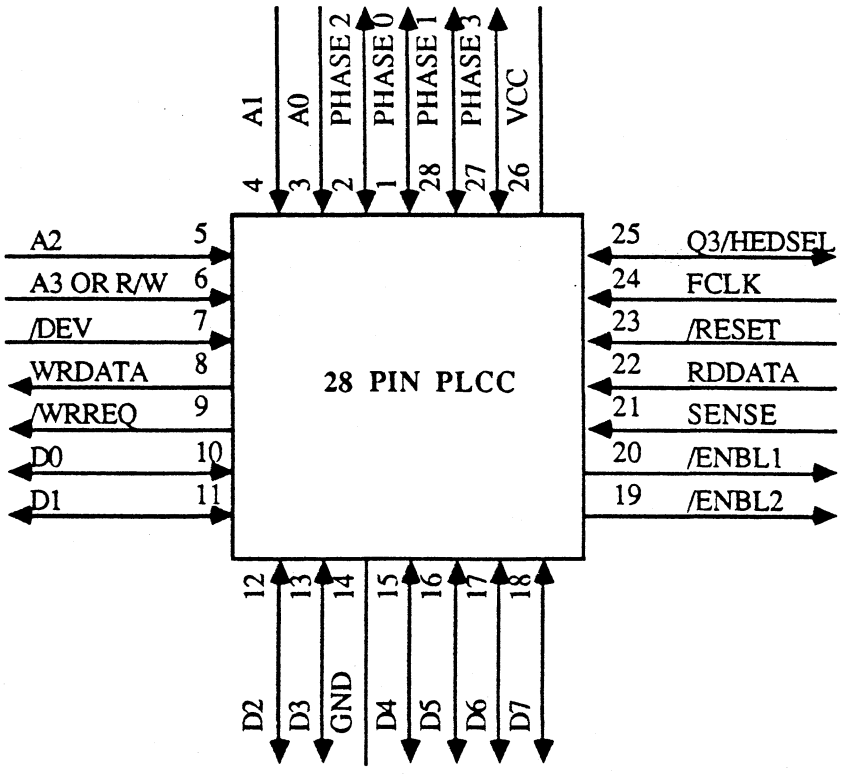
/RESET - This line is used to initialize the registers in the chip and to reset any current operation the chip might be in.

/3.5 SEL - This signal is used as an extra select line for more expandability.

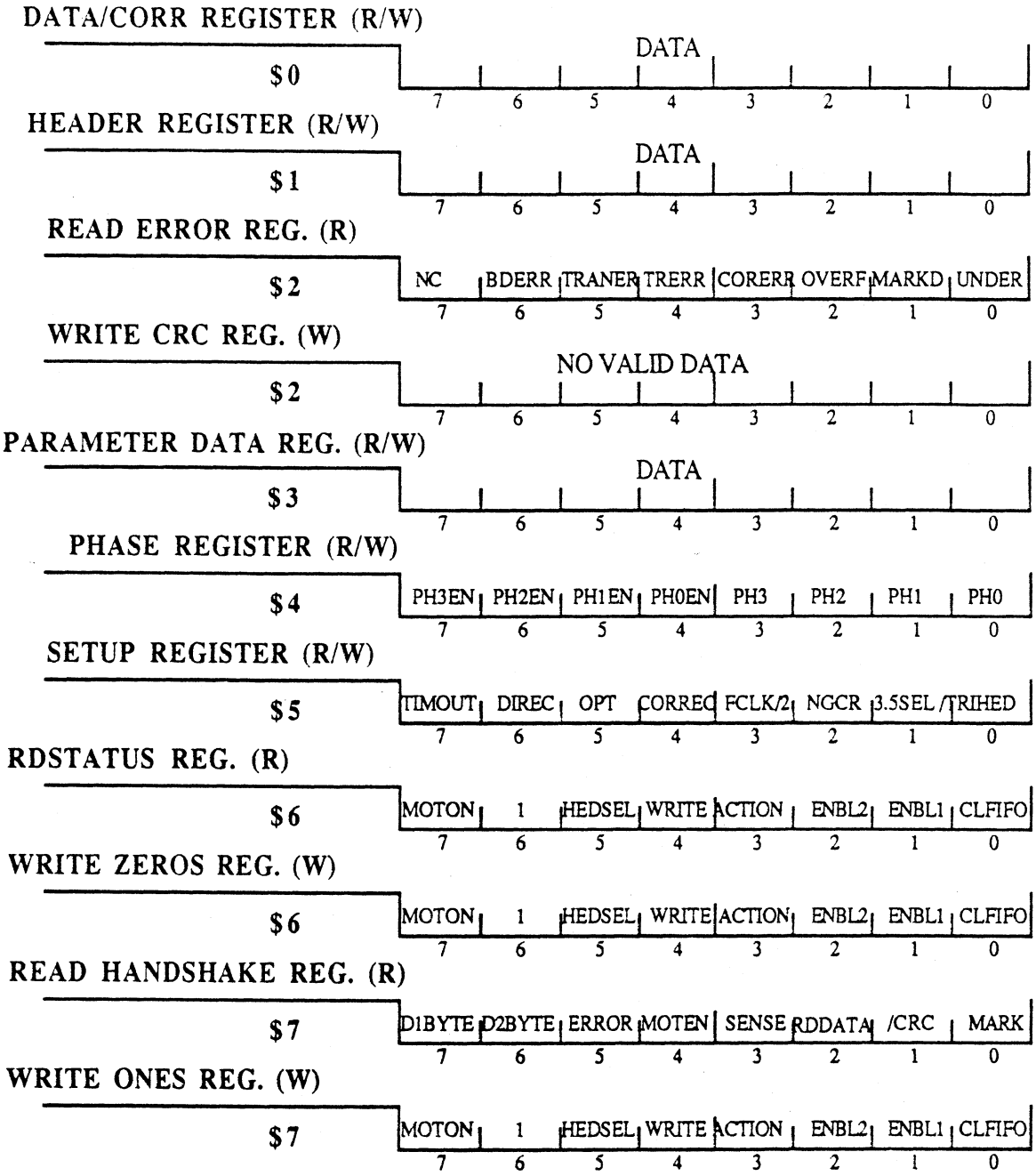
DAT1BYTE - This is the same signal as bit 7 of the Handshake register. This signal is brought out for possible future applications.

OPTIONAL PINOUT





ISM ADDRESS MAP



III. THEORY OF OPERATION

1. Introduction to MFM.

In order to understand how the ISM works it is important to understand what the problems are with reading from and writing data to magnetic disk media. Data is written on a disk by reversing the direction of the magnetic field of the R/W head which produces a magnetic transition on the disk. Reversing the magnetic field of the R/W head is accomplished by toggling the Write Data line, on the drive, from high to low or from low to high which causes the direction of current in the the coils of the R/W head to reverse. The result is magnetic transitions on the disk which may represent data. One simple way to represent data by transitions is to serialize data bytes and then toggle the write data line every time that the data bit is equal to a one. The figure below shows an example of how this would look.

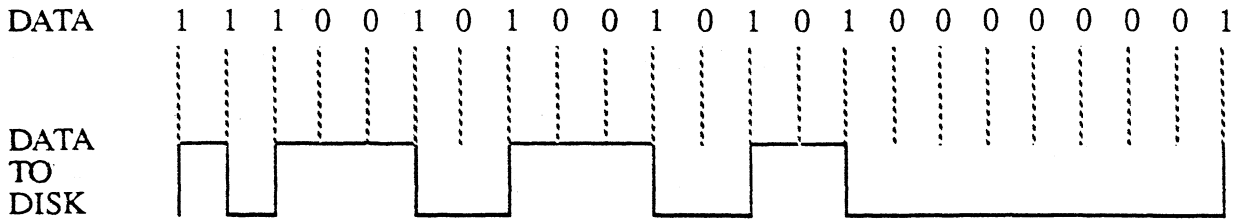


FIGURE 1.

This method of representing data has two basic problems. The first problem is that speed error can make it difficult to determine the number of zeroes in a long sequence of zeroes. The second problem is that a long sequence of zeroes will cause a long time between transitions and signal droop can cause erroneous transitions when read by a NRZ data detector.

These problems indicate that some sort of translation must take place to transform the data into a transitional pattern containing transitions which are not too far apart and provides some method to determine the time base to be used in decoding the data. Over the past 20 years many NRZ codes have been developed. While none of these codes are ideal, the clear winner with respect to common usage today is the Manchester or MFM code. The MFM code follows two basic rules. First, a transition occurs any time that a one is encountered in the data pattern, and second, a transition occurs between any two adjacent zeroes. Figure 2 below shows the MFM pattern for the same data that was used in figure 1.

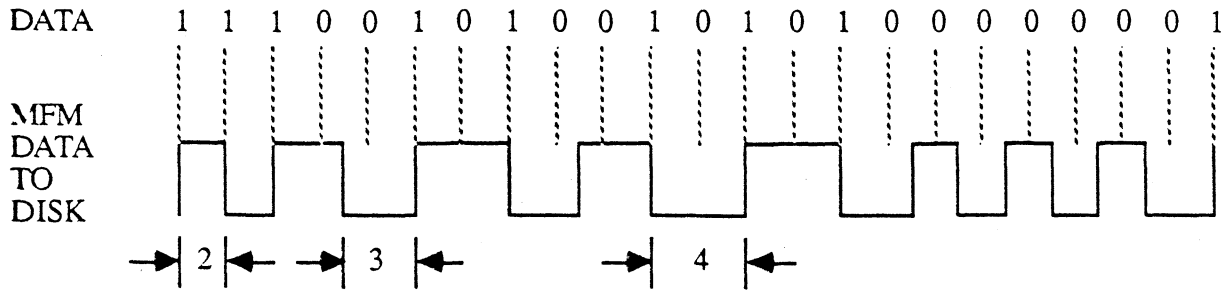


FIGURE 2.

The MFM code produces a series of 2, 3, and 4 unit distances (cells) between transitions which, based on the these distances, when read back can be resolved into the actual data represented.

2. MFM Sector Format

The concept of writing 2, 3, and 4 unit cells provides the mechanism by which the data is translated and written on the disk. But there must be some method for organizing the data so that a specific group of data can be easily located. This is done by writing the data in a Sector Format. A sector consists of 1) information which allows a controller to find the start of the sector, 2) details about which sector is being read, 3) which side is being read, 4) which track is being read (a Track is a group of Sectors), 5) the length of the sector, and 6) CRC error detection information. Figure 3 shows the organization of an MFM sector.

The beginning of a Track or Sector consists of several bytes of 4E's . These bytes serve as a buffer zone between regions of meaningful information. The next bytes in the pattern that are written are the twelve bytes of zeroes (2 unit cells), known as the "bytes of zeroes". These bytes are used to locate the beginning of either a Track, a Sector ID or a Sector Data Field. Following this is the three Mark bytes. The Mark byte is a special byte containing a pattern which violates the basic rules of MFM (i.e has a missing transition). This illegal pattern can be recognized, and provides two very important functions: first, since it is always in the byte that follows the bytes of zeroes, it serves as verification that the zeroes are indeed the beginning of a Track, Sector ID or Sector Data Field, and not data (1's or 0's) in a Data Field and second, the Mark byte provides a reference point from which the MFM rules may be applied to decode the data. (Without synchronizing on a known pattern it is impossible to tell the difference between a string of 1's and a string of 0's.) Figure 4 shows that this illegal pattern is generated by skipping a transition between two zeroes. This produces a 4 unit cell which is normally only valid by a 1 0 1 data pattern. Thus the Mark byte can be detected by finding a 4 unit cell which begins with a zero. After the Mark bytes the next byte encountered in the format pattern is the information byte. This byte is used to determine whether the region being read is the Track information, the Sector ID, or the Sector data field. The next four bytes in the Sector ID contain the track number, side number, sector number and sector length.

	NUMBER OF BYTES	BYTE WRITTEN
	* 80	4E
TRACK ID	* 12	00
	* 3	C2 (Mark Byte)
	* 1	FC (Index Mark)
	* 50	4E
SECTOR ID	12	00
	3	A1 (Mark Byte)
	1	FE (Id Address Mark)
	1	TRACK NUMBER
	1	SIDE NUMBER
	1	SECTOR NUMBER
	1	SECTOR LENGTH
	2	CRC INFORMATION
	22	4E
DATA FIELD	12	00
	3	A1 (Mark Byte)
	1	FB (Data Address Mark)
	256	DATA
	2	CRC INFORMATION
	54	4E
	** ???	4E

* These bytes are only written at the begining of a track.

** These bytes are only written at the end of a track.

Note: The sector ID and Data Field are repeated for each sector on a track.

FIGURE 3.

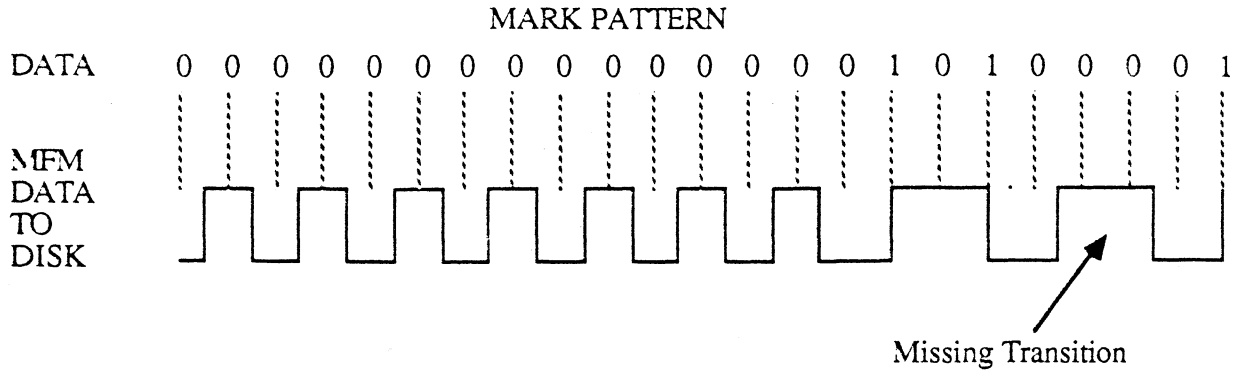


FIGURE 4.

The next bytes to discuss are the CRC bytes. The CRC bytes are used to detect data errors. These bytes are generated by shifting all the bits, from the Mark byte to the byte just before the CRC bytes, through a special shift register which produces a two byte CRC pattern. These bytes are then written on the disk. When the bits are read back from the disk, they all shift through this CRC shift register and once the two CRC bytes have shifted through, the bits in the shift register should all be zero. If the shift register is not cleared to all zeroes then this would indicate that an error occurred in reading back the data. The CRC register is always cleared to ones just before reading or writing the Mark byte. The CRC polynomial that is used here is shown in
figure 5.

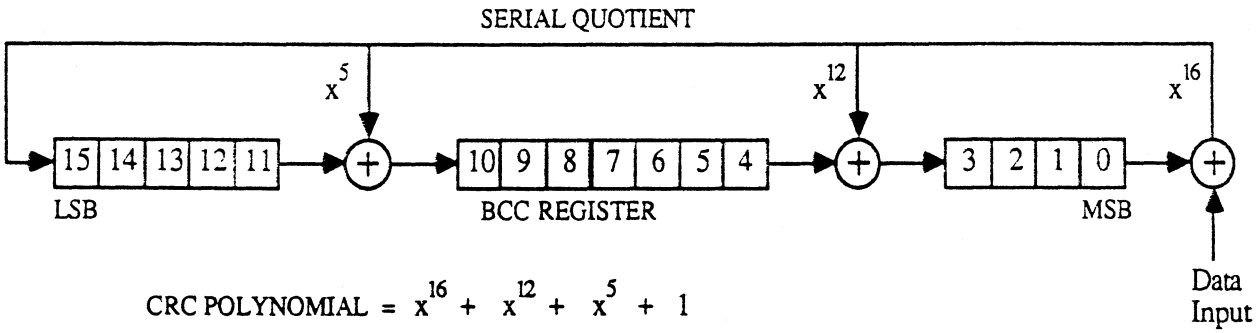


FIGURE 5.

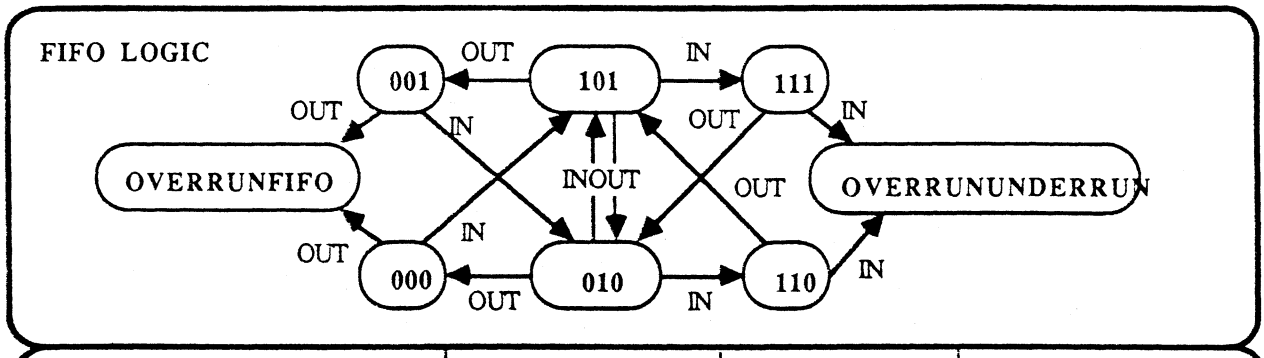
3. MFM WRITE

With the basic concepts of the MFM pattern and MFM Sectors now understood the rest of the discussion will focus on the function of the ISM and how it handles the problems of reading, writing, and interfacing with a processor. The discussion will begin by taking a detailed look at how the Write logic functions.

Figure 7 shows a block diagram of the major portions of the Write side of the ISM. This will be used as an outline for the discussion. Each block in the diagram will be discussed in enough detail to provide a full understanding of how data written by a processor is translated into the 2, 3, and 4 unit cells that must be written on the disk.

3.1 FIFO

The Write process begins when a processor writes a byte into the FIFO and sets a signal in the Mode Register called Action to a one. How the processor interfaces and what other signals must be set up will be discussed later, but this basic function will provide a start in understanding the functions of the Write logic. The byte which is written in the FIFO will now be tracked through the write chain to see how the data is written on the disk. The FIFO used in this application is a two byte FIFO consisting of two ten bit registers and some control logic. The FIFO contains eight bits of data, a bit which indicates whether the data is a Mark byte, and a bit which tells the system to write the CRC bytes. The control logic contains the state machine which is used to control the FIFO. The state diagram for this state machine is shown in figure 6.



AFULL	BFULL	AELDEST	READ IN TO FIFO	READ OUT TO COMP	WRITE IN TO SR	WRITE OUT TO FIFO	WRITE IN TO SR	WRITE OUT TO FIFO
0	0	1	LDB		RDB		RDA	
0	0	0	LDA		RDA		RDB	
1	0	1	LDB	RDA	RDB	LDA	RDA	LDB
0	1	0	LDA	RDB	RDA	LDB	RDB	LDA
1	1	1		RDA		LDA		LDB
1	1	0		RDB		LDB		LDA

FIGURE 6.

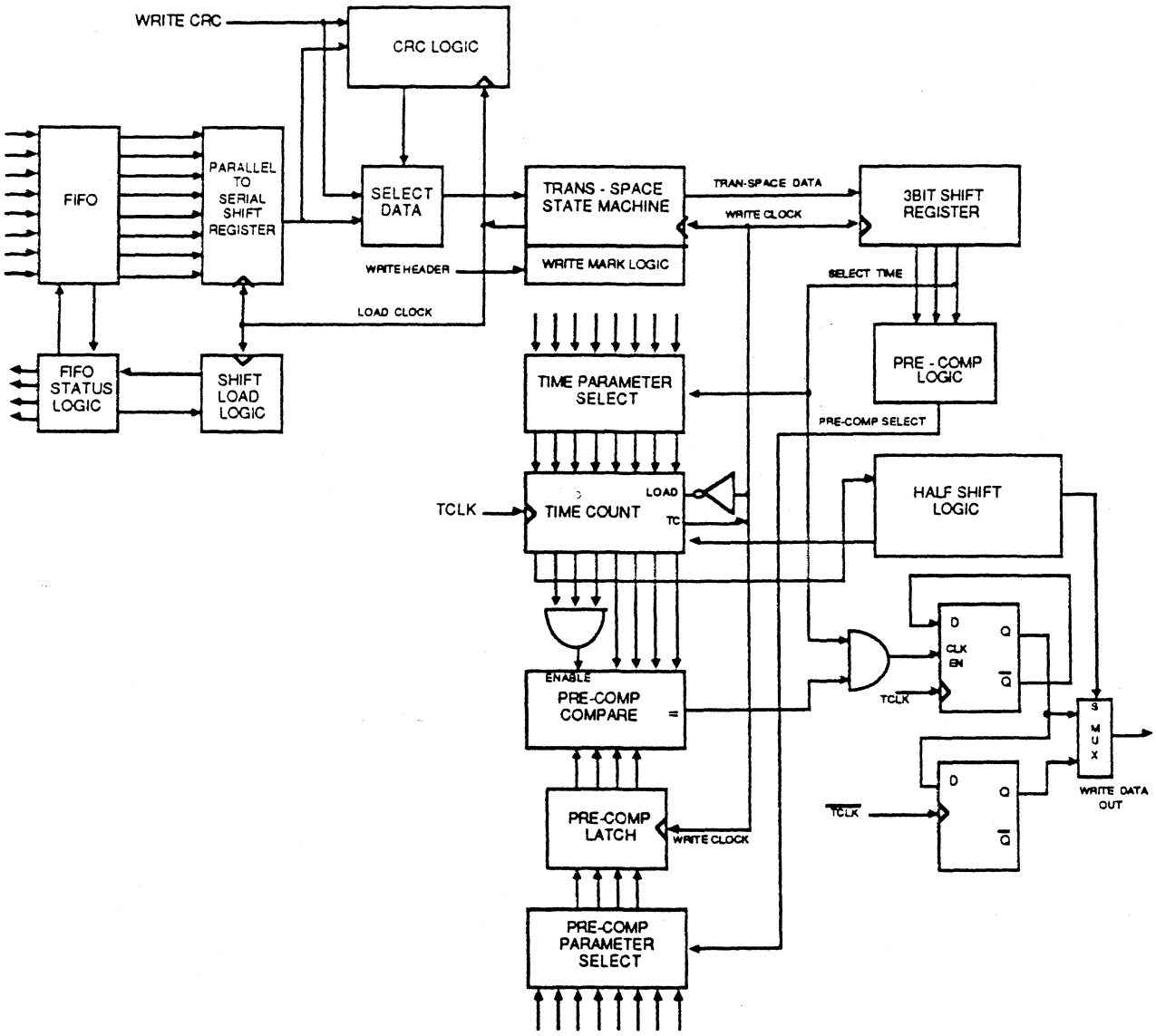

WRITE CRC
CRC LOGIC
FIFO
PARALLEL TO SERIAL SHIFT REGISTER
SELECT DATA
TRANS - SPACE STATE MACHINE
TRAN-SPACE DATA
WRITE CLOCK
3BIT SHIFT REGISTER
WRITE HEADER
WRITE MARK LOGIC
SELECT TIME
LOAD CLOCK
FIFO STATUS LOGIC
SHIFT LOAD LOGIC
TIME PARAMETER SELECT
PRE - COMP LOGIC
PRE-COMP SELECT
TCLK
TIME COUNT
LOAD
TC
HALF SHIFT LOGIC
ENABLE
PRE-COMP COMPARE
=
D
CLK EN
Q
QI
TCLK
S
M
U
X
WRITE DATA OUT
PRE-COMP LATCH
WRITE CLOCK
PRE-COMP PARAMETER SELECT

FIGURE 7.

The state machine diagram shows that the FIFO is used for both writing data and reading data. The FIFO consists of a signal called IN which represents a byte being transferred to the Shift Register from the FIFO when writing and a byte being transferred from the Shift Register to the FIFO when reading, and a signal called OUT which represents a byte being shifted from the processor to the FIFO when writing and a byte being shifted from the FIFO to the processor when reading. It can be seen from the state diagram that the same state machine will work for both writing or reading because the state machine is symmetrical and as long as an AFULL = 1 and a BFULL = 1 is thought of as needing attention by the processor (requiring an OUT). The only difference in the state machine between reading and writing is that the state bits must be preset to ones prior to engaging in a write operation, and must be preset to zeroes prior to a read operation. This is necessary because when writing the processor must fill the FIFO prior to setting the Action bit (the bit which starts the shifting of the bits inside the chip), and when reading the processor is waiting for the FIFO to be filled by the bytes being read from the disk. The IN signal is basically just the terminal count of a counter which determines when a byte has either been shifted out of the shift register when writing or shifted into the shift register when reading. There are two exceptions to how this works. The first has to do with a special mode in the chip called NGCR mode which will be discussed later. The second exception is that an IN occurs at the instant that Action is set when writing, which is used to transfer the first byte of data into the Shift Register immediately thus pre-setting the Shift Register with meaningful information rather than having an extra byte of garbage written on the disk. An OUT is generated any time that a data byte or Mark byte is written or read by the processor, or when the processor tells the chip that a CRC byte is to be written.

The Mark bit in the FIFO is used to pass on the information that the current byte is a Mark byte and that the transition should be skipped between the two zeroes in the Mark pattern. The bit will be set when a byte of data is written to the Write Header location. The CRC bit is set when a Write to the Write CRC location occurs. The data that is in the FIFO when this bit is set is meaningless because when this bit is set, data will stop shifting out of the shift register and will begin shifting out of the CRC register. This is how the CRC information is written on the disk.

3.2 SHIFT REGISTER

The next block on the write diagram to discuss is the Shift Register. The shift register is also used for both reading and writing. When writing, the function of the shift register is to load the eight data bits from the FIFO and shift them out in a serial fashion. When reading, the Shift Register is used to shift in a serial bit stream and then move these bits into the FIFO when a full byte of data has shifted in. The bits are shifted out starting with the most significant bit first when writing, and shifted in with the most significant bit first when reading.

3.3 CRC REGISTER

The serial bits that come out of the Shift Register are shifted into both the CRC Register and the Trans-Space State Machine. As discussed earlier, the CRC is cleared to ones just prior to writing the Mark byte. All subsequent data bits to be written shift through the CRC, as shown in figure 5, until it comes time to write the CRC bytes. At this time the bits in the CRC are shifted into the Trans-Space machine to be converted into the MFM data and written on the disk.

3.4 TRANS-SPACE

The Trans-Space State (TSS) Machine is used to translate the data stream into a form in which a one represents a transition and a zero represents a space. The Trans-Space form is used for converting the data into the MFM pattern. The front end of the TSS machine is a four bit shift register which provides a mechanism for knowing what the last two bits were, the current bit is, and the next bit will be. Most of the time the only information that is needed is what the current bit is and what the next bit will be. The exception is when writing the Mark byte. In this instance more information is needed because it must be determined were to leave out the transition. Referring back to figure 4, it is shown that the only time when a transition needs to be skipped is when there is a 1 0 0 0 pattern, thus, all four bits of information are needed. The table in figure 8 shows the transformation of the data into the transition space format.

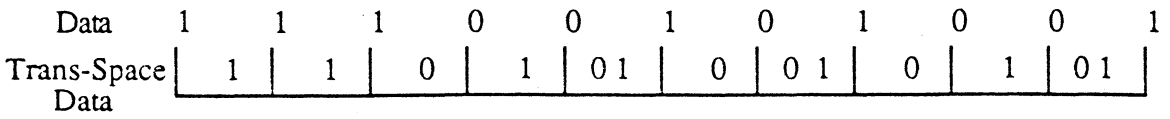
TRANS-SPACE TABLE

Current Bit	Next Bit	Transpace Data	Mark
0	0	1	0 0
0	1	0 1	0 1
1	0	0	0
1	1	1	1

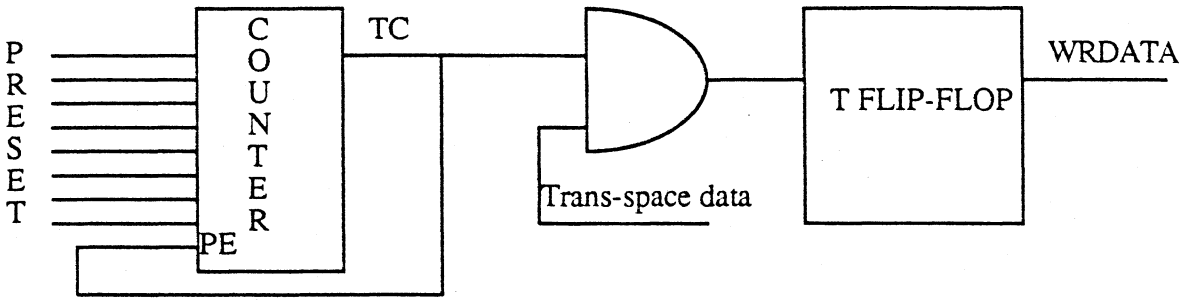
FIGURE 8.

A better understanding of how the translation from data to Trans-Space takes place and how this corresponds to the MFM pattern can be seen in an example.

Example 1:



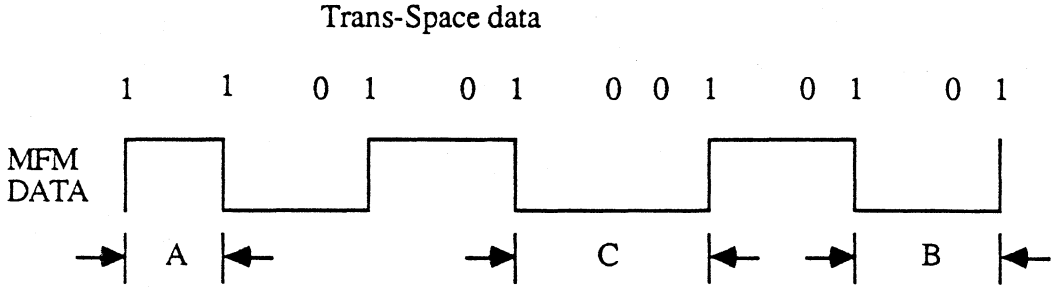
Now suppose that a 1is defined to cause a T flip-flop to toggle and a counter to preset to a value Time 1. And a 0 is defined to cause no toggle in the T flip-flop and the counter to preset to a value Time 0. Furthermore, suppose Time 1 caused the couter to count up to a value which is 2 units long and Time 0 causes the counter to count up to a value which is 1 unit long. Also assume that the T flip-flop can only toggle when the counter reaches it's terminal count.



PRESET = TIME 1 IF TRANS-SPACE DATA = 1

PRESET = TIME 0 IF TRANS-SPACE DATA = 0

The Trans-Space data generated above will thus create a pattern which looks like the following:



A = TIME 1 = 2 UNITS

B = TIME 1 + TIME 0 = 3 UNITS

C = TIME 1 + TIME 0 + TIME 0 = 4 UNITS

The example clearly shows that the translation into the Trans-Space pattern provides a simple way to convert the data into the MFM pattern by using a T flip-flop and a counter. An important point is that depending on the clock frequency at the counter, and depending on the values of the preset (as long as they are kept in a 2:1 proportion), the actual times of the cells (2's, 3's, and 4's) can be varied. For example, a standard IBM drive operates at 300 rpm with cell times of 4, 6, and 8 µs, whereas a Mac drive operates at five different speeds ranging from 390 to 590 rpm with correspondingly different cell times. If it is desired to write a disk on a Mac drive which can be read on an IBM drive, it simply takes changing the preset values (TIME 1 & TIME 0) in proportion to the speed zone where the disk is being written. This is one of the very

important features of the ISM. These presets (which from now on will be referred to as parameters) and many other presets used for other counters, which will be discussed later, are stored in RAM and can be written and read from an external processor. Thus, these parameters can be changed as desired in order to provide the flexibility to work with drives which function at different speeds.

3.5 PRE-COMPENSATION

If the properties of magnetic media acted in an ideal manner, than the discussion of the write logic would be finished at this point and the simplistic example shown above would work fine for writing the data on a disk. However, the properties of the media are not ideal so there must be some revisions made to the above example to help compensate for these non-ideal traits. The problem is that a 2 unit cell on a disk is crowded together more than a 3 or a 4 unit cell in a relative sense. The effect of this crowding is that the 2 unit cells will tend to push out their transition into the region of a 3 or 4 unit cell, when a 2 is next to a 3 or 4. This will cause a 2 unit cell to be longer than it is supposed to be and a 3 or 4 unit cell to be shorter when reading back the data. This problem is called peak shift.

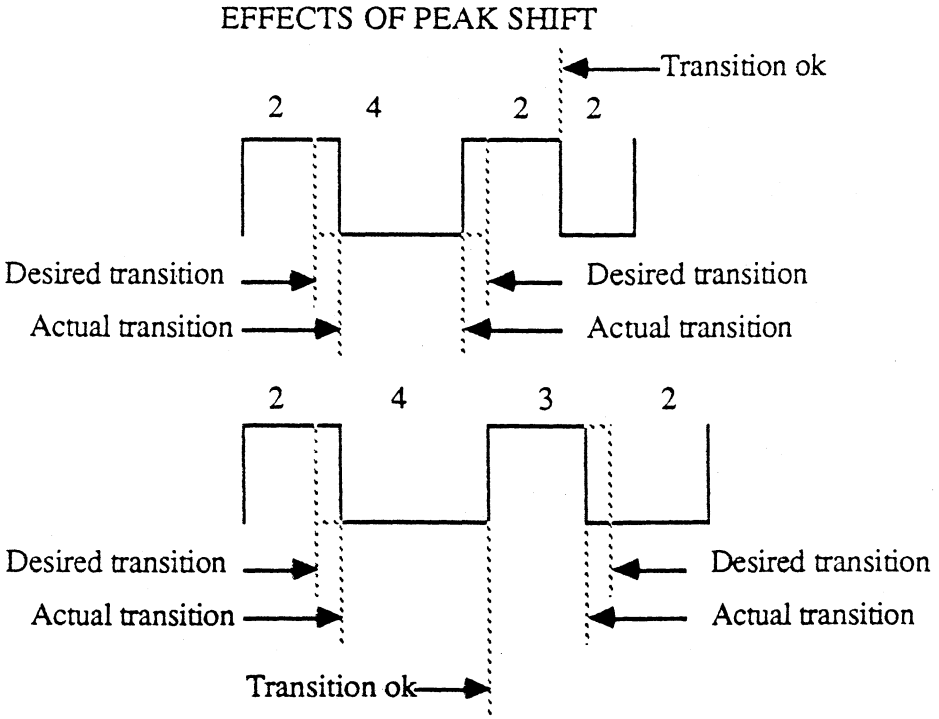


FIGURE 9.

The actual amounts that the transitions are affected due to peak shifting can be determined experimentally. Having this information makes it possible to correct for this problem by simply causing a transition to occur earlier or later when writing. In other words, make the 4's and 3's longer and the 2's shorter when they are next to each other. This type of correction is called Pre-Compensation (Pre-Comp). The earlier example showed that when the counter reached it's terminal count a decision would be made whether to toggle the T flip-flop or not. In order to accomplish the effect of Pre-Comp this idea needs to be modified slightly. Rather than using the terminal count, what is done is to use the four lower counter bits and compare them with some value. This value is another one of the parameters that is written in to RAM by a processor. The

value of the comparison point has three different possibilities, Early, Normal, and Late. Normal will provide a comparison point which will serve as the constant point of reference. Early will give a comparison point which occurs slightly earlier than the Normal point and can be used to cause a transition to occur a little earlier. Late will cause the transition to occur a little later than the Normal position. The value of Normal is arbitrary since it serves only as a comparison point and all other comparison points use the Normal value as it's reference. The value of Early and Late are determined based on how many clocks it takes to correct for peak shift which can be determined experimentally by studying the actual information read from the disk on the drive. The block diagram of the write side shows the comparator and the counter and how when the higher order bits have reached all ones and the lower order bits have reached the comparison value, the T flip-flop is enabled. The following example will make it easier to understand these concepts:

Example 2:

Suppose the desired cell times are 4, 6, and 8 µs and the clock frequency is 7.16 Mhz.
this implies

Number of clocks in 4 µs = 7.16 * 4 = 28.64 clocks ~ 29 clocks

Therefore Time 1 must represent a count of 29 clocks
and Time 0 must represent a count of 14 clocks.

Early, Normal, and Late are compared to the 4 least significant bits of the counter during the last 16 counts before Terminal Count, which implies that their values must lie somewhere between 0 and 15. Now suppose Normal = 10 counts. This would imply that the comparison
will occur (16 - Normal = 6 counts) before the Terminal Count. Since the

counter preset does not occur until the Terminal Count, this would imply

Suppose the pattern is a string of 1's. This would imply no precomping

and cell times of 29 clocks. Now since the preset for the counter always occurs when the counter reaches it's terminal count rather than when the comparison is true, it can be seen that the 29 clocks are made up by a combination of the distance from the previous comparison to the preset

thus Cell time = distance from comparison to Terminal Count
+ Time 1 - Distance from the next comparison
point to Terminal Count = (16 - 10) + 29 - (16 - 10) =

Now suppose that Early = 8 clocks and Late = 12 clocks.

And the pattern is a 1 1 1 0 0 1 1.

The second transition will occur at the normal boundary because it has a 1 preceeding it and following it. But the third transition needs to occur earlier because it is a 2 unit cell followed by a 4 unit cell. Thus the comparison

Which implies:

Cell time = (16 - 10) + 29 - (16 - 8) = 27 clocks.

This will cause the 2 unit cell to be shorter and the 4 unit cell to be

Now since the fourth transition is part of a 4 unit cell and the next cell is a 2

unit cell, the transition must be delayed. This is done by comparing to the

Cell time = (16 - 8) + 29 + 14 + 14 - (16 - 12) = 61 clocks.

Normaly the cell time for a 4 unit cell should be 57 clocks, but due to the fact that cell is sandwiched between two 2 unit cells it must be lengthened by two counts on each edge and the 2 unit cells must be shortened by 2 counts

The net effect is that a transition can be delayed or occur early in order to

The example explains how a transition can be delayed or occur early, but it does not explain how to determine which should occur to a particular transition. This is accomplished by shifting the Trans-Space data through a three bit shift register which provides the ability to know what is happening and what is going to happen for the next two data times. This information can be used to determine which comparator value should be loaded from RAM.

3BIT REG.			PRE-COMP SELECT
0	1	0	NORMAL
0	1	1	EARLY
1	1	0	LATE
1	1	1	NORMAL
X	0	X	NORMAL
t+2	t+1	t	

FIGURE 10.

The parameter data bus coming from the RAM is 16 bits wide. Thus, the Pre-Comp parameters and the Time 1 or Time 0 parameters must be loaded at the same time. This is done in the following fashion.

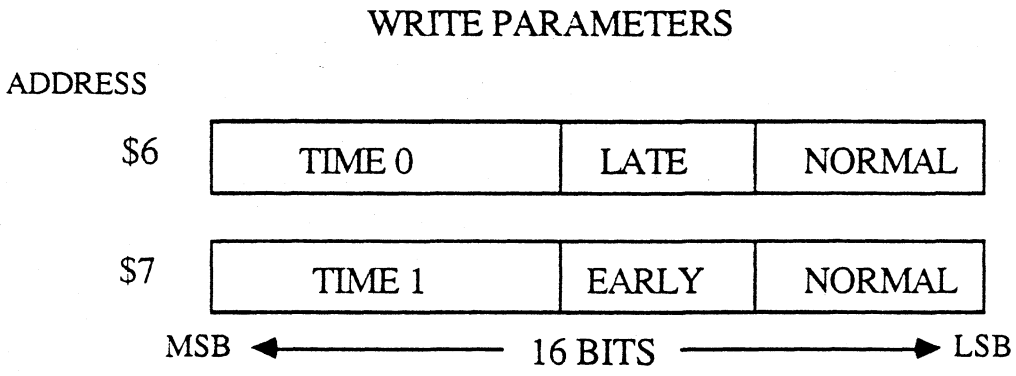


FIGURE 11.

This manner of grouping the parameters works as follows. If the high bit of the 3 bit shift register is a 1 then the Time 1 must be loaded because a transition is about to occur, and the Time 1 count always follows a transition, as discussed in an earlier example. The only possible transition which can follow this without having to load the Time 0 parameter is a 2 unit cell and the Pre-Comp effect of this cell is to either shorten it or not change it at all depending on what is to follow. In this case the only way that the transition can be 2 units long is if the second bit in the shift register is a 1. So if the third bit in the shift register is also a 1 then another 2 unit cell is to follow and the Normal parameter is selected. Otherwise the third bit is a 0 which implies that the next cell is going to be longer than a 2 and the current transition point must be Pre-Comped by selecting the Early parameter. If the high bit in the shift register is a 0 then the Time 0 value must be selected and the only possible Pre-Comping that can take place is to either lengthen the cell or leave it alone. This is true because the 0 has to be in the middle cell of a cell which is

larger than 2 units wide and the only choice for such a cell is to delay the transition or leave it alone depending on whether the next transition is a 2 or longer.

3.6 HALF WRITE

The only portion of the write logic left to discuss is the Half Shift Logic. In example 2 it was shown that the number of clocks in a 4 µs cell at 7.16 Mhz is 28.64. The number was then rounded to the nearest unit because there was no mechanism was described for obtaining any more than an integer number of clocks. This causes a round off error which forces the cell times to vary from the desired values. Depending on the clock frequency used this can be quite a significant error. In order to reduce this round off effect, circuitry is added which works on both edges of the clock which creates the effect of having half clock resolution. This means that the values can be rounded to the nearest half instead of the nearest unit. Using the half clock can be very difficult because of the high effective clock speeds that are generated. That is why the half clock is only used at the very end of the Write path using very short delay paths and very fast parts. The low bit of the parameter data is used to determine whether or not the transition should be shifted by a half clock value. The Schematical representation of how this half effect is generated is shown in figure 12.

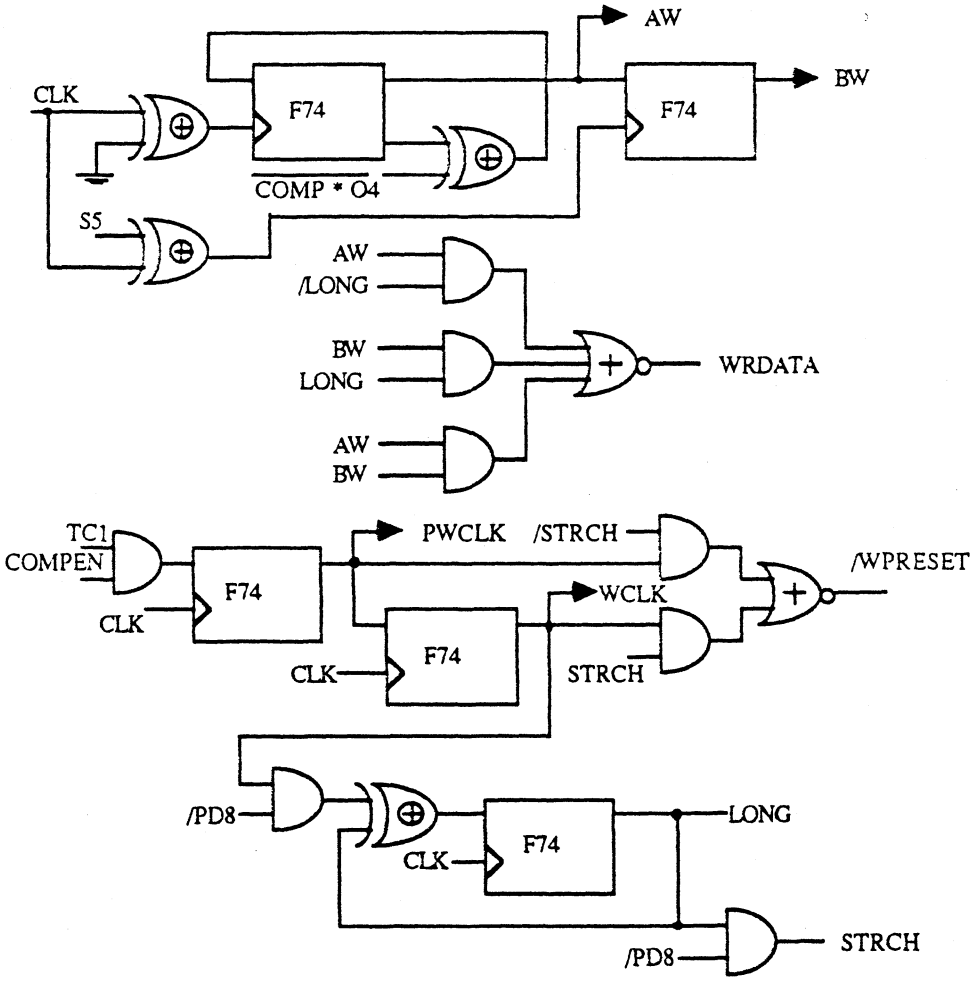


FIGURE 12.

The signal /Comp*O4 represents the output of the comparator and the value of the high bit in the three bit post Trans-Space shift register. In other words this represents the point at which the transition should occur and if no half bit mechanism were used the signal AW, which just toggles every time that /comp*o4 occurs, would represent the Wrdata to the disk. But it is desired to gain more resolution by using the half clock, so a signal called BW is generated which is a 1/2 clocked delayed AW signal. The signal TC1 and COMPEN is just the terminal count of the Time 1/Time 0 counter and the signal /PD8, which is the least significant parameter bit, is the bit which determines whether or not to cause the transition to occur on the half clock. The timing diagram in figure 13 shows how these signals can be used to create the half clock effect.

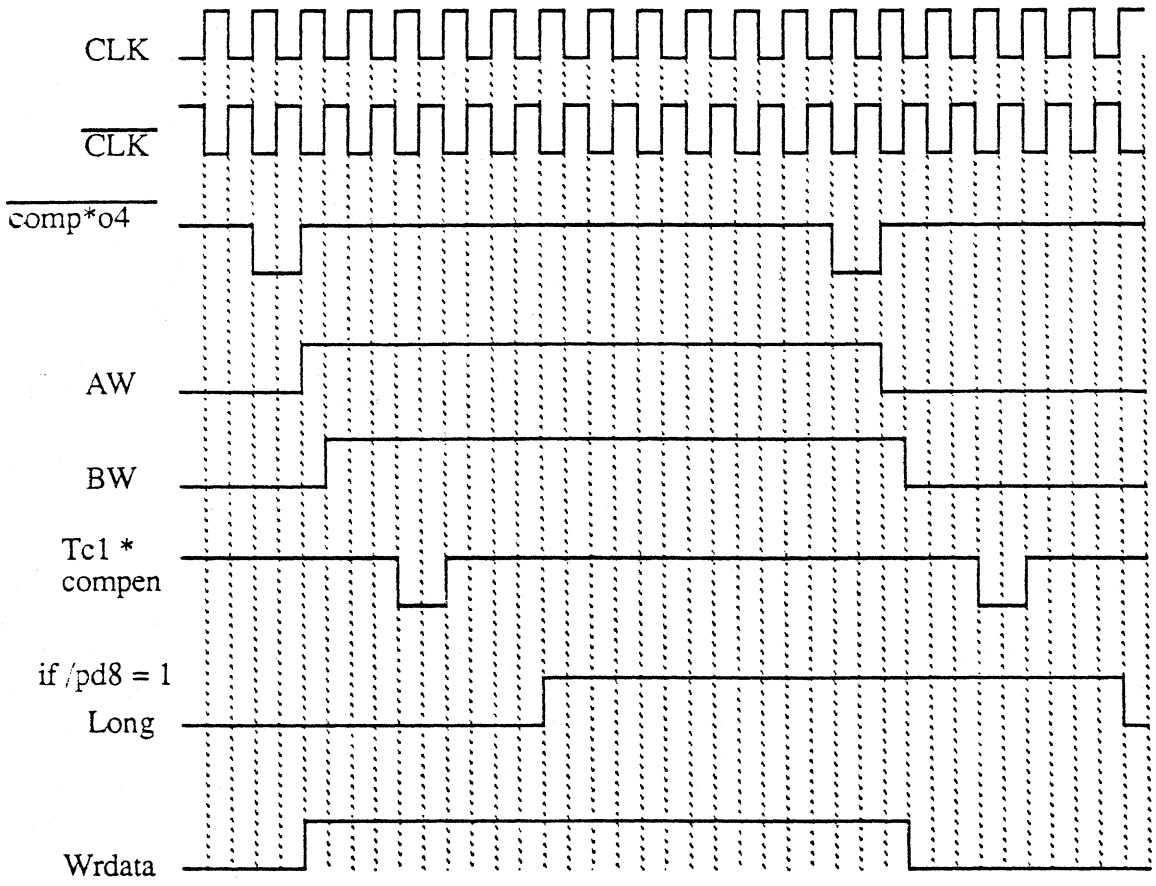


FIGURE 13.

If the Long signal was always equal to zero then Wrdata would always be equal to Aw and no half shift would take place. This occurs when the half bit (/PD8) parameter is equal to zero. When the long signal is present this causes the Wrdata pulse to lengthen, by the amount which BW offsets from AW, on alternate pulses. This produces the desired half clock delay in Wrdata. The WCLK signal, in figure 12, is a two clock delayed terminal count which shifts the bits from the Trans-Space machine into the Pre-Comp shift register and forms the signal which shifts the bits from the main Shift Register into the CRC and Trans-Space machine. The only other signal, in the diagram above, not yet discussed is the /Wrpreset signal. This is the signal which causes the Time 1/Time 0 counter to preset. This preset normally occurs one count after the counter reaches it's terminal count, but when the half shift takes place (Long is true) the preset is delayed for one more clock in order to compensate for the transition taking place late.

4. MFM READ

Data is read from a disk by means of a signal called Rddata. This signal consists of pulses which are spaced at 2, 3, and 4 units apart, which of course is the Data in it's MFM translated form. If all conditions were ideal the simplest way to convert the MFM format data into it's actual data would be to determine whether a cell is a 2, a 3, or a 4 and go through a reverse translation process, analogous to what was used on the write side, and transfer the data through the Shift Register and the Fifo to the controlling processor. But there are two problems with this simple theory. The first problem is a familiar one that was discussed on the Write side called Peak Shift. In the Write side it was shown how the effects of Peak Shift can be reduced by using Pre-Comp, but suppose the disk that is being read was not written on a controller which uses Pre-Comp, or suppose the Pre-Comp used was not enough. This could cause errors reading back the data because the effects of Peak Shift could make it very difficult to discern between a 2 and a 3 unit cell or a 3 and a 4 unit cell. This problem is solved using a very complicated form of correction called Post Compensation (Post-Comp). The other problem that can occur is that the speed of the disk drive or the frequency of the clock can be off, or there can be some other form of systematic error in the data. This can also make it very difficult to read back the data reliably. This type of problem can be corrected by use of an Error Correction machine. The Discussion of the Read logic will take a detailed look at how the Post-Comp and Error Correction machines work along with a description of how the beginning of a track or sector is located, how the Mark byte is detected, and what starts the process of transferring the data into the FIFO.

4.1 HALF READ

The first topic to discuss is the Rddata signal itself. As mentioned earlier this signal occurs at distances of 2, 3, and 4 units apart, but the actual width of each pulse may vary and may not be synchronous with the internal clock. In order to work with this signal in a synchronous environment the signal must be synchronized with the clock and must be transformed to be one clock wide. The simplest way to accomplish this is shown below.

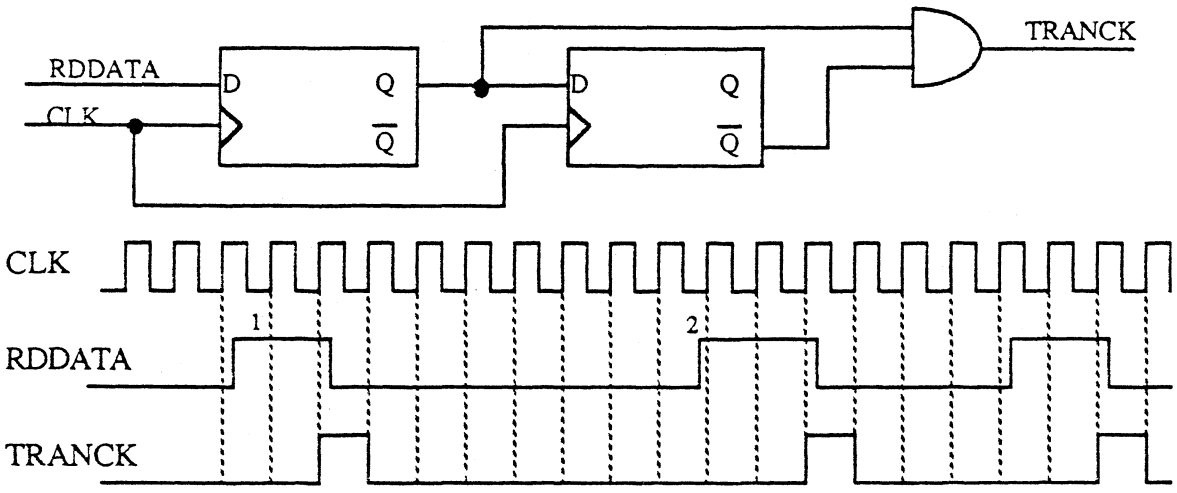


FIGURE 14.

This method of transforming Rddata into TranCk works fine most of the time, but due to the resolution limitations of the clock, it is sometimes necessary to be more precise. In order to understand why this is true, it is necessary to shift gears a little bit and discuss the problem of

distinguishing between a 2, 3, and 4 unit cell. The cells can be distinguished by using a counter which sets up boundaries which define comparison regions to the TranCk signal. Figure 15 shows a picture how this looks.

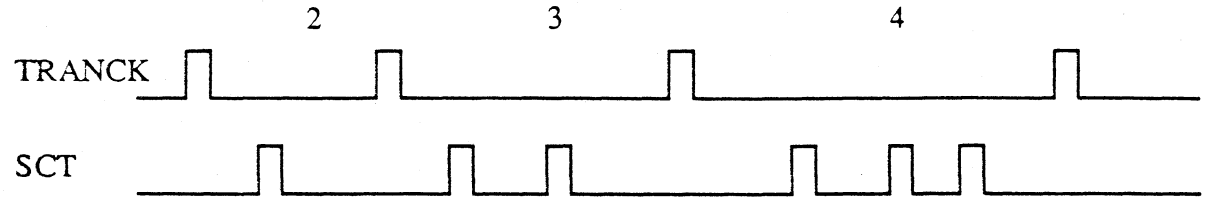


FIGURE 15.

SCT is generated by a counter which presets every time that a TranCk or an SCT occurs. The value that it presets to is another one of the programmable parameters that come from RAM. These values are calculated based on the clock frequency and the actual cell times in µs which allows the chip to be run at different clock frequencies and with variable rates of data coming off the disk. By counting how many SCT's occur between TranCk's it can be determined whether the pulse is 2, 3, or 4 units wide. If the cell times of the data coming off the drive are very accurate then there is no problem resolving the data because the parameters can be set to fit right in the middle of each region and there should be plenty of margin between the SCT pulses and TranCk pulses, but in reality due to drive and noise error there can be some error in the values of the cell times. This can cause the SCT pulses and the TranCk pulses to fall very close to each other making it difficult to tell the difference between two different cell times, as shown in figure 16.

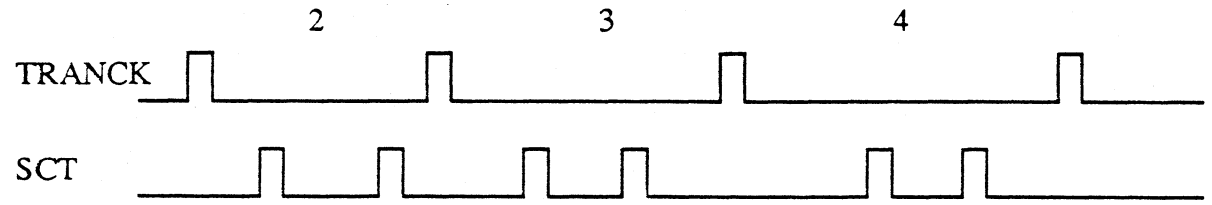
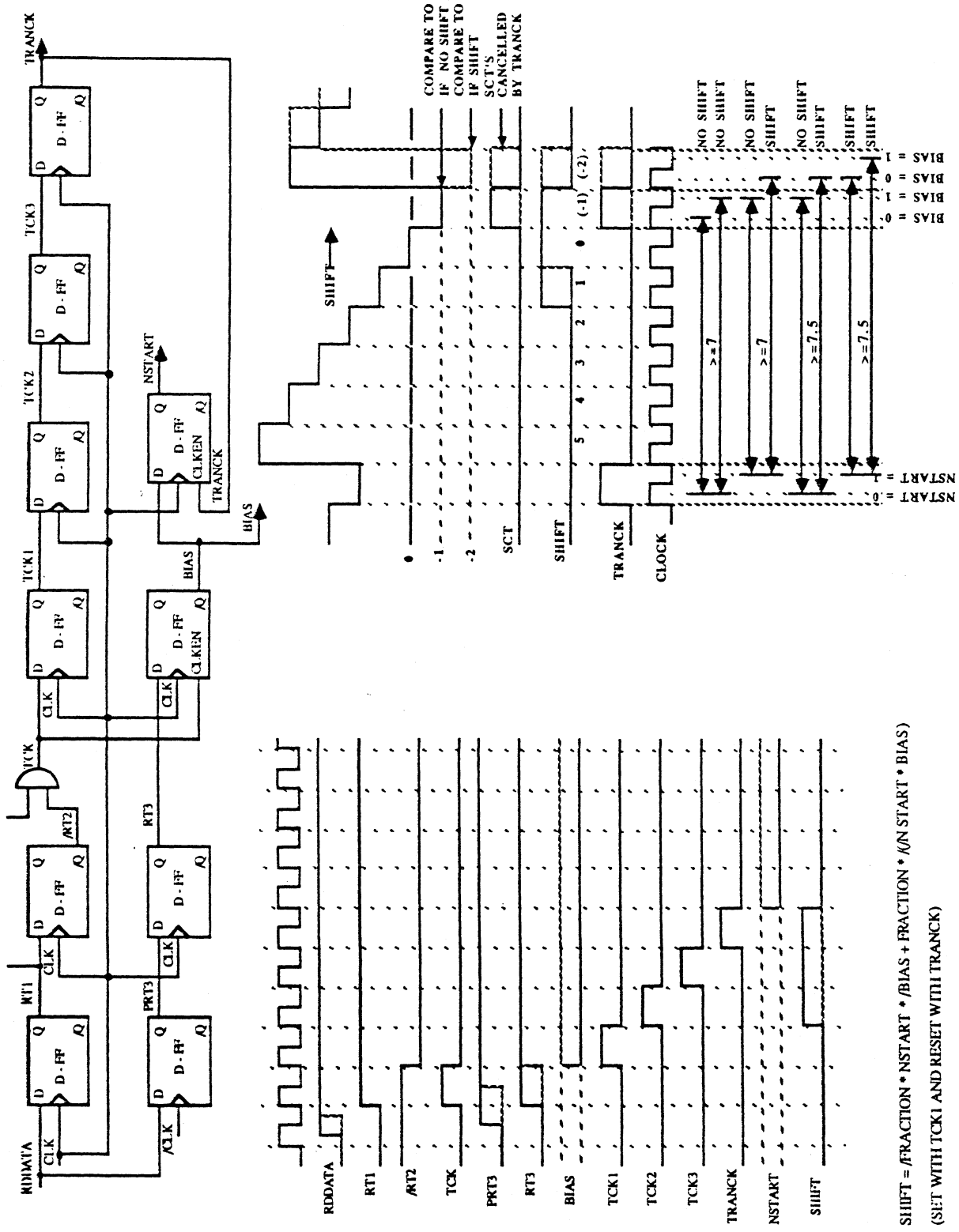


FIGURE 16.

This can transform what is intended to be a 2 3 4 pattern into a 3 3 3 pattern. If the difference between an undesired or an extra SCT occurring is one or two clocks then this error could have been introduced because of the manner in which Rddata was formed into TranCk. To understand this point it is necessary to refer back to figure 14. In figure 14 it can be seen that the first Rddata pulse (1) occurred just after the rising edge of the clock while the second pulse (2) occurred just prior to the rising edge of the clock. Since data can only be sampled on the rising edge of the clock, it can be seen that almost one full clock of error has been introduced in the length of the cell. This problem can be reduced by determining which half of the clock cycle the Rddata pulse occurred in and shifting the SCT pulse by one count to compensate. Shifting the SCT pulse will effectively change the distance between TranCk pulses. The overall effect is that the distance between Rddata pulses can be resolved to within one half clock of the actual distance instead of one clock. The effective half clock shift of SCT can take place in two manners. First to compensate for the problem just mentioned and second to allow for better resolution in calculating the parameters for the SCT counter. Figure 17 shows a schematical representation of how the shift is generated.


TRANCK
TCK3
TCK2
TCK1
NSTART
BIAS
RDDATA
RT1
/RT2
TCK
RT3
PRT3
CLK
CLKEN
D-FF
SHIFT
COMPARE TO IF NO SHIFT
COMPARE TO IF SHIFT
SCT'S CANCELLED BY TRANCK
SCT
TRANCK
CLOCK
NO SHIFT
SHIFT
>=7
>=7.5
BIAS = 1
BIAS = 0
NSTART = 1
NSTART = 0
SHIFT = /FRACTION * NSTART * /BIAS + FRACTION * /(N START * BIAS)
(SET WITH TCK1 AND RESET WITH TRANCK)

The figure shows that the TranCk signal is formed in the same manner as earlier shown with the exception that it is delayed for four clocks. This pipelining is necessary to be able to know when the TranCk is going to occur four clocks before it occurs. The Rddata signal is synchronized to the nearest half clock and then delayed by one clock to generate the signal RT3. When the TCK signal becomes valid RT3 is sampled. If Rddata occurred in the first half of the clock cycle RT3 would be a one, and if Rddata occurred in the second half of the clock cycle RT3 would be zero. This information is then latched in as signal called Bias. The signal Bias will be set to a zero if Rddata occurred in the first half of the clock cycle, or set to a one if it occurred in the second half of the clock cycle. The signal Nstart is used to latch Bias when TranCk occurs. This is used on the next Rddata to determine what has just occurred since the Bias signal will change on the next TCK. As mentioned earlier, the important point in this whole discussion is whether the SCT should or should not be shifted near a TranCk. This can now be resolved using the information generated. Since it is known when the TranCk is going to occur four clocks prior to it actually occurring, and it is known which half of the clock cycle the Rddata pulse that generated the TranCk occurred in and the same information about the previous Rddata pulse is known, a signal called Shift can be generated which will cause the comparison point in the SCT counter to be altered by one count. Thus producing the result of correcting to the nearest half clock. The equation for shift is given in the lower left corner of the figure. In order to understand how it works it is necessary to study the example in the lower right corner of the figure.

4.2 CORRECTION MACHINE

In the discussion of the MFM sector format it was mentioned that the beginning of a sector or track can be located by finding the 12 bytes of zeroes followed by the Mark byte. This is accomplished using the Correction State Machine (CSM). The CSM looks for a string of minimum cells by looking at the number of SCT pulses that occur between TranCk pulses. If the CSM sees 64 cells which have only one SCT pulse between transitions, then it knows that it has found a region of minimum cells. The machine then looks to see if the first non-minimum cell is part of a Mark byte. If this is the case then the rest of the bits will start shifting into the Shift Register and the FIFO will begin functioning. Otherwise the CSM will go back into the state which looks for a string of minimum cells.

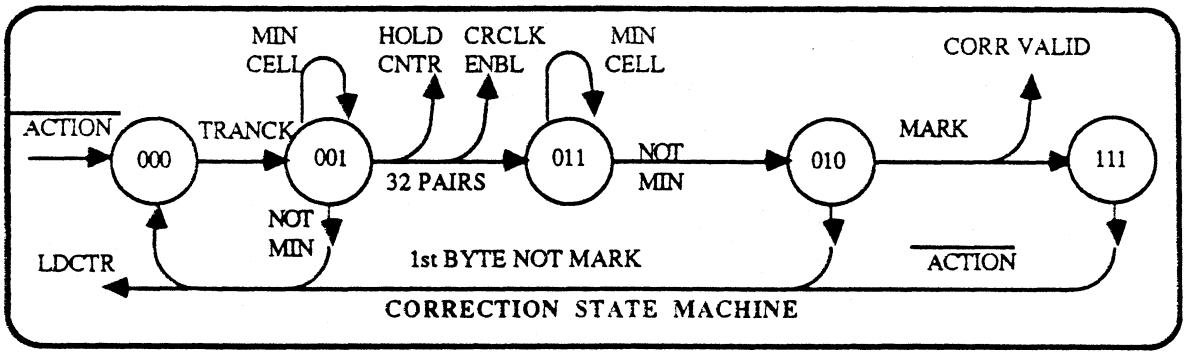


FIGURE 18.

The state diagram above shows how the CSM works. It starts out in the 000 state and stays there until it gets a transition. At this point it goes into the 001 state where it stays until it encounters 32 minimum cells. If 32 pairs of minimum cells are then counted the machine proceeds on to the next state, otherwise it goes back to look for another transition. Once it has gotten the 32 pairs it waits for the first non-minimum transition to occur. If this non-minimum cell is part of the Mark byte then it proceeds on to the next state where it remains until the

processor is finished reading bytes. If it is not part of the Mark byte then the machine goes back to look for zeroes.

4.3 ERROR CORRECTION

In the previous discussion it was first mentioned that the CSM waits for 64 cells to occur before proceeding on to the next state, then later it was stated that it waits for 32 pairs. These terms mean the same thing because a pair is defined to be two cells. The reason for thinking in terms of pairs instead of cells is because of the way the data is stored on the disk. In the discussion on the Introduction to MFM it was mentioned that the data is written on the disk by reversing the direction of the current flow which cause the direction of the flux to reverse which causes a magnetic transition in the R/W head. The transitions on the disk will look like the following when read back.

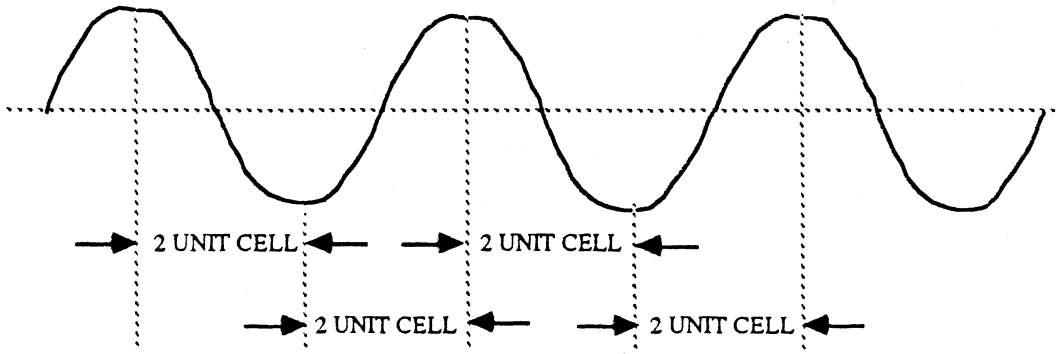


FIGURE 19

The first 2 unit cell is between the positive going transition and the negative going transition, while the second 2 unit cell is between the negative going transition and the positive going . This implies that two 2 unit cells are made up by a positive and a negative transition which is what is being defined as a pair. Distinguishing between the positive transitions and the negative transitions is important because due to the properties of the media there is some error in determining the exact location of the transition and the tendency is that the error for positive transitions is consistent in one direction while the error for negative transitions is consistent in another. This error can be corrected using the Error Correction machinery, but the correction must take into account whether the edge that is being corrected was created by the negative or positive transition. It is not possible from the Rddata signal to determine whether the transition was created by a positive or negative transition, so what is needed is to simply remain consistent by correcting one way on every other transition and correcting the other way on the other transitions. For example suppose there is a series of 2 unit cells in which one direction of the flux on the disk caused the transition when read back to occur early. This would create the following effect.

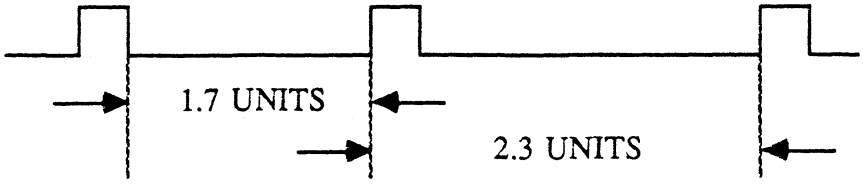


FIGURE 20.

The 2 unit cells have now become alternately smaller and larger. Another problem which can effect the length of the transitions is the speed of the disk drive. If the speed at which the disk is being revolved is too slow this will cause the cells to be longer than they are supposed to be which can make it difficult reading the data because the parameters that preset the SCT counter are determined based on the knowing what the speed is. Both of these problems can be solved by taking a sample of minimum cells, and by knowing ideally how many clocks occur between transitions, it can be determined how far off the cell times are from ideal. Once it is known what this error is, the SCT counter can be modified by causing skips or double counts to help align the SCT boundaries up in a fashion which is consistent with the non-ideal data. This error can be accumulated separately for alternate transitions thus allowing for the differences between the negative and positive flux pulses. This Error Correction is implemented as follows.

While the Correction State Machine is looking for the 32 pairs, there are two counters which are counting the number of clocks that occur between each transition. One counter counts the number of clocks in in the first half of the pair of cells, and the other counts the number of clocks in the second half of the pair of cells. When the CSM resets to the 000 state these counters are cleared to zero. Once the CSM indicates that 32 pairs have been found the counters will be held. The reason for using 32 pairs is to give a large enough sample such that the amount of error is representative of what is actually occurring rather than a random error that can occur over the course of just a few cell times. Since the number of clocks between minimum cells can be calculated based on the cell times and the clock frequency, it can be determined how many total clocks should occur over the course of 32 pairs. Thus, any deviance from this count can be considered error which must be corrected. For example, suppose the minimum cell time is 4 µs and the clock frequency is 16 Mhz. This would yield 4 * 16 = 64 clocks per minimum cell time. Over the course of 32 samples the total number of clocks should be 64 * 32 = 2048 clocks. In other words if all the cells times were perfectly accurate, each counter would count 2048 clocks over the period of 32 samples of minimum cells. Any deviance from this number would represent an error which must be corrected by shifting the SCT pulses in a proportional manner to the amount of error which occurs during one cell time. Suppose one of the counters only reached a count of 1984 clocks. This would mean that the cells corresponding to this counter were on the average (2048-1984)/32 = 2 clocks too short. Which means that the SCT counter must double count twice for a 2 unit cell, three times for a 3 unit cell, and four times for a 4 unit cell in order to keep the relative distances between the TranCk pulse and the SCT pulses constant. There are two difficulties in creating this effect. First of all, it was mentioned earlier that it is desirable to deal with various cell times in order to permit the flexibility of interchanging between variable speed and fixed speed drives. Thus it is desirable to be able to set a parameter which can scale the count in proportion to the cell times and the clock frequency, which will yield a proportional correction number which applies to the clock frequency used and the length of the cells. Second, it is necessary to scale this number down to correspond to the amount of correction that is needed for each cell time. This is accomplished using a device called a Rate Multiplier. A Rate Multiplier is a device which can scale the clock based on the Rate Multiplier constant.

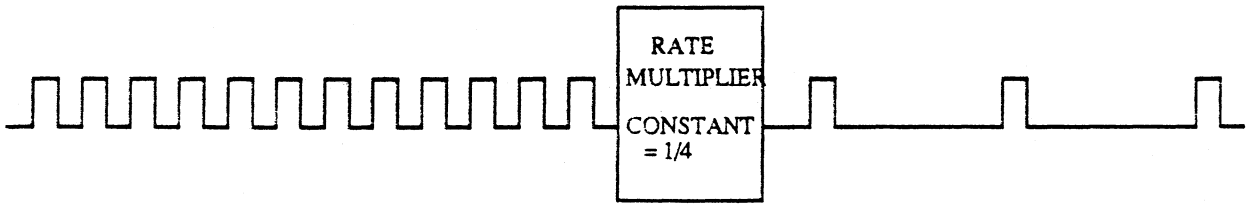


FIGURE 21.

By changing the constant, the number of pulses can be changed. Thus, it is a matter of defining the constant in such a manner that it normalizes the clock to create a function which corrects in the same manner regardless of the clock frequency or the cell times. In this application the Rate Multiplier Constant (K) is defined to be the value which causes an eight bit counter to count from zero to 255 over the period of the 32 samples if every cell is exactly the length that it theoretically should be. Any deviance from this count of 255 can be interpreted as the amount of error per 256 clocks. Using 4 µs cells and a 16 Mhz clock the factor

$$K = \frac{256}{64 \text{ COUNTS PER SAMPLE} * 32 \text{ SAMPLES}} = 1/8$$

OR IN GENERAL

$$K = \frac{256}{\text{DISTANCE BETWEEN CELLS IN µs} * \text{CLOCK FREQUENCY IN Mhz} * \text{\# OF SAMPLES}}$$

The rate multipliers are implemented in the following manner:

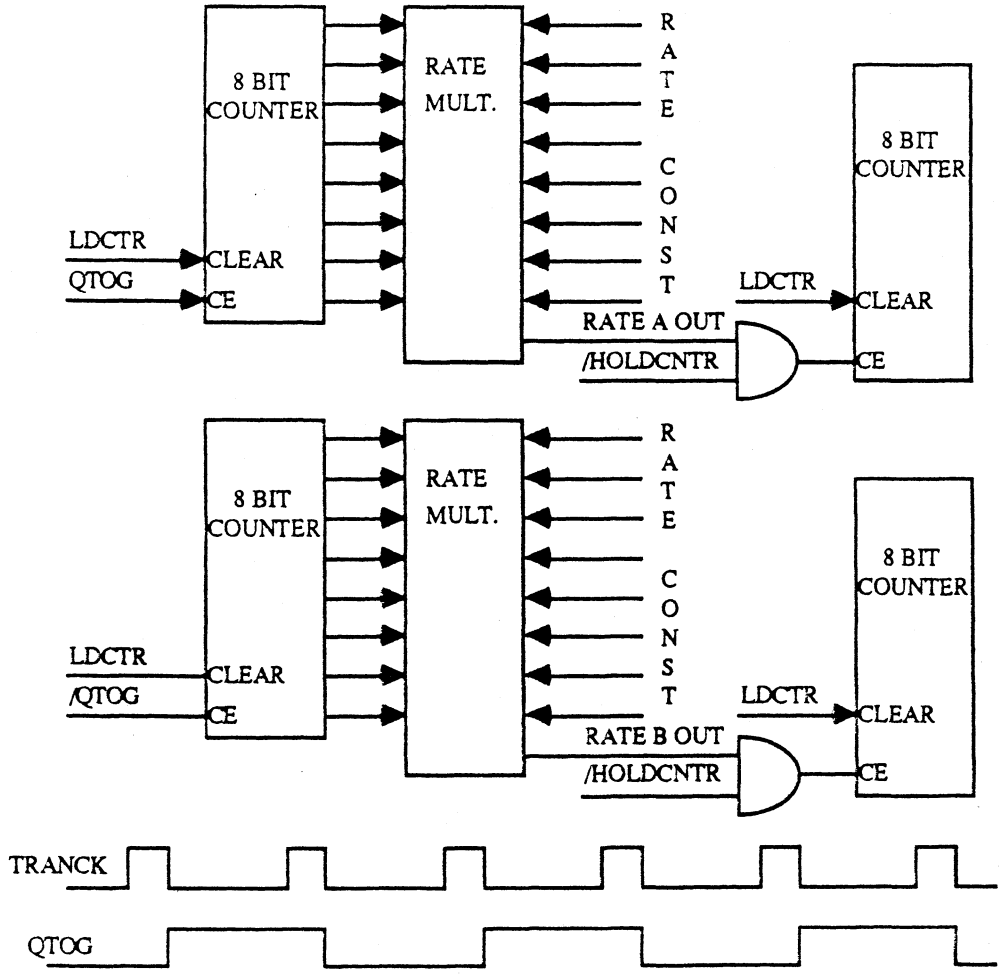


FIGURE 22.

The signal Qtog is used to determine whether the current cell is in the first or second half of the transition pair. The front end counters are used to count the number of clocks between each transition. These values are then scaled down by the Rate Constant (K) and used to count the 256 bit counter. Once the Correction State Machine has counted 32 pairs of minimum cells it will send out a signal called Holdcntr which will lock in the values of the correction. In the example used earlier the number of clocks per cell is 64 and K = 1/8. This implies that during every cell there will be 64/8 = 8 rate clocks. This will cause the 256 bit counter to count 8 times during each cell which will create 8 * 32 = 256 counts for every 32 cell sample. Now suppose that while Qtog is high there turns out to be 68 counts per cell, and while Qtog is low there turns out to be 60 counts per cell. In the first case this would imply 68/8 = 8.5 counts on the 256 bit counter, per cell, which would imply a total count of 8.5 * 32 = 272 counts. In the second case this would imply 60/8 = 7.5 counts on the other 256 bit counter, per cell, which would yield an overall count of 7.5 * 32 = 240. Once these numbers are obtained, the counters are held and these correction numbers are used to correct on all ensuing cells until a new set of "bytes of zeroes" is synched on. The amount of correction and the type of correction depends on which half of the cell pairs the current cell is a part of. The difference between these correction numbers and 256 represents the difference between ideal cell times and actual cell times. In practice the counter would not count past 255, but would instead start over at zero again and count up to 15. This number would then represent the number of clocks per 256 clocks by which the cell was too long. This can be compensated by skipping a count in the SCT counter 16 out of every 256 clocks. This will produce the effect of narrowing the cell to its proper width with respect to the SCT pulses. This can be accomplished using another Rate Multiplier whose multiplier is determined by the distance by which the count exceeded or came up short of 256.

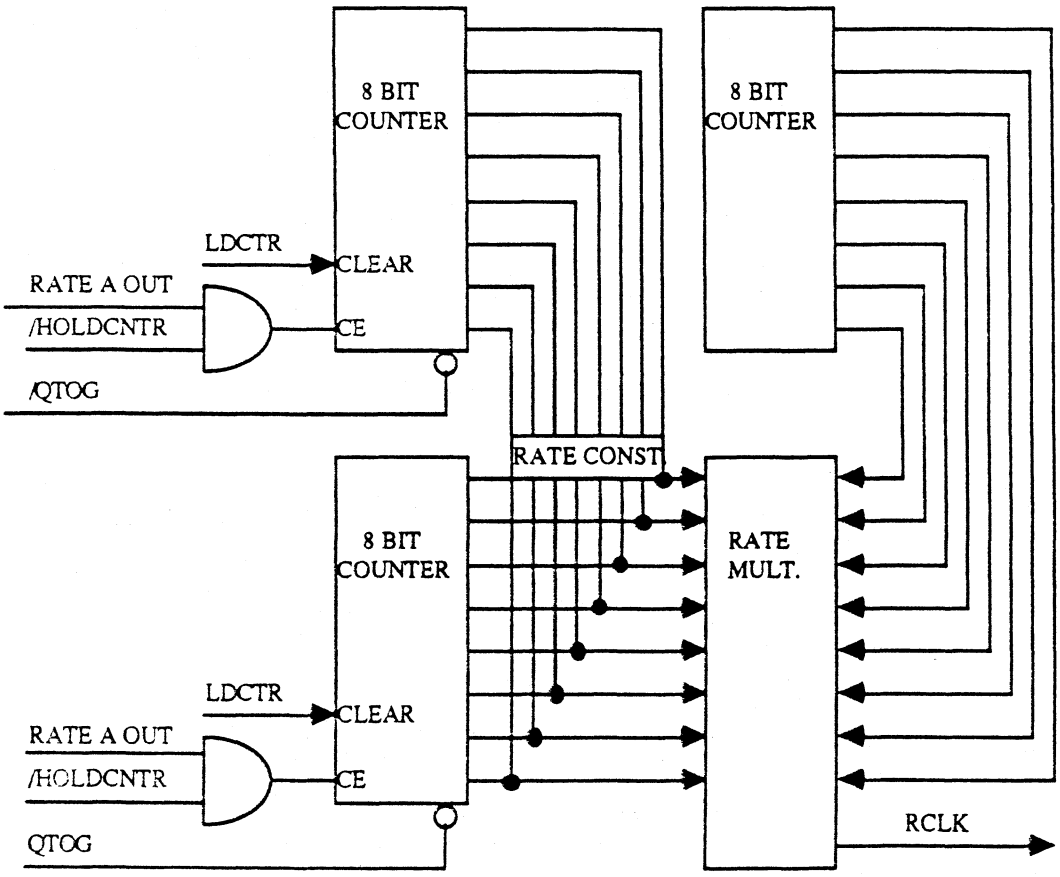


FIGURE 23.

In the case in which the count came up short to 240 counts the number available as the rate constant is 240. This will produce an Rclk which has 240 clocks per 256, but it desired to only have 16 clocks per 256 since the number came up short by only 16 counts. This can be obtained by merely inverting the rate signal which will produce the effect of having 16 clocks per 256. In this case since the count came up short, the cell times must be shorter than expected. Thus, it is desired to shorten the SCT pulses in order to keep the same relationship. This is accomplished by double counting when the rate clock occurs. An important point at this time is that when a skip or double count is generated in the SCT counter, there must also be a skip or double count in the counter which is used for this Rate Multiplier. This is necessary because extra skips can occur or fewer double counts can occur because the relative rates of the two clocks are different. The overall result of the Error Correction machine is that asymmetry in the in the two halfs of a cell pair or a speed error in the drive can be corrected by determining a Correction Number which represents the error over a period of 32 samples. This number can then be used to correct the cell times on the remaining data to be read by causing the SCT counter to skip or double count to compensate for the improper cell times.

4.4 POST COMPENSATION

In the earlier discussion it was shown how the SCT pulses can be used to determine the length of the cells. In reality there are two counters used for accomplishing this. An SCT counter and an LCT counter. The SCT counter loads parameters which are calculated to represent a cell time which has a short cell following it (a 2 unit cell). While the LCT counter loads parameters which are calculated to represent a cell time which has a long cell following it (a 3 or 4 unit cell). This is done because of the effect of Peak Shift which can cause the transition to be delayed, occur early, or occur properly depending on what the current cell is and the next cell is going to be. The parameters loaded must also depend on the previous cell time because this can also cause the current cell time to vary in length due to the Peak Shift effects on the previous transition. Thus the counters have count regions as follows:

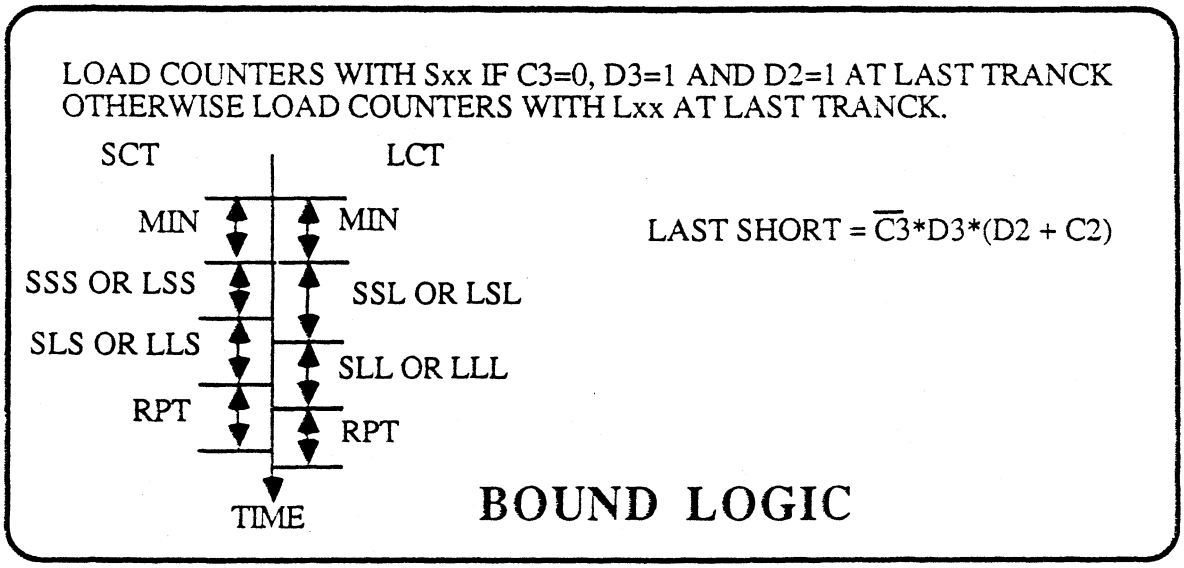


FIGURE 24.

Min represents the minimum for a cell time. If the transition occurs before Min occurs then an error is flagged that the transition is too narrow. If the transition occurs in the next region this indicates that the cell is a 2 unit cell. occurring in the next region implies a 3 unit cell,

and occurring in the final region indicates a 4 unit cell. If the final SCT or LCT pulse called RPT occurs before the transition occurs then this indicates that the pulse was illegally long which will cause another error to be flagged. SLS stands for previous short, current long, next short while SLL stands for previous short, current long, next long. Whether or not the previous was short can be determined by looking at the shift register in the Sequence Resolver, which will be discussed shortly. therefore, if the previous is known to be short then all the parameters which assume a previous short can be loaded. The SCT counter will then load with a value which assumes that the next cell is going to be short, while the LCT counter will load with a value which assumes the next to be long. Where the different bounds become useful is when there is a marginal transition in which the transition occurs between the determining SCT and LCT pulse which causes an uncertainty in the cell time.

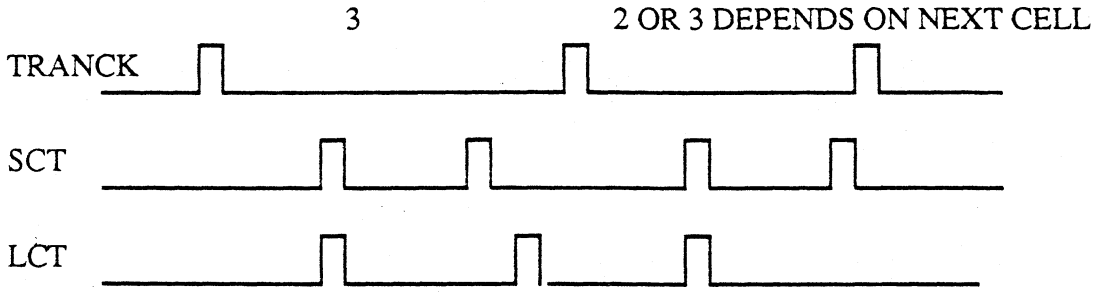


FIGURE 25.

By carrying both sets of information through the Sequence Resolver, the actual cell time can be resolved by shifting the information through a shift register which allows enough delay to make it possible to see what the next cell time is going to be. In the example above if the next cell is a 3 or 4 unit cell then the uncertain cell would be a 2, and if the next cell is a 2 unit cell the cell would be a 3. The Sequencer Resolver is shown in figure 26.

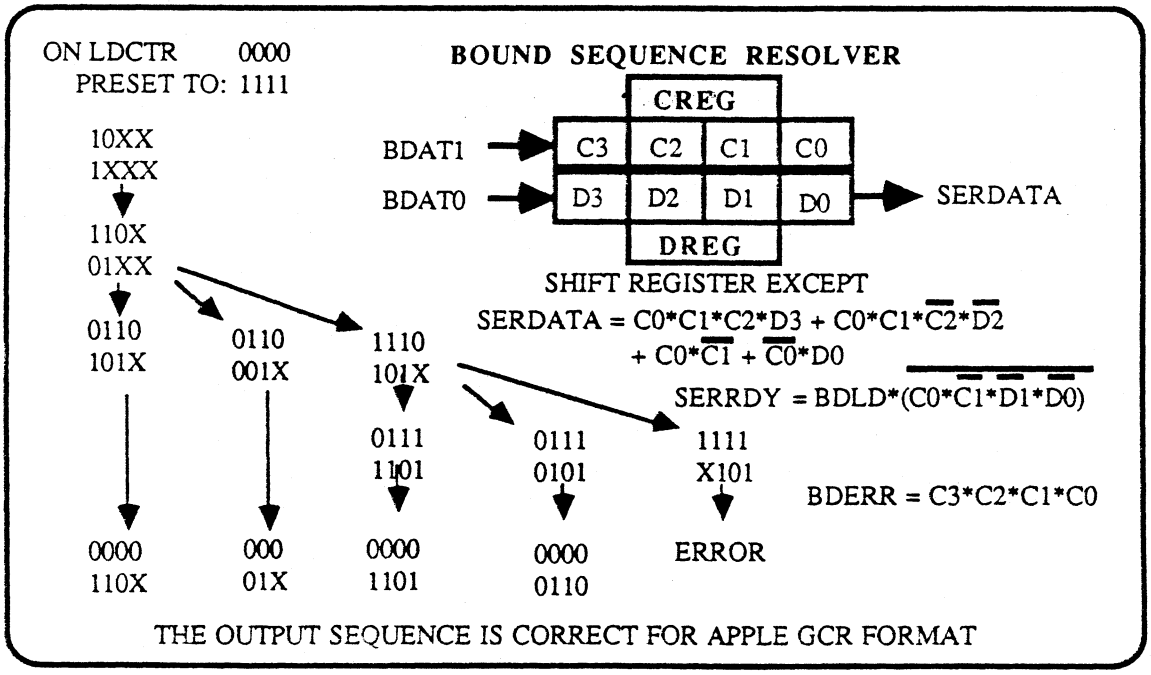


FIGURE 26.

In general, the data is resolved by loading parameters based on whether the previous was long or short and then counting two bounds, one which assumes that the next is short, and one which assumes the next is long. If both machines generate the same number of SCT and LCT pulses between TranCk's then there is no question what the cell time is, but if the number of pulses differ as shown if figure 25 then the cell cannot be resolved until the next cell time is determined. If the next one is long then the data is resolved based on what the long counter considers the cell time to be, but if the next is short then the data is resolved based on what the short counter considers the cell time to be. The bits BDAT0 and BDAT1 are the information that is generated by the SCT and LCT machines.

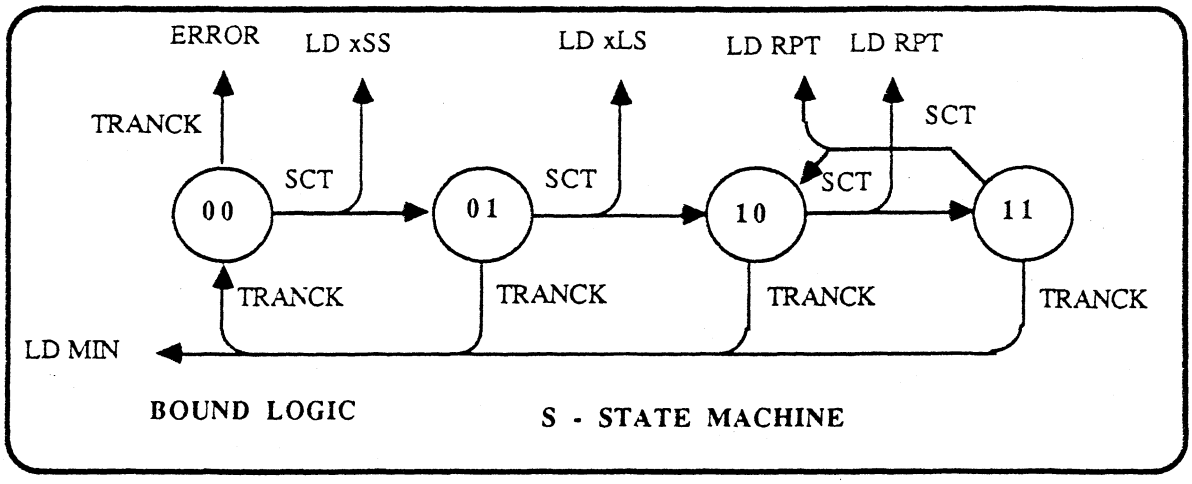


FIGURE 27.

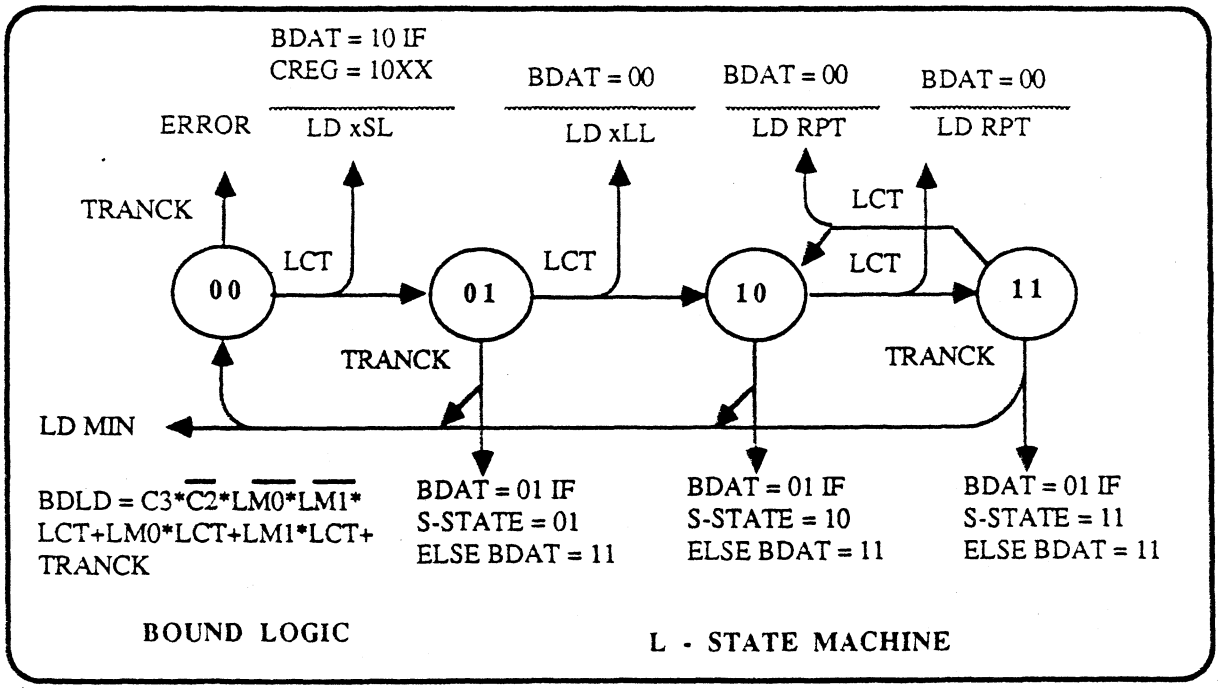


FIGURE 28.

In order to understand the effect of BDAT which is generated based on the states of the S and L machines, a couple of examples will be shown which show how the different cell times are translated into the Serdata And Serrdy signals using the S, L, and Sequence Resolver machines.

Example 3:

Suppose that all the cells are non-marginal (the SCT and LCT machines are in the same state when the transition occurs). This would imply from the L-State Machine diagram that:

	2 UNIT CELL	3 UNIT CELL	4 UNIT CELL
BDAT1	0	0 0	0 0 0
BDAT0	1	1 0	1 0 0

This can be seen by studying the conditions for BDAT. Since the CREG is cleared to zeroes before the read process begins, and since the L machine and S machine are always in the same state at the transition, this is the only data that can possibly be generated. Looking up at the Sequence Resolver it can be seen that since C0 is always a zero the bits will just be shifted into and out of the Sequence Resolver at the same rate at which they are shifted in. Thus, the data coming out of the shift register will just be the BDAT information delayed by 4 clocks. Looking at the data in the above table, it can be seen that this is in the same form as the Trans-Space data that was used on the Write side. This implies that the data has been resolved into a recognizable digital pattern which can be resolved by going through an inverse Trans-Space process to produce the original data that was written.

Example 4:

Suppose in this case that there has been a string of non-marginal cells, then all of a sudden a marginal one occurs. This would create the following BDAT pattern.

BDAT1	1	0
BDAT0	1	0

Which implies the Sequencer Resolver bits will look like:

CREG	1	0	0	X
EREG	1	1	X	X

On the next LCT a BDAT of 1 0 will be shifted into the register which will cause the Sequencer resolver bit to look like:

CREG	1	1	0	0
EREG	0	1	1	X

If the next cell turns out to be a non-marginal 2 unit cell this would imply

CREG	0	1	1	0
EREG	1	0	1	1

This would cause Serdata = 1 to be shifted out.
On the next shift

CREG	x	0	1	1

EREG	x	1	0	1

This would cause Serdata = 0 to be shifted out.
On the next shift

CREG	x	x	0	1
EREG	x	x	1	0

This would cause Serdata = 1 to be shifted out.
Thus the marginal cell was resolved to a 101 pattern which implies that it turned out to be a 3 unit cell. This makes sense because the next cell is a two unit cell which implies that the short counter is the correct one thus the cell must be a 3 instead of a 2 which was indicated by the long counter. If instead the next cell following the marginal cell turned out to be a 3 unit cell this would imply:

CREG	0	1	1	0
EREG	0	0	1	1

This would cause Serdata = 1 to be shifted out.
followed by:

CREG	0	0	1	1
EREG	1	0	0	1

This would cause Serdata = 1 to be shifted out.

Thus the marginal cell in this case is resolved to be a 2 unit cell which makes sense because the longer bound would now be chosen which indicates that the cell is a 2 unit cell rather than a 3 unit cell.

In the Sequence Resolver (figure 26) the left hand side contains all the possible combinations of marginals which can occur and their desired resolution. It can be seen that two marginal cells can occur in a row, but any more than this will cause an error. In order to fully understand the Sequence Resolver it is necessary to study this diagram it great detail.

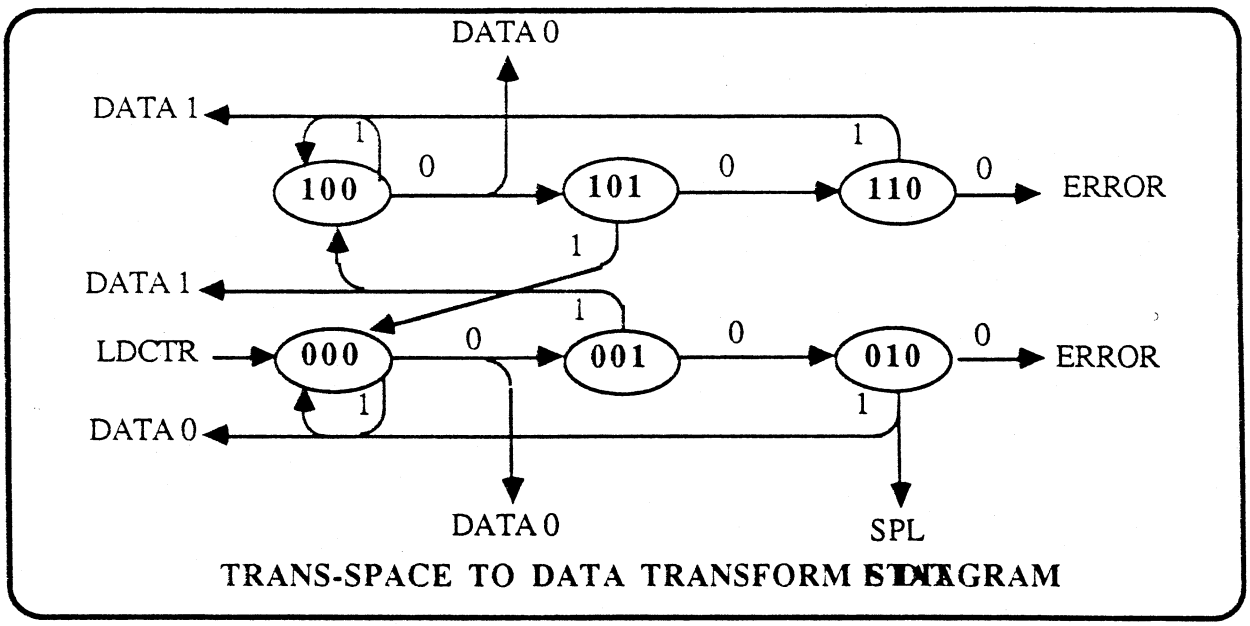
One important point before ending the discussion of the Sequence Resolver is the question of how to preset the parameters for the next cell if the previous is an uncertain 2 or 3 unit cell. What is done is to assume that the previous was a long and load the corresponding parameter. If the second LCT pulse occurs then the current must be long which implies that the previous must have actually been short. Thus, a special parameter is loaded for third LCT and SCT pulse which helps make up for the incorrect assumption. This parameter is determined in such a way that the pulse will occur in the location that it would have had the assumption that the previous was short had always been used.

4.5 DATA TRANSFORMATION

Once the data has been converted into Serdata, which was shown to be the Trans-Space data form, all that is required is to go through an inverse Trans-Space machine which can convert Serdata into the actual data. This machine is shown in the figure below.

DATA TRANSFORMATION

PREVIOUS DATA	CURRENT TRANS-SPACE	RESULT	PREVIOUS DATA	CURRENT TRAN-SPACE	RESULT
1	(1)	1	0	(1)	0
1	(01)	0	0	(01)	01
1	(001)	01	0	(001)	00



TRANS-SPACE TO DATA TRANSFORM STATE DIAGRAM

FIGURE 29.

The simplest way to understand the operation of this machine is to step through an example. In figure 4 it was shown what the MFM pattern would look like for the string of 0's followed by the A1 Mark byte. The corresponding Trans-Space pattern for this wave form would be:

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1

Since this is the pattern which is first dealt with once the Correction Machine locks up on the string of zeroes, it seems the most reasonable to use as an example. The first point that should be noted is that the string of Trans-Space 1's is interpreted to be a string of data 0's because the machine always assumes that what it is locking up on is a string of zeroes. If the Mark byte occurs right after the string of zeroes then it is known to be a valid assumption. If the Mark byte does not occur then the correction machine goes back into the state where it is looking for a string of zeroes. Looking at the table above it can be seen that the Trans-Space 1's represent data 0's since the previous data is assumed to be 0. The first non-minimum cell, a 01, will be interpreted as 01 data since the previous data bit was a 0. The next one will be a 001 with a previous data of 1 which will produce a data pattern of 01. Following this will be a 01 with a previous data of 1 which will cause a data 0. After this is 001 with a data 0 before it. This represents the Mark pattern since a 4 unit cell was preceded by a data bit 0. This then causes a Mark flag to become true which causes the Correction State machine to go into it's final state and also causes the FIFO

to start loading in the bytes as they are read. The Mark pattern will generate a 00 data pattern. The final Trans-Space data is a 01 which is preceded by a data 0. This will cause a data 01. Putting this all together in the data form will yield:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1
|          STRING OF Zeroes          | A1 MARK BYTE |
```

Thus the Trans-Space data has been transformed back into its proper data form.

5. OTHER MODES.

There are FOUR other modes in the chip which make it possible to gain more flexibility in dealing with different types of drives and different data types. These modes are the Drive Option mode, the GCR mode, Time Out mode, and another mode which brings out certain signals which can be used for future expansion..

5.1 DRIVE OPTION

In the discussion of the Write side it was shown that the data is written by toggling a pulse at increments of 2, 3, and 4 units apart.

APPLE TYPE DRIVE

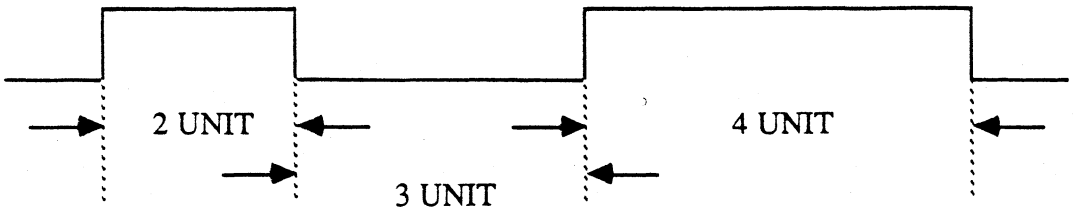


FIGURE 30.

This data pattern to the disk works fine for Apple standard drive, but for IBM type drives the wave form needs to consist of pulses spaced at 2, 3, and 4 units apart.

IBM TYPE DRIVE

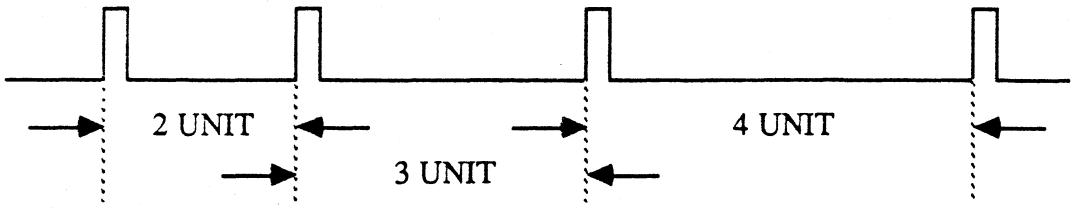
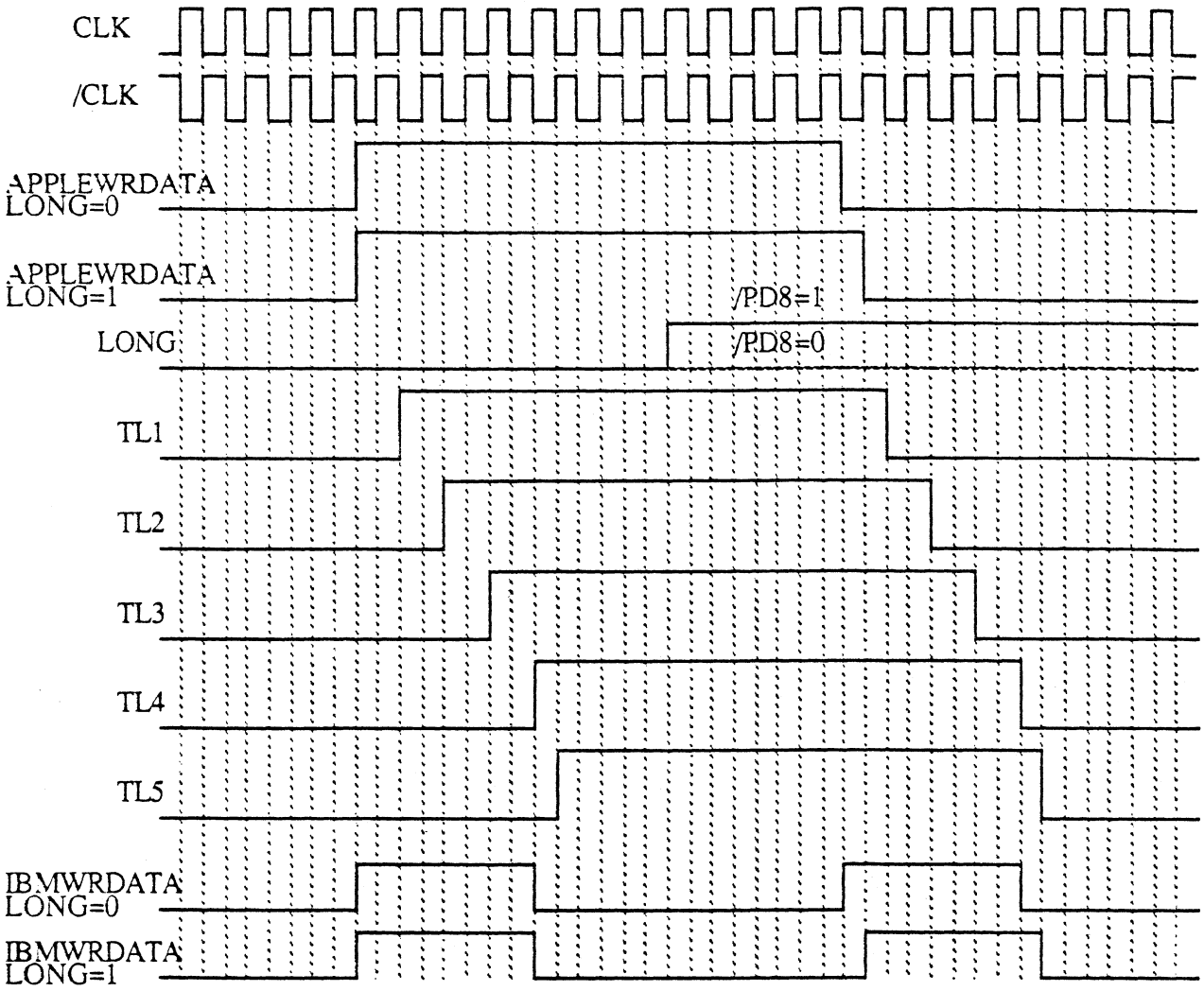


FIGURE 31.

Therefore a mode is added to the chip which makes it possible to get this type of Wrdata signal, so that it can be made possible to deal with both types of drives. This can be accomplished by delaying the the toggling Wrdata pulse, that is already generated, by 4 clocks when not Long (when not shifting by an extra half clock) or by 4 1/2 clocks when Long (extra half shift is desired).



WRDATA = /TL4 * WRDATA * /LONG + TL4 * /WRDATA * /LONG +
/TL5 * WRDATA * LONG + TL5 * /WRDATA * LONG

FIGURE 32.

This delaying of Wrdata makes it possible to create 4 clock wide pulses on each transition of the Wrdata pulse. The trailing pulse can be delayed by half a clock to correspond to the Wrdata being a half clock longer when Long is equal to one. When reading from an IBM style drive, the only difference is that the trailing edge of the Rddata pulse is the valid edge. This problem can be solved by simply inverting the Rddata pulse as it comes in.

5.2 GCR

GCR stands for Group Code Recording. There are many variations to GCR patterns, but the one of interest here is the Apple GCR formats. Apple GCR format consists of patterns of data bytes which always have a leading bit one and whose ones never have more than two zeroes between them.

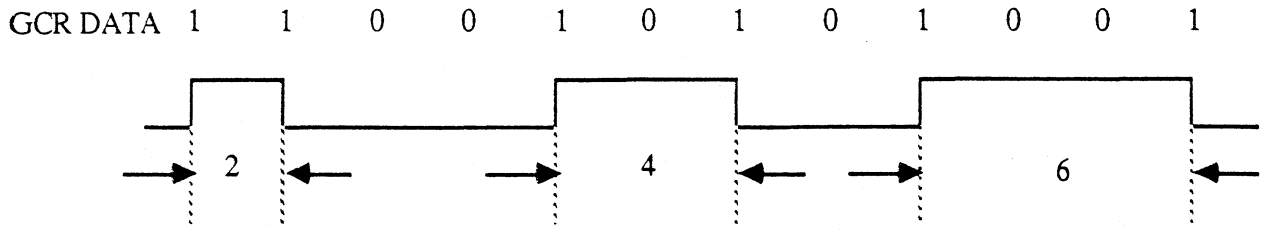


FIGURE 33.

Looking at the data pattern it is easy to see that the GCR data is the same as the Trans-Space data pattern. The only difference is that the cell times are 2, 4, and 6 units long instead of the 2, 3, and 4 unit cells that are used in MFM. Thus, it is possible to write the GCR pattern by merely bypassing the Trans-Space machine since the data is already in Trans-Space form. The only other thing that needs to be done is to set Time1 = Time 0 when calculating the parameters. This will produce the different set of cell times.
Reading the GCR pattern is slightly more complicated. Since the GCR sector format does not contain the bytes of zeroes, it is not possible to synchronism up on the beginning of a sector using the Correction State Machine. This also means that it is not possible to obtain any symmetry information using the Error Correction machine, therefore it is not possible to do any correction for asymmetry or disk speed error. The beginning of a GCR sector consists of a string of ones followed by two zeroes, as follows:

1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1

This pattern can be synched on in the following manner. Since the Bytes of Zeroes do not exist as in the MFM pattern, the Correction State Machine is always set to the 1 1 1 state. This means that everything will always get passed on to the Shift Register. The data can be Resolved into the Trans-Space pattern using the SCT and LCT machines in the same manners as was done for reading MFM data. Which means that all the Post-Comp mechanisms will still work. The only thing that needs to be done differently is to calculate the parameters based on the different cell times. Now, since the Trans-Space data is actually the same as the GCR data, the transformation machine can be bypassed and the data can go straight into the Shift Register. Every time that the high bit in the shift register becomes a 1, the shift register will transfer the byte of data into the FIFO and will clear all its bits to zero. This makes it possible to sync to the above string of ones. For example, suppose the first bit shifted into the shift register is the fifth 1 in the first string of ones above.

1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
| | | | | |

On the next byte it will sync on the third 1, and on the following byte it will sync on the leading 1. From here on out it will always sync on the leading 1 since the byte is only transferred to the FIFO when the leading bit is a 1. The first byte after the sync pattern and all

other GCR bytes will always have a leading 1 which will make it possible to remain synched on the leading bit of each byte of data. The bytes can then be transferred to the processor were it must sort out the data from the Group Code pattern.

5.3 TIME OUT

The chip can be put into Time Out mode by setting a bit in the Setup Register. This mode causes the Drive Enable signals to remain active for 1/2 s (at 16 Mhz) when the Motoron bit is shut off. This makes it possible to begin another Read or Write operation, within this time frame, without having to bring the drive back up to speed.

III. REGISTERS

*** INDICATES WILL CLEAR ON RESET**

A processor can communicate with the ISM via sixteen eight bit registers. These registers provide all the status plus the modes which must be setup for reading or writing data from a disk.

DATA/CORR REGISTER $0 READ/WRITE

The data register is the location were data is read from or written to the FIFO. If a Mark byte is read from this location an error will occur. A read from this location when Action is not set will provide the two bytes of Error Correction information. The register is setup to toggle between the two bytes on successive reads thus providing both bytes of information. If there is still valid data to be read when Action is not set, it can be read by reading the Mark Register.

MARK REGISTER $1 READ/WRITE

This location is used for reading and writing Mark bytes. Writing to this location will cause the missing transition between the two zeroes to occur. Reading from this location will allow a Mark byte to be read without causing an error.

ERROR REGISTER $2 READ

This location provides information on the type of error that has occurred. If any of these bits become set, an error flag will be set in the Handshake Register. Once any error has become set no other error can be set until the register is cleared. Reading the register will cause the register to clear or a reset will cause the register to clear. This register must be cleared prior to beginning a read or write operation.

* DATA BIT 0 = 1 UNDERRUN FIFO

In Write mode this error indicates that the FIFO is being Underrun by the processor. In other words, the FIFO is empty and the processor has not acknowledged the handshake by writing another byte. In read mode this error indicates that the FIFO has two bytes to be read, but the processor is not reading them fast enough.

* DATA BIT 1 = 1 MARK IN DATA

This error indicates that a byte which was read from the Data location was a Mark byte.

* DATA BIT 2 = 1 OVERRUNFIFO

In write mode this bit indicates that the processor is writing faster than the FIFO is requesting bytes. In read mode this bit indicates that the processor is reading bytes faster than they are available.

* DATA BIT 3 = 1 CORRECTION ERROR

This bit indicates that the correction number obtained in the Error Correction Machine is so large that the error cannot be corrected for.

* DATA BIT 4 = 1 TRANSITION TOO NARROW

This bit indicates that the transition occurred before the first SCT pulse which indicates that the cell was too narrow to be a legal cell.

* DATA BIT 5 = 1 TRANSITION TOO WIDE

This error indicates that the fourth SCT pulse occurred before the transition which implies that the transition was too wide to be a valid cell.

* DATA BIT 6 = 1 UNRESOLVED TRANSITION

This error indicates that there were three marginal transitions in a row which implies that the transitions cannot be resolved.

DATA BIT 7 NOT USED

WRITE CRC $2 WRITE

A write to this location will set a status in the FIFO which will cause the CRC bytes to be written on the disk. Since the status bit moves through the FIFO, the CRC bytes will shift out after the last bit of data is written.

PARAMETER DATA REGISTER $3 READ/WRITE

This is the location where the sixteen bytes of parameter data is written and read. This register consists of a counter which increments the RAM address every time that a write or read to this location occurs. Thus, the sixteen bytes of data can be written or read by successively writing to or reading from this location. The increment counter presets the addresses to zero any time that a write to the Write Zeroes ($6) location occurs or a /Reset occurs. The data is stored in RAM in the following sequence:

RAM ADDRESS	PARAMETER
0 0 0 0	MIN CELL TIME
0 0 0 1	CORRECTION MULTIPLIER
0 0 1 0	SSL
0 0 1 1	SSS
0 1 0 0	SLL
0 1 0 1	SLS
0 1 1 0	RPT
0 1 1 1	CSLS
1 0 0 0	LSL
1 0 0 1	LSS
1 0 1 0	LLL
1 0 1 1	LLS
1 1 0 0	LATE/NORM
1 1 0 1	TIME 0
1 1 1 0	EARLY/NORM
1 1 1 1	TIME 1

PHASE REGISTER **$4 READ/WRITE**

This register is used to read and write the phase lines which are used to control or read status from the disk drive. There are four phase lines which can independently be programmed as either inputs or outputs depending on the state of the other four bits. The Phase lines default to outputs on Reset.

* DATA BIT 0 is used to set the polarity of the PHASE 0 line when programmed as an output.

* DATA BIT 1 is used to set the polarity of the PHASE 1 line when programmed as an output.

* DATA BIT 2 is used to set the polarity of the PHASE 2 line when programmed as an output.

* DATA BIT 3 is used to set the polarity of the PHASE 3 line when programmed as an output.

DATA BIT 4 = 0 Indicates that the PHASE 0 line is an input.
DATA BIT 4 = 1 Indicates that the PHASE 0 line is an output.

DATA BIT 5 = 0 Indicates that the PHASE 1 line is an input.
DATA BIT 5 = 1 Indicates that the PHASE 1 line is an output.

DATA BIT 6 = 0 Indicates that the PHASE 2 line is an input.
DATA BIT 6 = 1 Indicates that the PHASE 2 line is an output.

DATA BIT 7 = 0 Indicates that the PHASE 3 line is an input.
DATA BIT 7 = 1 Indicates that the PHASE 3 line is an output.

SETUP REGISTER $5 READ/WRITE

This register is used to set the chip into its various modes. This register will reset to all zeroes when a Reset occurs.

* DATA BIT 0 = 0 Will cause Q3/HEDSEL pin to be an input.
  DATA BIT 0 = 1 Will cause Q3/HEDSEL pin to be an output.

* DATA BIT 1 = 0 3.5 not selected.
  DATA BIT 1 = 1 3.5 selected.

* DATA BIT 2 = 0 Normal operation.
  DATA BIT 2 = 1 Sets the chip into GCR mode.

* DATA BIT 3 = 0 Normal operation.
  DATA BIT 3 = 1 Causes the internal clock frequency to be divided by two.

* DATA BIT 4 = 0 Disables the Error Correction Machine.
  DATA BIT 4 = 1 Enables the Error Correction Machine.

* DATA BIT 5 = 0 Sets up the read and write signals for a Apple type drive.
  DATA BIT 5 = 1 Sets up the read and write signals for a IBM type drive.

* DATA BIT 6 = 0 Normal operation.
  DATA BIT 6 = 1 Causes the read and write Trans-Space logic to be bypassed. This bit must be set whenever the GCR modes is set.

* DATA BIT 7 = 0 Will produce no timeout in turning off the Motoron bit.
  DATA BIT 7 = 1 Causes the Motoron bit to stay on for ~ 1/2 s (at 16 Mhz) after it is disabled.

TEST MODE = Bit 2 = 1 and Bit 4 = 1.This combination should always be avoided!

HANDSHAKE REGISTER $7 READ

DATA BIT 0 = 1 MARK

Indicates that the next byte to be read from the FIFO is a Mark byte

DATA BIT 1 = 0 CRC ZERO

Indicates that the CRC Register became all zeroes when the second CRC byte passed through the register. This bit is valid when the second CRC byte is the next to be read from the FIFO.

DATA BIT 2 Is used to read the Rddata signal from the drive.

DATA BIT 3 Is used to read the Sense status signal from the drive.

DATA BIT 4 This bit is a one if bit 7 of the Mode register is a one or if the timeout counter is timing out.

DATA BIT 5 = 1 ERROR

Indicates that one of the bits in the Error Register has been set to a one. This bit is cleared by reading the Error Register.

DATA BIT 6 = 1 DAT2BYTES

In write mode this bit indicates that there are two bytes of available space in the FIFO. In read mode this bit indicates that there are two bytes to be read from the FIFO.

DATA BIT 7 = 1 DAT1BYTE

In write mode this bit indicates that there is one byte of available space in the FIFO. In read mode this bit indicates that there is one bit to be read from the FIFO.

MODE REGISTER $6 WRITE ZEROES $7 WRITE ONES

This register is used to set the various status bits of the chip. A bit can be set to zero by writing to the Write Zeroes location with the corresponding bit set to a one. A bit can be set to a one by writing to the Write Ones location with the corresponding bit set to a one. This Scheme is used in order to make it possible to modify a particular bit without have to re-write the entire register. The register is cleared to zeroes when a Reset occurs. The Action bit will be cleared any time there is any Error while writing.

* DATA BIT 0 = 0 Normal operation
  DATA BIT 0 = 1

This bit is used to clear the FIFO. This bit must be set and then cleared on successive operations. Read or Write mode must be set prior to setting this bit since the FIFO will clear to opposite states depending upon whether a write or read operation is about to take place.

* DATA BIT 1 = 0 Drive 1 not enabled.
  DATA BIT 1 = 1 Drive 1 enabled

* DATA BIT 2 = 0 Drive 2 not enabled.
  DATA BIT 2 = 1 Drive 2 enabled.

* DATA BIT 3 = 0 Action not set.
  DATA BIT 4 = 1 Action set.

This bit is used to start the read and write operation. This bit should only be set after everything else has been setup. When writing, two bytes of data should be written into the FIFO prior to setting this bit in order for there to be something in the FIFO to start shifting immediately, rather than having as extra byte of garbage shifted onto the disk. This bit will automatically clear if an error occurs while in write mode, but will not automatically clear if an error occurs while in read mode.

* DATA BIT 4 = 0 Sets the chip into Read mode.
  DATA BIT 4 = 1 Sets the chip into Write mode.

DATA BIT 5 = 0 Selects side 0 on the Drive.
DATA BIT 5 = 1 Selects side 1 on the Drive.

DATA BIT 6 = 1 This bit is reserved for future expansion and will always read back a 1

* DATA BIT 7 = 0 Motoron disabled.
  DATA BIT 7 = 1 Causes the Enable 1 and Enable 2 signals to be turned on to the drive. This bit must not be cleared until after the Action bit is cleared and must be set prior to setting Action

READ STATUS REGISTER $6 READ

This register is used to read back the status of the Mode register.

DC CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Limits		Unit
		Min	Max	
V_{DD}	Supply Voltage	-0.3	+7.0	V
V_I	Input Voltage	-0.3	V_{DD} +0.3	V
V_O	Output Voltage	-0.3	V_{DD} +0.3	V
I_{IN}	Input Current	-10	+10	mA
T_{LEAD}	Lead Temp (Solder, 10 sec)	300		°C
T_{OP}	Operating Ambient Temp Range	0	+70	°C
T_{ST}	Storage Temperature Range	-55	+150	°C
P_D	Power Dissipation		250	mW

DC CHARACTERISTICS (all Guaranteed Operating Conditions)

Symbol	Parameter	Limits		Unit
		Min	Max	
I_{DD}	Supply Current		50	mA
V_{IL}	Input Low Voltage		+0.8	V
V_{IH}	Input High Voltage	+2.0		V
V_{OH}	Output High Voltage (I_{OH} = +3.2 mA)	+2.4		V
V_{OL}	Output Low Voltage			
	/WRREQ, PHASE 1 (I_{OL}= +10.0 mA)		+0.4	V
	/ENBL1, /ENBL2 (I_{OL}= +5.0 mA)		+0.4	V
	All Others (I_{OL}= +3.2 mA)		+0.4	V
I_{IL}	Input Leakage Current (V_{IN} = V_{DD} OR GND)	-10	+10	uA
I_{OZ}	Output Leak. Current (3 State,O.C. Outputs)	-10	+10	uA
R_{IL}	Pull Up Resistance (V_{IL} = GND) RDDATA, SENSE	5	20	KΩ

FIGURE 35.

DC CHARACTERISTICS (Continued)

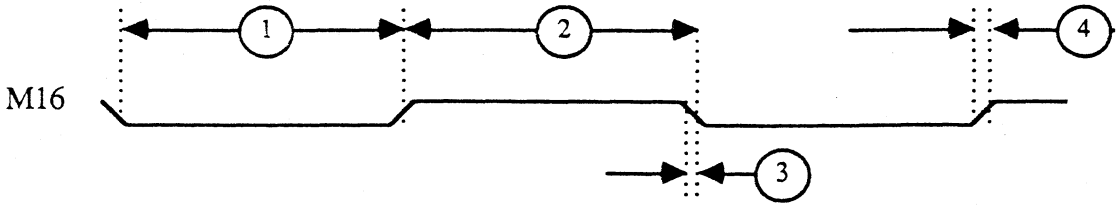
GUARANTEED OPERATING CONDITIONS

Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
V_{DD}	Supply Voltage	+4.75	+5.0	+5.25	V
T_{OP}	Operating Ambient Temp Range	0		+70	°C

FIGURE 36.

AC TIMING CHARACTERISTICS

GENERAL CLOCK TIMING

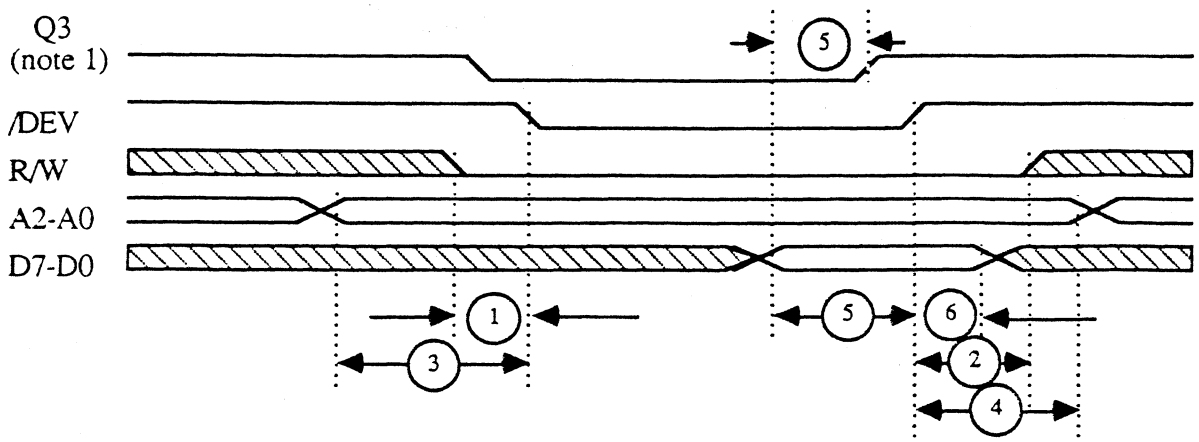


Symbol	Description	Limits	
		Min	Max
① $t_{wCK(LO)}$	M16 Clock Low Width	30ns	DC
② $t_{wCK(HI)}$	M16 Clock High Width	30ns	DC
t_{cyc}	M16 Clock period	60ns	DC
③ t_f	Fall Time (All outputs - 90% to 10%) Output Loading = 50 pF to Ground		10ns
④ t_r	Rise Time (All outputs - 10% to 90%) Output Loading = 50 pF to Ground		10ns
③ t_f	M16 Fall Time		10ns
④ t_r	M16 Rise Time		10ns

FIGURE 37.

AC TIMING CHARACTERISTICS (Continued)

PROCESSOR WRITE TIMING



Symbol	Description	Limits	
		Min	Max
① $t_{sR/W(DEVf)}$	R/W Low to /DEV Fall Setup Time	15ns	
② $t_{hDEVr(R/W)}$	/DEV Rise to R/W Low Hold Time	0ns	
③ $t_{sA(DEVf)}$	Address Valid to /DEV Fall Setup Time	15ns	
④ $t_{hDEVr(A)}$	/DEV Rise to Address Valid Hold Time	0ns	
⑤ $t_{sD(DEVr)}$	Data Valid to (/DEV or Q3) Rise Setup Time	35ns	
⑥ $t_{hDEVr(D)}$	(/DEV or Q3) Rise to Data Valid Hold Time	10ns	

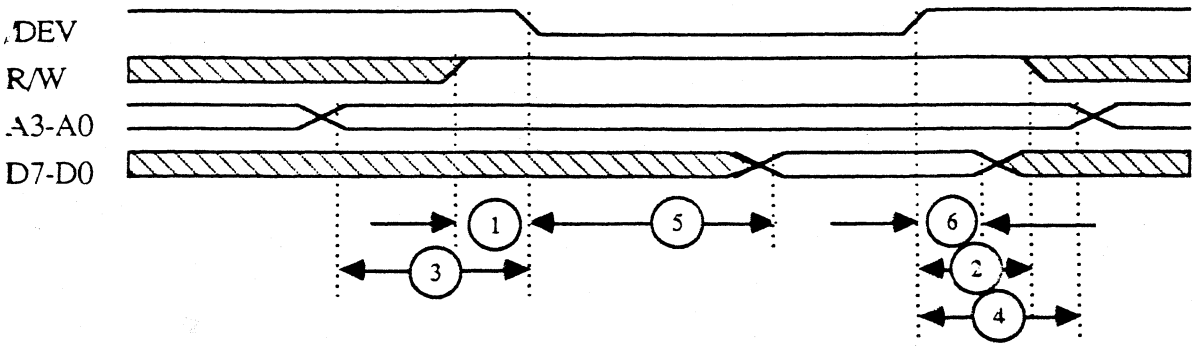
FIGURE 38.

(note 1) Q3 is used to latch the data if bit 0 of the Setup register is low only if a Q3 occurs during /DEV. If no Q3 occurs then the data is latched on the rising edge of /DEV.

If bit 0 of the Setup register is high then the data is always latched on /DEV.

AC TIMING CHARACTERISTICS (Continued)

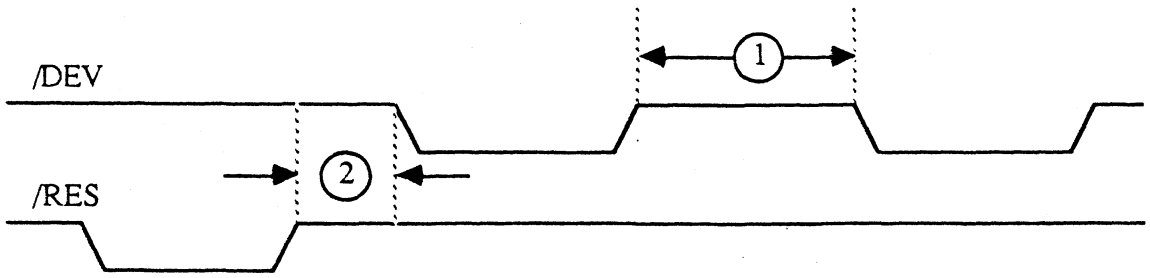
PROCESSOR READ TIMING



Symbol	Description	Limits	
		Min	Max
① $t_{sR/W(DEVf)}$	R/W High to /DEV Fall Setup Time	15ns	
② $t_{hDEVr(R/W)}$	/DEV Rise to R/W High Hold Time	0ns	
③ $t_{sA(DEVf)}$	Address Valid to /DEV Fall Setup Time	15ns	
④ $t_{hDEVr(A)}$	/DEV Rise to Address Valid Hold Time	0ns	
⑤ $t_{dDEVf(D)}$	/DEV Fall to Data Valid Delay Time		95ns
⑥ $t_{hDEVr(D)}$	/DEV Rise to Data Valid Hold Time	0ns	

FIGURE 39.

/DEV REQUIREMENTS



Symbol	Description	Limits	
		Min	Max
① $t_{/DEVr(/DEVf)}$	/DEV Rise to Next /DEV Fall	note 1	
② $t_{/RESr(/DEVf)}$	/RES Rise to /DEV Fall	note 2	

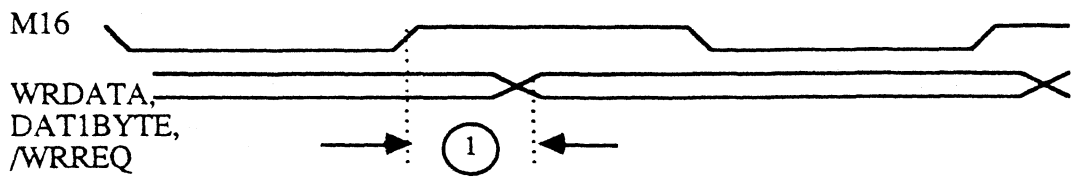
note 1: 4 clocks if Bit 3 of the SETUP Register = 0
8 clocks if Bit 3 of the SETUP Register = 1

note 2: 2 clocks if Bit 3 of the SETUP Register = 0
4 clocks if Bit 3 of the SETUP Register = 1

FIGURE 40.

AC TIMING CHARACTERISTICS (Continued)

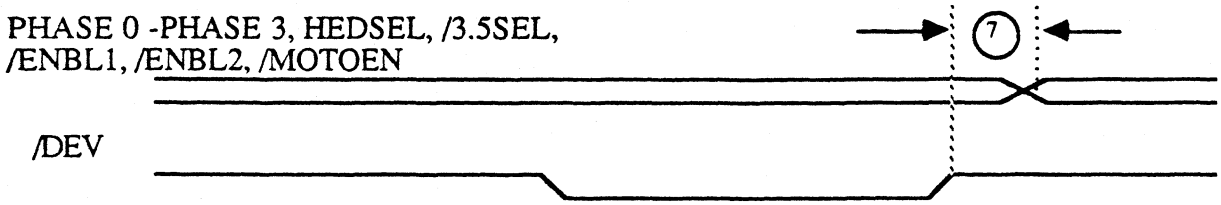
CLOCKED OUTPUT TIMING



Symbol	Description	Limits	
		Min	Max
① $t_{dCKr(WRD)}$	M16 Clock Rise to Output Delay Time		50ns

FIGURE 41

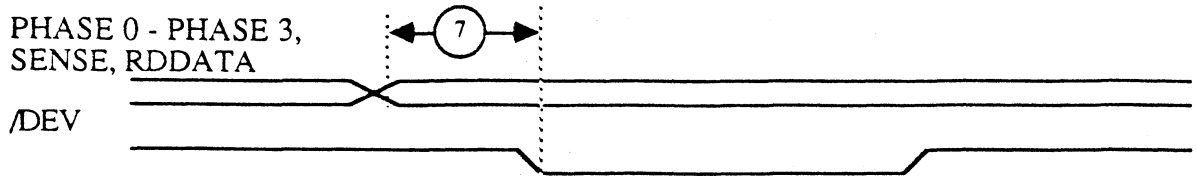
/DEV OUTPUT TIMING



Symbol	Description	Limits	
		Min	Max
⑦ $t_{dDEVr(CNT)}$	/DEV Rise to Control Output Valid Delay Time		35ns

FIGURE 42

INPUT SETUP TIMES



Symbol	Description	Limits	
		Min	Max
(7) $t_{sSTAT(DEVf)}$	Asynch Input Valid to /DEV Fall Setup Time		15ns

FIGURE 43

Reliability and Testing

Final Test

16 Mhz Sentry test program.

Marking

010-0101-0

Ⓒ Ⓜ APPLE 1987

ElectroStatic Discharge Requirements

The ASIC must meet MIL-STD-38510 ESD zap test requirements of 400 Volts from a 100 pF charging capacitor and 1.5 KOhm series resistance. The ASIC must pass the parametric and functional performance specifications after the zap test has been performed. Proper precautions shall be taken by the vendor to insure that ESD damage does not occur to the parts during packaging and shipment. The devices shall be delivered to the customer in rails or on conductive foam that resist generation of triboelectric charge as the device is inserted into, removed from , or allowed to slide around in it. The rails or foam shall further be placed in a static shielding bag for shipment in order to eliminate damage from external stray fields.

Latchup Susceptibility

Diligent effort in the design of the ASIC will be undertaken to reduce the susceptibility of the ASIC to DC, transient, and substrate current latchup conditions. This reduction of susceptibility shall be accomplished without significant degredation of MOS performance and without increasing the die size significantly.