Dandelion Disk Controller

Logic and Documentation Drawings

DDC00.sily - This page

DDC01.sily - Overvew of Disk Controller for SA100x and SA4xxx rigid disks

DDC02.sily - Processor Interface Block Diagram

DDC03.sily - Serializer/DeSerializer Block Diagram

DDC04.sily - Output Conditioning circuits Block Diagram

DDC05.sily - Input Contitioning circuits Block Diagram

DDC06.sily - Disk Cable Connections for PWB card

HSIO47.sil - Control and Write Data registers

HSIO48.sil - Status and Test Drivers, Read Data Register

HSIO49.sil - Service Request, Overrun and Word Status Buffer

HSIO50.sil - Serializer / DeSerializer

HSIO51.sil - Field and Word State Machine

HSIO52.sil - MFM Encoding, Pre-Compensation and Address Mark Generation

HSIIO53.sil - Disk Output Buffers and Drivers

HSIO54.sil - Phase Decoder Logic

HSIO55.sil - Disk Input Buffers and Receivers

HSIO56.sil - Disk Clocks, Input Data Mux.

HSIO57.sil - Data Separator and Address Mark Detection

HSIO58.sil - Input Multiplexer

sHSIO59.sil - Cables and Discretes for Stichweld card

pHSIO59.sil - Cables and Discretes for PWB

sHSIO60.sil - Discrete Phase Comparator for Stichweld card

pHSIO60.sil - Phase Decoder Oscillator for PWB

pHSIO61.sil - Phase Comparator for PWB

HSIO62.sil - Testability signals

. Descriptions of the drawing contents follow the drawings. Each drawing has an accompanying Bravo file describing it.

Other Documentation

This file is:	[IRIS]<Workstation>HSIO>DDC-Rev-Q.press	Logic & Documentation Drawings
	[Iris]<Workstation>HSIO>DDC-Rev-Q.SilDm	Logic drawings in Sil format
	[Iris]<Workstation>HSIO>DDC-Rev-Q.DocDm	Documentation in Sil and Bravo formats
	[IRIS]<Workstation>HSIO>Proms>DDCProms-Rev-A.dm	Prom Programs

File: DDC01.bravo in [Iris]<Workstation>HSIO>DDC-Rev-Q.DocDm
Contents: Overview of DDC01.sily, a drawing of the Dandelion Disk Controller.

Overview

This drawing shows a block diagram of the Dandelion Disk Controller. It identifies the major components of the system and their connections.

There are four major blocks in the Dandelion Disk Controller (DDC). They are the Input Conditioning, Output Conditioning, Processor Interface and Serializer/DeSerializer circuits. Disk read data, disk clocks and reference clocks arrive via the Input Conditioning circuits, as do disk status lines. The disk control lines, disk write data and write clocks are sent via the Output Conditioning circuits. The Processor Interface generates microcode service requests, detects the overrun condition and passes data, status and commands along the X-Bus. Disk data is converted from 16 bit parallel words to a serial data stream and back in the Serializer/DeSerializer.

Constraints

Cost

The Dandelion is intended to be a relatively low cost workstation. To this end, the hardware it contains should be minimized. This leads to low manufacturing, testing and service costs. The guiding principle of the controller's design has been that only functions which occur too quickly for microcode to handle or require hardware buffering are implemented in the controller. For example, step pulses may be sent relatively slowly, so the step line is toggled by having the microcode send control words in which the step line is alternately set and reset.

Another result of the cost constraint is that one controller board should serve to control both the SA1000 and the SA4000 drives. It should be able to support drives with 2 to 32 heads. The effort required to change the board from an SA1000 configuration to an SA4000 configuration should be minimized. In fact, it should be limited to unplugging the cables to the SA1000 and plugging them into the SA4000.

Disk Format

The disk is divided into cylinders. Each cylinder represents a distinct position of the read/write heads. Each cylinder is divided into tracks, one per read/write head. The SA1002 drive has 2 heads, the SA4100 has 32. Each track is divided into sectors. There are 28 sectors per track on the SA4000 type drives, 16 sectors per track on SA1000 type drives. Each sector is divided into three fields, Header, Label and Data. The Header field is used to specify the sector's physical position on the disk (cylinder, head and sector numbers), the Label specifies the page's position in the file system and the Data field holds the actual data. Finally, each field is broken into 4 areas. A pattern of all zeros is followed by a synchronization word or address mark which is followed by the field's data and, lastly, a word of CRC checksum. The length of the synchronization pattern varies between drives, it is 7 words on the SA1000 drive and 6 words on the SA4000 drive. A synchronization word of all ones is used to define the first word boundry on the SA4000 drive. An address mark serves a similar purpose on the SA1000 drive. The Header field contains 2 words of data, the Label field 12 words and the Data field 256 words. The CRC checksum word following the data area of each field is used to implement an error detecting code.

One of the constraints on the design is that it must be possible to read, write or verify each field in all sectors of a cylinder in one revolution. This means that in addition to the raw data rate constraint, it must be possible for the microcode to carry out the inter-field, inter-sector and inter-track overhead operations with the hardware available. A design which requires a great deal of setup between sectors or fields may not be acceptable. It should be possible to perform any combination of operations on the fields of a sector.

An exception to this rule is that when a write is performed to one field, further fields of that sector must either also be written or are assumed to be lost. The microcode must also be capable of aborting operations on later fields based on the results of operations on earlier ones. For example, if the Header and Label fields of a sector are to be verified before the Data field is written, the Data write should be aborted if either the Header or Label verify operations fail.

The SA1000 drive does not contain a data separator, the SA4000 drive does contain one. The controller board sends and recieves MFM (Modified Frequency Modulation) encoded data from the SA1000 drive and NRZ (Non Return to Zero) data from the SA4000. The SA1000 data rate is 4.27 MBits/Sec (234 ns/bit). The SA4000 data rate is faster at 7.14 MBits/Sec (140 ns/bit).

Dandelion Architecture

The Dandelion processor is implemented using 2901 byte slice chips. These provide 16 registers readily accessible to the microcode. There are five independent microcode tasks running at any one time, the Disk, Display, Ethernet, I/O Processor and Mesa Emulator. These tasks share the 16 registers, along with the 8 Link and 256 U registers. The Disk task is allowed two of the fast R registers, one link register and 40 U registers.

The I/O tasks are run in round-robin fashion. Each microcode's turn is called a *click*, five clicks comprise a *round*. The disk microcode task is allocated click number 2 in every round. If an IO task does not use its click, the Emulator runs there instead. The processor can execute one main memory reference in parallel with three microinstructions in each click. The time in which each microinstruction executes is called a *cycle*.

The processor executes a microinstruction every 137 ns, a round every 2.05 microseconds. A single 16 bit word may be stored in main memory every round. The disk microcode can sustain a transfer rate of 16 bits/2.05 x 10^{-6} sec or 7.8 MBits/sec. Thus one can only be sure of the microcode executing one click per word transferred from either the SA4000 or the SA1000.

Function Allocation

The verify operation requires that each bit be checked against a template from memory, a CRC checksum be maintained, a memory address updated and a word count decremented. Four pieces of information must be maintained, an address, a word count, the data to be verified and some sort of checksum. While it would be possible to combine the address and word count by requiring all field templates to begin (or end) on page or nibble boundries, this is rather inconvient and messy. The designer has been unable to find an encoding scheme which makes it possible to combine the data to be compared and the checksum. These seem to be the only remotely workable combinations. Hence all four quantities must be kept independently.

The four quantities must be divided between the two R registers in the processor and registers in the controller. The lack of U register speed precludes their use. One must spend an entire click to update one U register (read it, change it, then store it), yet the microcode is only allowed one click per word transferred. Due to the main memory addressing scheme, the address must reside in one of the R registers. Were the checksum to be done in microcode, the microcode would have to read the data to be checked. Because of the close timing in SA4000 operations, the controller data is guaranteed to be stable during only one cycle in the click. Hence, the checksum could not be done in microcode and while the verification is done in hardware because microcode would need two cycles to write the memory data to be verified to the controller and read the disk data to compute its checksum. The disk data could be stable during one or the other cycle, but not both. It would be possible to compute the checksum and maintain the wordcount in the controller while doing the address and verification in microcode, but the microcode would be messy and the status of an operation would be partially in microcode, partially in hardware. The controller as designed allocates the address and the wordcount to microcode and the data and checksum to hardware.

Input Conditioning

As explained above, one controller board should serve for both the SA1000 and SA4000 type drives. This means the board may receive either MFM (SA1000) or NRZ (SA4000) data from the disk. It may either receive its data clock directly from the disk data separator (SA4000) or may derive it from MFM data (SA1000). The clock used to encode or strobe write data may similarly be received from the disk (SA4000) or be derived from the processor clock (SA1000).

The Input Conditioning circuitry has seven jobs.

1. It buffers the disk status signals for use by the Processor Interface.
2. It uses one of the status lines to decide whether the drive is of the SA1000 or SA4000 type. This indicator is used here, in the rest of the controller, in the microcode and in the software.
3. It either passes NRZ data received from a SA4000 disk or translates MFM data received from a SA1000 type disk, sending the resulting NRZ bit stream to the Serializer/DeSerializer.
4. It uses the SA1000/SA4000' flag to pass on the correct clocks to the rest of the board.
5. It produces the AddressMarkFound signal by recognizing MFM address marks. These are used to delineate fields when using the SA1000 drive. A combination of Sector marks and synchronization words are used in the SA4000 drive.
6. It produces the SectorMk signal by either passing on the SA4000's Sector signal or recording the value of bit 14 of an SA1000 address mark. This is done so the microcode can identify a Header field.
7. It produces a consistent value of SeekComplete, either passing the SA1000 version directly or delaying the SA4000 version until the read/write heads have settled.

Processor Interface

This section of the board is responsible for communication between the controller and the Dandelion CP. It receives the X-Bus lines along with the control signals for the Disk controller's ports. Using these, it transmits status and data to the CP or latches control and data words. The disk status is received from the Input Conditioning section of the board, the direct commands (Step, FaultClear, etc) are sent to the disk through the Output Conditioning section. The CRC and verify error status bits are received from the Serializer/DeSerializer. Commands enabling transfer of data are sent there. The 16 bit data words used by the Serializer/DeSerializer are routed through the Processor Interface. With the help of the BitCount state bits from the Serializer/DeSerializer and the SA1000/SA4000' flag, the Processor Interface generates service requests to the microcode and maintains an overrun indicator. This flag is used to disable the disk's WriteEnable signal, ensuring that no more than one word of random data should be written if the transfer is interrupted. This may happen if the processor fails or encounters a Kernel interrupt.

The Processor Interface is responsible for correctly sampling all error indicators. Some, like the CRC error indicator are only valid on some word boundries, others are valid all the time. In addition, control lines that should only change on word boundries (WriteCRC) are sampled in the Processor Interface.

Serializer/DeSerializer

The Dandelion transmits data in discrete 16 bit words, the disk drives in serial data streams. One of the main tasks of the controller is to translate between these two representations. The Serializer/DeSerializer contains a single 16 bit shift register and two 16 bit buffers. During read operations, the shift register is loaded with NRZ data from the Input Conditioning circuits. On word boundries, the ReadData buffer is loaded from the shift register. This is read by the CP in turn through the Processor Interface. During Write operations, the CP sends data to the WriteData buffer. The shift register (SerialNRZ.0..15) is loaded with these words on word boundries. Verify operations are similar to Write operation in that data sent through the WriteData buffers is loaded into the SerialNRZ shift register. The data being read from the disk is compared with that being shifted out the the register. If any bit mis-matches are detected, the VerifyError signal is raised.

To find word boundries, a count of the number of bits in the shift register must be maintained. The Serializer/DeSerializer computes this in BitCount.(0..3). The interpretation of this count varies depending on the operation being performed and the part of the field being transferred. See the Serializer/DeSerializer description for more details.

As mentioned above, the verify check is done here. The error signal is cleared between each transfer. The CRC check is also done here. The CRC shift register is held preset until the end of the synchronization word passes. Its input is at the input of the shift register during read and verify operations and at the shift register output during write operations. The WriteCRC signal causes the contents of the CRC generator/checker (a Fairchild 9401) to be inserted in to the output data stream. This signal is set by the microcode after the last data word of a field is sent to the controller.

Output Conditioning

Control signals leaving the controller for the disk must be buffered. This is done in the Output Conditioning circuits. The data is either passed on as NRZ data to a SA4000 drive or translated to MFM format and pre-compensated before being sent to a SA1000 drive. The type of clock sent to the drive depends on the drive. The SA1000 type drives require a TimingClock with frequency 1/16 of the bit rate. The SA4000 type drives require a WriteClock with the same period as the bits in the data stream. The type is clock is selected in the Input Conditioning circuits and passed through the Output Conditioning section.

When connected to a SA1000 type drive, the Output Conditioning circuits are also responsible for writing address marks at the beginning of each field. These marks are used to identify the clock|data phase of the incoming MFM data, define the first word boundry in the field and tell what type of field is being read. The address mark is a sequence of pulses which form an illegal MFM string but still can be recognized by the Input Conditioning circuits. Some pulse strings are illegal but may not be transmitted correctly. In particular, any sequence having no pulses for two full bit times is an illegal MFM string, but may not be read correctly by the drive because of its automatic gain control (it will keep increasing the gain until something is found).

File: DDC02.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: A Description of DDC02.sil, the Processor Interface circuitry of the Dandelion Disk Controller

Overview

This drawing is a block diagram of the Processor Interface section of the Dandelion Disk Controller. This section is responsible for buffering data sent between the Dandelion CP and the controller, transmitting and receiving this data according to the Dandelion bus protocols and requesting service from the Disk microcode.

Constraints

The controller has been designed with the idea of minimizing the amount of hardware used. As much functionality as possible has been left in the microcode and software. This results in fairly simple controller hardware.

Many of the lines used to control the disk are set directly by microcode and are ignored by the controller. For example, the Step and Direction lines controlling the disk's head motion are merely bits in the control register that are relayed directly to the drive. The same is true for many of the status signals returned by the drive, they are read and interpreted by the microcode or software.

The controller contains one word of buffering for write and verify operations and one word for read operations. The Dandelion architecture allows the designer to calculate the minimum and maximum latency between a service request and the processor's response and ensure an overrun never occurs in normal operation. If the disk microcode stops servicing the hardware, write operations must be disabled to restrict the amount of random data written on the disk.

Control Register

This 16 bit register receives its inputs from the X-Bus, sending them to both the disk drive and to the controller. It is reset by IOPReset'. The control bits are arranged so that when reset, the controller is dormant.

Status/Test Multiplexer

Three types of 16 bit quantities may be read from the controller. One is data from the disk, the second is the status of the current disk operation, the third is a group of test points on the disk and display controllers. The first will be discussed below under Read Data Register. The second two are independently sent to the X bus. One might also think of them as being multiplexed via tri-state drivers. The operation status is composed of some lines from the drive itself (Track00, DriveNotReady, etc) and some from the controller (Verify Error, Overrun, etc). These are the normal lines read using the ←KStatus command to guide the execution of a disk operation. The test lines are read using the ←KTest command by diagnostic microcode to directly test the control and status lines leading to the disk.

Some of the Status signals should only be sampled on word boundries. The CRC error flag, for instance, is only valid after the last bit of the CRC checksum has been seen. Sampling on word boundries also gives the microcode a chance to turn off a data transfer before the final status flags have been changed to reflect the random bits following the data. This sampling is done by the Word Status Register.

Write Data Register

Data is sent from the processor to the controller in 16 bit words. These are in turn loaded into the SerialNRZ shift register. The words are buffered in the Write Data register before being loaded into the shift register. The buffer is pre-loaded before a transfer begins and is loaded by the microcode in response to a service request thereafter. By calculating the minimum and maximum latencies between request and service, one may be assured that the buffer is always loaded after the previous word has been used but before the current word is needed.

Read Data Register

Like the Write Data register, this is a single word of 16 bits. It is loaded from the SerialNRZ shift register each time a word boundry passes. Just before it is loaded, a service request is sent, asking the disk microcode to remove the word. As with the Write Data buffer, one may assure oneself that this will always happen after the buffer is loaded but before it is loaded again.

Since the NRZ clock is used to drive both the state machine that produces WordBoundry' and the shift register, sampling SerialNRZ from that shift register with WordBoundry' directly would be hazardous. Instead, the Read Data buffer register is inserted before the actual Read Data register. This buffer register is clocked with inverted NRZ clock, so it samples a stable shift register (max prop delay = 35 ns, min bit clock period = 140 ns and 70 > 35) and the Read Data register samples stable ReadData lines.

A wrap-around feature has been included in this controller allowing diagnostic microcode to verify that data may be written and read correctly. The method for using the feature depends on the disk being controlled. The SA4000 provides one clock used throughout the controller. The data sent out is intercepted just before the final drivers and inserted into the input data stream. It is then shifted back into the shift register. By having the microcode start a write operation, then perform reads instead of writes, one may verify that the data being written is correcIty re-received.

The SA1000 supplies no clock. The clock used to write the data is derived from the stable processor clock. If this clock were used for the entire controller, the controller's data separator would not be tested. To test the data separator, we allow it to re-produce the NRZ data using a clock derived from the re-received MFM data stream. Because of jitter between the derived clock and the reference clock, we may not reliably route the re-produced NRZ data back to the shift register. Hence one may not expect to see the data sent in the ReadData register. The address mark recognizer section of the data separator does record the polarity of bit 14 of the address mark however. One may test the controller by sending address marks and sampling the Header tag status bit after each one. Each address mark must be send in its own field, that is the TransferEnable bit should be reset between each one. The Header tag status bit should follow bit 14 of the address mark just written.

Service Request / Overrun Machine

As seen above, the controller must be able to generate service request to its microcode and determine when the requests have not been answered. This is the task of the Service Request / Overrun machine. The timing of Service Requests is based on the BitCount within a word, the time within a field, the operation being performed and the data rate of the disk. Only two disks are supported and the data rates of both are fixed.

During data transfer operations, it is crucial that the disk microcode keep pace with the hardware. If the microcode is early or late, especially during write operations, disk data may be destroyed. The Overrun section of this machine will set the Overrun signal whenever a buffer is needed by the controller before it has been serviced by the microcode. Thereafter, no data may be written and the Service Request signal is

set until the microcode finishes the operation and turns it off. The microcode should sample the status at the end of the operation, testing the Overrun signal.

Service requests may be used not only to synchronize the transmission of data but also to sense status conditions. For example, it would be wasteful to burn 1/5 of the processor to wait for an IndexFound signal. The same holds true for a SeekComplete. These and other signals may be used to generate a service request. The signals are chosen using the Operation field of the Control register.

File: DDC03.bravo in [Iris]<Workstation>HSIO>DDC-A-Rev.DocDm
Contents: A Description of DDC03.sil, the Serializer/DeSerializer portion of the Dandelion Disk Controller

Overview

One of the main tasks of the controller is to translate from the processor's discrete 16 bit data words to the disk's continuous serial data stream and back again. The 16 bit parallel-in, parallel-out shift register shown in this drawing performs this function.

The disk format specifies a 16 bit CRC checksum on the end of each field of a record. The CRC Generator/Checker shown in the drawing calculates the CRC during write operations and checks it during Reads and Verifies.

The disk controller must be able to check the data on the disk against a template in main memory. The results of this check must be available in time to abort an operation on a later field. This check is done in hardware by the Verify Checker.

The machine responsible for keeping track of the current state within a field and within a word is located here. The Field State contains the signals WordBoundry', PLdSerialNRZ' and SyncWdFound. The Word state is given in BitCount.(0..3).

Constraints

The controller must be able to read, write and verify data with a minimum of hardware, hence the use of a single shift register.

A CRC code must be both generated for write operations and checked during read and verify operations. The CRC generator input must sample the output data and produce the first bit of the checksum immediately after the last bit of data. Its control signals must be synchronized on word boundries.

The verify checker hardware should be minimized, hence the use of a bit serial checker.

The Field and Word State machine must be able to produce the Field State signals that correspond to the operation in progress. It must be able to command the Input Conditioning circuits to recognize the address mark if one exists, it must recognize the synchronization word in a data field, it must be able to signal the presence of word boundries in the serial data stream.

SerialNRZ Shift Register

Two 74199s make up this register. They have separate parallel inputs and outputs, obviating the need for in internal three state bus. The serial input and clock are supplied by the Input Conditioning circuits. The parallel load signal, PLdSerialNRZ', is supplied as part of the Field State.

CRC Generator / Checker

The generator portion of this circuit must receive the output bit stream. The checker portion must receive the input bit stream. The CRC checksum must be appended to the output data stream. This is accomplished by having one multiplexer feed either the shift register input or output to the 9401 CRC chip while another selects between the shift register output and the CRC checksum output. The first multiplexer

feeds input data to the CRC chip during read and verify operations and output data during write operations. The second multiplexer sends output data to the disk while WriteCRC is inactive, CRC checksum when WriteCRC is active.

The CRCError flag is a reflection of the state of the CRC register. It is true while the register is non-zero. This is most of the time. The flag is only valid for one bit time after the last bit if the incoming CRC checksum has been read. The flag is sampled at this instant by the Processor Interface using the Field State information.

Verify Checker

A field on the disk is compared to one in memory by having the controller receive the memory version via disk microcode and the disk version simultaneously. The disk version of the field is shifted into the shift register while the memory version is shifted out. The words are aligned such that the disk and memory versions of bit n of a word arrive at the shift register input and output at the same time. The Verify checker compares these two bit streams, signalling an error if they ever disagree. The Verify error signal is sampled by the Processor Interface on word boundries. This gives the microcode a chance to freeze the operation so the results of the check through the last word are available.

The Processor Interface actually delays the Verify error signal by one word time before making it available in the disk status word. This is done so the result of the verify check through the last data word of the field is shown at the same time as the CRC check done through the CRC checksum word following the data. Since the Verify Error signal is valid one word time before the CRC error signal and both should be available to the microcode at the same time, the Verify Error signal is delayed.

Field and Word State Machine

Sections of the controller change their operation as a function of the time within a field. The Data separator searches for an address mark at the beginning of the field, then decodes data. The CRC and Verify checks are only valid if done over the data or data and checksum portions of the field. The shift register must be pre-loaded with the first word to be verified, hence should be loading, not shifting, until the data section arrives. There are other examples. The Field and Word Timing machine produces WordBoundry, PLdSerialNRZ' and SyncWdFound to control these activities. Together they are known as the Field State.

WordBoundry' is used to load the 16 bit buffer registers and sample selected controller command and status bits (WriteCRC, CRCError, VerifyError).

PLdSerialNRZ' is used to parallel load the shift regsiter with data to be written or verified.

SyncWdFound is set as soon as the synchronization word has been recognized at the beginning of the field. It is used to control the CRC and Verify checkers as well as the Service Request and BitCount generation.

It is also necessary for sections of the controller to know the time within a word. This is produced as a four bit number, BitCount.(0..3). It is first used by this machine to produce the Field State signals. The Processor Interface also uses it to generate Service Request and Overrun. The definition of BitCount changes with the state of the machine. While synchronizing with disk data, BitCount equals the number of sync word bits seen so far. During the data transfer, BitCount equals the number of bits shifted into, but not read from, the SerialNRZ shft register MOD 16. For example, during a read operation, the clock edge that causes the shift register to be read shifts in a new bit, so BitCount ranges from 1 to 16 Mod 16 or [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0]. The word boundry is on the 0000 to 0001 transition of BitCount. During a write or verify operation, the clock edge that loads the shift register with a word immediately

follows the one that shifted the last bit of the previous word into SerialNRZ.0. Thus the shift register has at most 15 bits shifted in before they are replaced with a new parallel word. In these cases, BitCount takes on the values [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. The word boundry is on the 1111 to 0000 transition. See the description of the Field and Word State Machine description for complete information.

File: DDC04.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of DDC04.sil, the Output Conditioning circuitry of Dandelion Disk Controller.

Overview

The disk drive and controller are connected with two types of transmission lines, single-ended and differential. The control signals are sent on single-ended lines, the data and clock on differential. The Output Conditioning section of the controller supplies the appropriate drivers for both types.

When connected to a SA1000 type drive, the output data must be encoded in MFM format and pre-compensated. The SA1000 drive also uses address marks to define field boundries. The address marks are generated and the encoding is done here. When connected to a SA4000 type drive, the encoding circuitry is bypassed.

Constraints

Output lines must be driven as per the Shugart specifications. Open-collector drivers are 7406s and 7438s, the differential drivers are 75114s. The active pull-up output of the 75114 is connected, making these active pull-up differential drivers. While this is not the recommended driver for the SA1000, it works quite well.

The MFM encoding is done according to standard MFM rules. NRZ bits are separated into clock and data halves. A flux transition, indicated in this description by a "1", may occur in either half. All NRZ 1s are sent with clock|data = 01. NRZ 0s are sent as clock|data = 10 unless the preceeding NRZ bit was a 1. In that case, the MFM encoding is clock|data = 00.

The pre-compensation rules are taken from those used for the Shugart data separators. There is a four bit shift register used to calculate the pre-compensation for each bit. The valid bits in this register must be safely written before the WriteEnable signal is turned off at the end of field. The SA4000 has a similar shift register in its pre-compensation circuitry. To allow its contents to be written, WriteEnable should not be disabled for at least 4 bit times after the last bit has left the controller.

The fields on the SA1000 drive each begin with a VFO synchronization pattern of all zeros followed by an address mark. This address mark is generated here. It differs from a normal MFM encoding of an NRZ data pattern in that two clock transitions are deleted. The encoding machine deletes these transitions whenever the proper NRZ pattern is present on its inputs and SyncWdFound is false.

The clock used to transmit the MFM data must be a stable reference clock, not the clock derived from the incoming MFM data. Data written using the derived clock would have an unacceptable amount of jitter. This reference clock is supplied by the Input Conditioning circuitry.

The cabling for the SA1000 consists of one 50 conductor ribbon cable and one 20 conductor ribbon cable leading directly from the controller card to the drive. The SA4000 cabling starts with the same two cables, but they plug into a connector on the back of the Dandelion chassis. From there, a single 50 conductor round-wire cable leads to the SA4000 drive in its own chassis. The wiring of the round-wire cable may be done in whatever manner is necessary to derive the SA4000 signals from those used or ignored by the SA1000.

Open-Collector Drivers

As stated above, the drive control lines such as HeadSelect.(1..16), DirectionIn, Step and so on must be

driven using 7406 or 7438 open collector drivers. The WriteEnable line is disabled whenever an overrun condition occurs. At most one incorrect word can be written if a disk operation is incorrectly started or interrupted.

Differential Drivers

Shugart specifies an open collector driver for the SA1000 and an active pull-up differential driver for the SA4000. Since the SA1000 and SA4000 must share cabling and board space and no extra differential pairs exist on the SA1000 cables, we use the active pull-up drivers for both drives. This causes no problems.

The Disk Output Clock for the SA1000 is the Timing Clock. It has a period 16 times that of the NRZ bit stream. It is used by the drive's stepping circuitry. When connected to an SA4000, The Disk Output Clock is the WriteClock. This clock has the same period as the bit stream and is used to sample it at the drive.

The SA4000 receives and sends NRZ data, so when connected to a SA4000, the Disk Output Data is the straight NRZOutput. The SA1000 requires pre-compensated MFM data, so in that case the output data is translated. The single-ended version of the Disk Output Data is re-received by the Input Conditioning circuits to test the board's data paths.

MFM Encoding, Pre-Compensation and Address Mark Generation

Only when the controller is attached to a SA1000 type drive is this section used. As the title implies, it has three main purposes.

The MFM encoding rules were explained in the Constraints section above. This section of the Output Conditioning circuitry encodes data according to these rules using a 5 bit shift register and a prom.

The prom is also used to calculate the amount of pre-compensation required. Data is stored on a disk in the form of transitions in the polarity of magnetic flux on the media. The read heads can sense these changes and return a series of pulses, one per change. After being stored, the flux transitions tend to physically migrate away from each other across the media. The degree to which they do this is one limit to the recording density. This tendency may be counteracted by grouping the pulses so they spread out to the correct locations. This means writing the pulses on the beginnings of bursts a little late, those on the ends of bursts a little early. In the case of the SA1000, "a little" is about 10 ns. This is what is meant by "pre-compensation." The prom produces each MFM transition along with a number representing the transition's pre-compensation interval. The pulse is then sent through a delay line and a multiplexer is used to pick the proper delay line tap for the output data.

In order to read the disk, the controller must derive a number of signals from the bit stream. First, a clock must be found and the bit stream decoded. This is done on the SA4000 drive with a built-in data separator. The encoding will only be done properly if the read heads are first enabled over a series of 0s on the disk. The SA1000 controller has its own data separator and uses address marks to determine the clock|data phase of the incoming data. After this, the controller must find the word boundries in the bit stream. The controller uses a sync word of all ones when connected to the SA4000 and the address mark for the SA1000 drive. Having found the field, the microcode must decide which field it has read. The SA4000 drive supplies sector marks and the microcode knows the first field after the sector mark is the Header field. The address marks for the SA1000 drive each contain a tag set only for the Label and Data fields. This tag may be checked by the microcode along with the error flags. It replaces the SectorFound flag.

The Address Mark writing process is quite similar to ordinary encoding. The microcode supplies a word whose MFM encoding is almost an address mark. The proper clock transitions are deleted to create the address mark when the this NRZ pattern is seen and SyncWdFound is false.

A stable reference clock must be used to write the MFM data. When the processor's 51 MHz clock is divided by 6, a 117 ns clock results. The nominal 1/2 bit period for the SA1000 disk is 115 ns. The 2% difference in period is easily accepted by the Shugart drive and allows us to dispense with a separate crystal for the reference. During SA1000 write operations, the NRZClock is obtained by dividing the MFM reference clock by two. Under this scheme, all controller clocks will be stable.

File: DDC05.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of DDC05.sil, the Input Conditioning section of the Dandelion Disk Controller

Overview

The Input Conditioning circuitry is responsible for buffered clock, drive status and data signals for the controller. These signals are based on the signals arriving from the disk, the operation being performed and the type of disk drive.

The SA1000 drive sends MFM data containing an encoded clock to the controller. The SA4000 drive sends NRZ data with a separate clock signal. The Input Conditioning circuitry either derives or passes the NRZ bit stream and generates an NRZClock signal from either the data, the disk clock or a stable reference. The method used depends on the SA1000/SA4000' signal and the operation being performed.

Constraints

In order to produce a stable stream of output MFM data, the Output Conditioning circuitry needs a stable reference clock. This is provided by dividing the 25.5 MHz HalfClk signal from the display controller by 3 to produce a clock with 117 ns period, RefMFMClock. Because of the use of this clock, the data rate on the SA1000 is 4.27 MHz, not the 4.34 MHz maximum specified by Shugart. During write operations a stable NRZClock is obtained by dividing the RefMFMClock by two.

Because of noise considerations, the single ended status lines from the disk drive must be received by Schmitt-Trigger buffer gates.

It is necessary to be able to tell the difference between an SA1000 type drive and an SA4000 type drive. As little extra hardware as possible should be employed for this purpose. Switches and jumpers are discouraged because of manufacturing, component and field service considerations. Our solution to this problem involves letting a spare wire on the SA1000 50 pin connector be grounded when connected to the SA4000. The line is normally pulled up.

The Clock and Data lines are differentially driven, hence must be differentially received. During write operations, the single ended version of the output data is re-received and used as input data. This is done to allow diagnostic microcode to test the controller's data paths. One could install an additional differential receiver and receive the differential data, but the added cost is not justified by the added coverage. One would need the new receiver and a multiplexer to test one driver.

The SA1000 drive must be supplied with a TimingClock at 1/16 the bit frequency or 1/32 the MFM clock frequency. The SA4000 drive needs a WriteClock that is locked in phase with the output bit stream. These two clocks are created here and the proper one is sent on.

A data separator is needed for four purposes. First, it must recognize the address marks that serve as field boundries for the SA1000. Second, it must derive the NRZ clock used to send the data. Third, it must derive the NRZ data stream from the MFM bit stream. Lastly, in recognizing the address mark at the beginning of each field, it marks the positions of the bit stream's word boundries.

The SA4000 has hard sectoring and provides the controller with sector marks. This feature is simulated on the SA1000 by writing and recognizing address marks and tags within address marks. The Input Conditioning circuitry must recognize the address mark tags and choose whether to send on the tag value or the drive's Sector signal.

SeekComplete Delay

The Seek Complete signal returned by the SA1000 drive is delayed until the heads have settled on the new cylinder. This is not true of the SA4000 drive. An extra delay must be added when the SA4000 drive is used. This delay is only active on the inactive to active edge of SeekComplete, allowing the microcode to sense the loss of SeekComplete immediately. It is implemented by using the drive's Seek Complete signal to clear a counter. The counter is incremented by Sector pulses from the drive after the drive's SeekComplete goes HI. When the counter reaches 29, it stops counting and the stop signal is used as the delayed SeekComplete. This delay could have been implemented in software or microcode. The time base used by the Mesa software is too coarse for efficient operation. A microcode routine would burn 1/5 of the processor for about 20 ms on each seek, a high price to pay. For these reasons, the delay has been put in hardware.

The SA4000 has an additional feature allowing one to turn off the holding current to the stepper motor by releasing DriveSelect. This may result in a considerable power savings. Upon re-activating DriveSelect, one must wait 20 ms for the stepper to re-acquire the cylinder and settle. The SeekComplete Delay facilitates this by allowing the absence of DriveSelect to clear the derived SeekComplete. Thus by waiting for SeekComplete, normally part of any operation, one may be sure the heads have settled no matter what their previous state.

Divide by 3 Clock Generator

The 25.5 MHz HalfClk signal from the processor is divided by 3 to produce a stable reference clock for the controller during write operations. It is also used to calibrate the Data Separator during seek operations and to generate the SA1000's TimingClock.

Schmitt-Trigger Buffers

Shugart specifies that the status signals from the disk should be received by Schmitt-Trigger buffers. This is done here. The RawSA1/SA4' signal is also received in this fashion in case the cable is noisy.

Differential Line Receivers

The Differential clock and data lines from the drive are received using differential receivers. When connected to the SA1000 drive, the clock receiver is not connected to a differential signal and is not used.

Input Select Multiplexer

As explained in the Constraints section, the method for generating a number of signals depends on the the type of drive attached. The Disk Output clock should be locked in frequency and phase with the data if connected to an SA4000 drive but should change at 1/16 the bit rate when connected to an SA1000. The NRZ Clock always bears the same relation to the NRZInput, but it must either be received from the SA4000 drive or derived from the SA1000 input data. The NRZInput is similarly either received or derived as is the SectorMk signal. The SeekComplete signal must either be delayed by 20 ms (SA4000) or passed on directly (SA1000). The Input Select Multiplexer is used to choose among these alternatives.

Wrap-Around Multiplexer

In order to test the controller's data paths, diagnostic microcode may read either the data that has been written (SA4000) or the Tag bit of the written address mark (SA1000). Only the driver and receiver chips are not tested in the SA4000 test, the circuitry between the data separator output and shift register input is not tested in the SA1000 test. The Wrap-Around multiplexer is used to receive the written MFM or NRZ data stream. This is either decoded, the address mark being detected, or shifted into the shift register. This tests not only the shift register and multiplexers but also the Data Separator in the Input Conditioning circuitry and the Encoder in the Output Conditioning section. To use the SA4000 version of the test, diagnostic microcode should begin a write operation, then read data instead of writing it. The shift register should contain a shifted version of the pattern being written. The shift accounts for the time taken to get through the output and input circuits. So long as the write buffer is not changed by the microcode, the same pattern will be continuously written. Hence after two words are written, the shift register will almost contain a *rotated* version of the output word. The MSB of the ReadData register will always contain the LSB of the pattern written as this is left over when the ReadData register is loaded.

One does not read the re-received data in the SA1000 version of the test. This is because jitter between the derived NRZ clock and the reference clock prevents reliable transmission of re-received data to the shift registers. The encoder and data separator are tested by having them generate and recognize an address mark and display the value of its tag bit. The tag bit is available in the SectorFound status bit as always and may be checked by microcode. Note the circuitry between the data separator output and shift register input is not tested. The controller may be able to pass the wrap-around test, yet not read or verify correctly. The only way to do a complete wrap-around would be to run the entire controller from one clock. It as deemed more important to test the data separator's ability to derive a clock and recognize data than to test this relatively small section of the circuit.

Data Separator

The Data Separator is only used when the controller is connected to the SA1000. drive. Two functions are incorporated into the it. At the beginning of a field, it recognizes MFM address marks used to set the clock|data phase of the incoming data and define the first word boundry in the bit stream. During data transfer, it derives the NRZ clock and data signals from the incoming MFM data. The Operation field determines which function is being carried out. During seek operations, the data separator maintains calibration using the RefMFMClock.

File: DDC06.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of DDC06.sil, Cables for the Dandelion Disk Controller

Overview

The Dandelion Disk Controller may be used to control either one SA1000 type drive or one SA4000 type drive. The SA1000 drive requires 2 cables, one 50 conductor and one 20 conductor. These ribbon cables are called HSIO50 and HSIO20 respectively. The SA4000 is mounted in a separate enclosure and requires one 50 conductor round-wire cable, called ExtDk50, for signals and one cable for power.

Constraints

It must be possible to directly connect the controller to the internally mounted SA1000 type drive using the two cables HSIO50 and HSIO20. No intermediate connectors are permissible. No wires may be re-arranged in the cables, if a signal connects to pin n on the controller, it must connect to pin n on the SA1000 drive.

The SA4000 drive and the Dandelion processor reside in separate metal enclosures. A single 50 pin round-wire cable, ExtDk50, is used for signalling between them. One end of this connector plugs directly into the SA4000 drive, the other into a connector in the rear of the Dandelion chassis. To connect the controller to a SA4000 type drive, HSIO50 and HSIO20 are connected to one side of this rear connector and ExtDk50 to the other side. The connector contains internal wiring to route the signals in the SA4000 cable to their counterparts in the two SA1000 cables.

The use of switches or jumpers to configure the controller board to the drive used is strongly discouraged. It should be possible to configure the controller to the drive by simply plugging HSIO50 and HSIO20 into either the SA1000 drive or into the rear chassis connector to ExtDk50. To this end, one of the pins is unused on the SA1000 drive and grounded when connected to the SA4000 drive (HSIO50.4). The pin is pulled up on the Controller card generating the signal RawSA1/SA4'.

HSIO50 and HSIO20

The pin assignments for these two connectors is given of the left side of the drawing. The extra signals needed by the SA4000 drive have been assigned to unused and spare pins on the SA1000 cables. These are marked "NA" and "SPARE" on the Shugart drawings respectively. These cables should be pin-for-pin compatible with the SA1000 cable specifications.

Signals whose meaning depends on the drive used have been given "neutral" names. For example, the signal called "MFM Write Data" on the SA1000 and "NRZ Write Data" on the SA4000 has been named "DiskWriteData" in this cable.

ExtDk50

The pin assignments for this cable are given on the right side of the drawing. In addition, the locations of the named signals in HSIO50 and HSIO20 are repeated here. This should be an aid in wiring the rear chassis connector. The HSIO50 and HSIO20 connections to the ExtDk50 ground returns have been made arbitrarily. Given signal X uses pin n on ExtDk50 and pin m on HSIOxx, the ground return on pin n + 1 of ExtDk50 is generally the ground return on m-1 of HSIOxx. For example, SeekComplete' is ExtDk50.22 and HSIO50.8. The ground return on ExtDk50.23 is the one on HSIO50.7. The ground return pin assignments

may be changed by the cable maker if desired.

File: HSIO47.bravo on [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of HSIO47.sil, the Control and WriteData registers in Processor Interface of Dandelion Disk Controller

Overview

The controller receives its control signals and output data through these circuits. The control information is divided into two groups, Drive Control and Operation. The write data is buffered in a single 16 bit register.

DiskControl Register

The register is loaded by the processor when a "KCtl ← xx" type instruction is executed in microcode. This may be done as part of a Mesa "Output" instruction. The command word is divided into two parts intended for the drive and the controller. The meaning of the bits in the Drive Control field are explained in the appropriate Shugart manuals. A quick list of Operation bit meanings is given below, the interested reader should consult the drawings where the bits are used for details.

In the Drive Control field, the DriveSelect bit has been included even though only one drive may be connected at a time. This is because releasing DriveSelect has useful side effects. The SA1000 type drives lack a FaultClear input, Write Faults are cleared by de-activating the DriveSelect signal. The SA4000 drive has a feature enabling it to cut the power to its stepper motors when not selected. This can result in a substantial power savings. The power may be cut by software when the drive has been idle for some nominal interval. When re-selected, one must wait 20 ms before using the drive. This time interval may be sensed using the SeekComplete signal which is automatically cleared when the drive is de-selected.

The FaultClear bit is only active when an SA4000 drive is connected to the controller. Write Faults on the SA1000 are cleared by turning off DriveSelect as explained above. An SA4000 WriteFault is cleared by activating both DriveSelect and FaultClear.

The FirmwareEnable bit is set whenever the disk microcode is running. In addition to acting as a status bit for higher level software, it is used to generate a service request for overhead operations.

BTransferEnable is set whenever a data transfer is taking place. The transfer operation encompasses the recognition and writing of the VFO synchronization pattern, sync word and the CRC checksum as well as transferring the data. When reset, all the state machines used to transfer or recognize data are reset. Because of the number of loads on this signal and the timing requirements, this signal is produced by a 74S74. If the 74LS273 were used, buffers would be needed. Timing is critical when stopping an operation. The combined propagation delay of the LS273 and buffers would prevent the controller from reliably stopping a transfer operation before status resulting the from the word after the CRC checksum had been posted. The critical signal is WordBoundry'. During read operations, it will rise at the 0000 to 0001 transition of BitCount, clocking in new status bits. The control word stopping the operation may arrive with about 60 ns to go before this transition. With the use of 74S74s to produce both BTransferEnable and WordBoundry', we may be sure of holding WordBoundry' LO.

The WriteCRC bit causes the CRC checksum to be written at the end of a field. The BTransferEnable and BWriteEnable lines must also be true for this to be accomplished.

WakeupControl.(0,1) are used to specify the condition generating the microcode service request. The conditions allowed are:

TransferEnable | WakeupControl.(0,1)	Condition
000	FirmwareEnable
001	SeekComplete

010	SectorFound (valid only on SA4000)
011	IndexFound
100	Word Ready from Read operation
101	Word Needed for Write or Verify operation
110	<no wakeup>
111	<no wakeup>

The WriteEnable bit controls the write amplifier on the drive. In addition, it is used by the controller to decide when a write operation is taking place. The WriteGate to the drive is enabled only when WriteEnable and TransferEnable are true and Overrun is false. A buffered version of WriteEnable called BWriteEnable is used on the controller.

WriteData Register

Data to be written or verified is stored in this register. When needed, it is loaded into the SerialNRZ shift register. One 16 bit buffer is all that is needed even at the SA4000 data rate. Because of the Dandelion architecture, we can guarantee it will always be serviced before the next word is needed.

File: HSIO48.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of HSIO48.sil, the Status and Test multiplexer and ReadData register in Processor Interface of Dandelion Disk Controller

Overview

This drawing shows all circuits used to send data from the controller to the processor via the X-Bus. The Status and Test drivers send either the status of the disk and controller in response to ←KStatus' or diagnostic test data in response to ←KTest'. Data read from the disk is buffered in the ReadData register and read when ←KIData' is LO.

Constraints

It should be possible to read all signals being sent to this disk. This is done so a diagnostic program may decide whether a disk problem lies in the drive or in the controller. Since extra bits are available after reading all signals sent, the un-buffered versions of signals received from the drive are generally available.

Independent drivers are used for the KStatus and KTest ports to reduce the time needed to release the X-bus. A design with less X-bus loading would use a 3-state multiplexer, such as the 74S257. However, a gate would be needed to enable the outputs when either ←KStatus' or ←KTest' went LO. The extra propagation time of this gate and the multiplexer would cause X-bus data to be held into the next cycle.

The display controller has no status port, yet it should be possible to read the control signals being sent to the drive. The Horizontal Sync (BHoriz), Vertical Sync (BVert') and both polarities of the Video signal are available on the Test port. Diagnostic microcode can be used to sample these signals.

The clock edge used to update WordBoundry' is the same one used to change SerialNRZ.(0..15). Because of this, sampling SerialNRZ.(0..15) with WordBoundry' is hazardous. A buffer is needed before the ReadData register to ensure the data sampled is stable when WordBoundry' rises. This is the ReadData Buffer Register. WordBoundry' is discussed more fully in HSIO51.bravo

File: HSIO49.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: A description of the Service Request and Overrun machine as well as the Word Status buffer in the Processor Interface section of the Dandelion Disk Controller.

Overview

This is a relatively complex drawing. The top two thirds shows a circuit that generates the Service Request (KReq') and the Overrun signal flag. The two registers at the bottom of the drawing sample status flags at WordBoundry' intervals. A tranfer operation is stopped by writing the proper control word. This word will be written at some time during the first word following the CRC checksum. Since the status flags are sampled only at WordBoundry intervals, the microcoder may be sure data bits following the checksum do not effect the status.

Constraints

An I/O microcode task in the Dandelion processor runs only on request from its associated controller. Service requests are used not only to let the microcode execute set-up and overhead tasks but also as a means of sensing conditions without busy waiting. Clicks not used by I/O tasks may be used by the Emulator task.

It should be possible for the Emulator to awaken I/O microcode. This is done by having the Emulator write a control word to the I/O hardware which will cause a service request. This control word should not only specify a condition but assure the condition is true or will become true. In the disk controller design, the wakeup condition is set to "FirmwareEnable = 1" and that bit is turned on.

Several conditions should be sensed without busy waiting, a practice which wastes control store space and time needed by the Emulator. Conditions which occur relatively infrequently may be used to generate Service Requests directly. Such signals as SeekComplete, SectorFound and IndexFound are prime candidates.

The microcode could run during all of its clicks during a data transfer, sensing if it was needed each time, but this would require more complex code and another hardware status bit. For the SA1000, it would also waste about half of its clicks. Thus a service request is generated when a word should be transferred.

To calculate when a data service request should be sent, we examine Roy Ogus's memo [Iris]<Workstation>notes>IOLatencies.notes. The relevant formulas are:

Write, Verify Operations: $3Tr(c + 2) - a < p < r(6T - t_{Setup})$

Read Operation: $Tr(3c + 5) - a < p < r(5T - t_{Setup})$.

Where:

T = processor cycle time
= .137 microseconds

r = disk bit frequency
= 4.27 MHz for SA1000
= 7.1 MHz for SA4000

c = max number of clicks from end of one service click until end of the next
= 5 for the disk task

a = number of bits transferred at a time
= 16 bits/word

p = time at which service request should be sent in bit times before buffer is used.

t_{Setup} = setup time for memory data register
= 0.05 microseconds

For the SA1000,
Write, Verify Operations: $-3.71 < p < 3.30$
Read Operation: $-4.30 < p < 2.71$.

For the SA4000,
Write, Verify Operations: $4.54 < p < 5.51$
Read Operation: $3.56 < p < 4.53$

For example, a service request must be sent some time between 3.3 bit times before and 3.71 bit times after the SerialNRZ buffer is loaded during a write or verify operation when connected to an SA1000. Thus the request may be sent at any bit boundry from 3 bits before to 3 bits after the load. Similar interpretations apply to the other conditions. Note the SA4000 conditions are very strict. The Write and Verify operations require a pre-request of exactly 5 bit times, the Read operation exactly 4 bit times. One should also note that control words are actually written at the *end* of their cycles. One cycle is equal to 15/16 of an SA4000 bit time, so one should realize a control word will be written approxmately one bit time later than the second number in the inequalities indicates.

Another way of looking at these numbers is to calculate when the service resulting from a given service request may arrive. Say we take an SA4000 read operation as an example. The Serializer/DeSerializer maintains a count in BitCount that ranges from 0 to 15. This count gives the number of bits in SerialNRZ that have been *shifted* in but have not yet been read out. On the NRZClock transition loading the ReadData Buffer, the first bit of the next word is shifted into SerialNRZ.15 so BitCount goes to 1. Thus the buffer is loaded on the 0000 (= 16 MOD 16) to 0001 BitCount transition. As seen from the last equation above, the service request must be sent to its synchronizer 4 bit times before this, when BitCount ← 13. The earliest the processor, which is not synchronous with the controller, may read the buffer is at time 13 + 4.53 MOD 16 or time 1.53. This is only one half of a bit time after the buffer was loaded! The earliest time a control word (like the one that turns off the operation) will be written is about at time 1.5 + 1 or 2.5. The latest time at which the processor will service the buffer is at time (13 + 16 + 3.56) MOD 16 or time 0.56. This is only one half of a bit time before the buffer will be loaded again! The calculations of the latest service assume that it will take place at the end of the cycle, so the latest a control word may arrive is also at time 0.56. Obviously, these requirements are quite strict.

Since events in the controller and processor are asynchronous, the service request from the processor must be synchronized. This is done by sending it through two flip-flops in series clocked by the processor clock. Even if the first flip-flop enters a meta-stable state, the chance that it will remain in one for one processor clock time is less than the probability that the chip will fail completely. Thus a chain of two flip-flops is sufficient for synchronization at this clock frequency.

The CRCError flag is only valid on the clock transition immediately following the last bit of the CRC checksum. The WordBoundry' signal must rise on this transition. The same consideration applies for the VerifyError signal. Luckily the ReadData buffer should also be loaded on the clock transition immediately following the last bit of each word, so the same signal may be used in all applications. Note the RawCRCError signal is produced by a very slow chip, the Fairchild 9401, and the WordBoundry' signal by a relatively fast one, the 74S74. The 9401 is clocked by NRZClock', which is one gate delay (74S04) behind NRZClock which clocks the 74S74. Thus we feel safe that RawCRCError will not change by the time WordBoundry' rises.

The SWriteCRC signal controls combinational gating that allows the output data stream to be fed by the CRC chip's internal register. Each data or checksum bit is made available for transmission on the rising edge of NRZClock and transmitted during that clock. SWriteCRC should go true on the first NRZClock transition following the last data word of a field. This is also when WordBoundry' rises, so SWriteCRC may be clocked by WordBoundry' with the status signals and the ReadData register.

Detecting Processor Service

The processor may reply to a data service request by reading the ReadData buffer (←KIData'), writing the WriteData register (KOData←') or writing the control register (KCtl←'). The last choice is included so that the WriteCRC command may be sent instead of a data word without causing an overrun condition. The three signals used are produced by a decoder on the CP board so must by synchronized using preWaitClk' here.

The processor clock is not synchronized with the controller clock. The processor has a 137 ns clock, the controller either a 140 ns or 234 ns clock depending on the disk being controlled. Because of the asynchrony and its longer period, the controller cannot depend on sampling to sense the processor's I/O signals.

The method used to trap the processor's service signal forces the signal to toggle a flip-flop output, called ServiceTrap. This is done near the top of the page. The toggle method makes it possible to sense the service signal without wasting prom outputs or time resetting a set-reset type indicator. The trapped signal is synchronized using two register elements in series. The actual prom input is generated by detecting the fact that the transition has reached a third register element using the exclusive-or gate.

In order to correctly calculate the Overrun condition, we must calculate the longest time a signal can take in getting though this chain. Say the service completes just before the end of period m. Because of the delay in the gates and the flip-flop, the transition may not enter the synchronizer chain until the beginning of period m + 2. It may not exit until time m + 3. Thus the Overrun machine should wait until 3 bit times after the last legal service interval before declaring an overrun. During a read operation, the ReadData buffer is loaded at time 1, so the check is made when BitCount = 4. During write or verify operations, the WriteData buffer is read at time 0, so the overrun check is made when BitCount = 3.

Generating KReq' and Overrun

A number of signals are needed to generate these two flags. As shown in the constraint section, the machine must know the operation being performed, the time within the field and the BitCount. To generate the Overrun flag, it also needs the Serviced flag and the number of requests pending in ReqState.(0,1).

BitCount cycles through the numbers 0, 1, ... ,15 as explained above and in HSIO51.bravo. SyncWdFound goes true on the NRZClock edge that ushers in the first data bit of the field (this is immediately after the sync word). WakeupControl.1 and BWriteEnable specify the operation in progress. A read operation is being done when both are zero. When WakeupControl.1 is set and BWriteEnable reset, a verify is being done. A write is in progress when both are set.

While ReqState.(0,1) is less than or equal to 2, it specifies the number of service requests pending. Two requests may be queued when a pre-request is sent during an existing request. When equal to 3, ReqState.(0,1) indicates an overrun condition.

For a detailed explanation of the conditions used to set ReqState and Overrun, read the amply commented code used to create the Prom shown in the drawing. This is filed as SrvcReqProm.mesa in [Iris]<Workstation>HSIO>Proms>DDCProms-Rev-A.dm.

The general idea used goes as follows: Service Requests are sent at fixed intervals and serviced at random times within the following word. A service request for an SA4000 write or verify operation is sent whenever BitCount←11, regardless of the state of SyncWdFound. Note a pre-request must be sent before a verify operation so the second word to be verified will arrive while the first is being checked. A Service request for an SA4000 Read Operation is sent after SyncWdFound is true and BitCount←13. Service requests during SA1000 Read, Write and Verify operations are all sent when BitCount←0.

Calculating the next value for ReqState is done in two steps. First a check for an overrun condition is done. If this conditions does not hold, a check is made to see if the number of pending requests should be increment or decremented.

The overrun conditions occur if a) one already exists (it is cleared by stopping the transfer), or b) two requests are pending (ReqState = 2), the relevant buffer is needed and it has not been serviced. Actually, this is determined 3 bit times after the buffer is used because of the delay involved in creating the Serviced flag. As discussed in the previous section, the check is made when BitCount = 4 during a read and when BitCount = 3 during a write or verify operation. This applies to both SA1000 and SA4000 operations.

If it is either not time to check for overrun, fewer than two requests are pending or the buffer was serviced at the last second, we check to see if the number of requests pending should be increased or decreased. If no requests are pending, the Serviced line is ignored. If it is time to make a request (BitCount = 12 for an SA4000 read or 10 for an SA4000 write or verify or 15 for any SA1000 operation), the ReqState is incremented. If a previous request was serviced (Serviced = true) and 0<ReqState<3, ReqState is decremented. If both conditions hold true, they cancel out as the new request replaces the old.

Note the conditions used to generate requests are different for the SA1000 and SA4000 type drives. The Service Request prom uses the SA1000/SA4000' signal to generate the proper request at the proper time.

At the beginning of a verify operation, a pre-request must be sent before the first word is checked. This allows the second word to arrive in time. When using an SA4000 disk, this is done by sending a request whenever BitCount = 11, even when the synchronization word is being recognized. When connected to an SA1000 drive, BitCount skips 11, going directly to 15. Luckily, this is when the SA1000 sends all requests anyway.

KReq' Multiplexer and Synchronization

The 8:1 multiplexer in the middle of the page is used to select the condition producing the service request. It in turn is controlled by the BTransferEnable and WakeupControl lines. With these, the microcoder may select the condition used to cause a service request. These have been discussed in the Constraints section.

The service request is synchronized to the processor clock using two registers in series. While the first may enter a meta-stable state, it will recover by the time the processor clock strobes the second register. Hence, the request sent to the processor will be stable.

When debugging new microcode, the multiplexer is an excellent chip to watch, especially if the disk stops unexpectedly. The process of verifying that a service request is being sent or determining why it is not being sent usually begins here.

Status Word Buffer

Some of the status and control signals must only change on word boundries. This was discussed in the Constraints section above. The register in which the signals are strobed is shown at the bottom of the page. Some of these signals change on the rising edge of NRZClock, as does WordBoundry'. It would be hazardous to sample the former with the latter. The buffer register inserted before the Status Word Buffer samples the signals on the falling edge of NRZClock when they are stable. This is in turn sampled on the rising edge of WordBoundry' (which is in the rising edge if NRZClock) when it is stable. The status flags are cleared under processor control by executing the ClrKFlags command. Note the ClrKFlags' signal is produced by a decoder on the processor so must be synchronized before being used.

Note also the WriteEnable signal is delayed 5 bit times here. This is to ensure it does not turn off while bits are still in the SA1000 or SA4000 pre-compensation circuits. There are shift registers in these circuits used to encode the NRZ data and calculate the proper pre-compensation intervals. If the processor responds to the last service request of a write operation very quickly, it can disable WriteEnable before the CRC checksum is entirely written. Having the microcode wait an extra word would solve the problem were it not for the fact that the extra overhead causes a sector to be too long. This has been determined empirically. With the extra overhead, the microcode takes longer to process a sector than the disk head takes to pass between two sector marks. Thus the WriteEnable is held up in hardware until the last bits have been written out.

File: HSIO50.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of SerialNRZ shift register, CRC Checker/Generator and Verify checker in Serializer/DeSerializer portion of Dandelion Disk Controller

Overview

Some of the more interesting data paths in the controller are on this drawing. The shift register used to do the serial to parallel and parallel to serial conversion is shown as the top two chips on the page. The CRC Checker/Generator and associated circuitry are below it and the Verify checker is at the bottom of the drawing.

Constraints

One of the main functions of the controller is to convert data from the discrete 16 bit words used by the processor to the continuous serial data stream used by the disk drive. A single shift register may be used for both tasks as only one operation is performed at a time.

It must be possible to both read words from and write words to the shift register. In order to avoid the control circuitry associated with a tri-state bus, 74199s were chosen as the shift register chips. These have separate pins for their parallel inputs and outputs.

Each sector is composed of three fields, Header, Label and Data. A 16 bit CRC checksum must be generated and checked for each field. The first checksum bit must be available as soon as the last data bit of the field has been written. The results of the CRC checker must be available as soon as the checksum has been read in. This could all be done in microcode for the read and write operations. The checksum would be a simple exclusive-or of all the words. Unfortunately, the verify operation would require that data be both read from the disk to be checked and written to the disk to be verified. The SA4000 data is only guaranteed to be stable during one cycle, so it may not be both read and written. For this reason, the CRC is done in hardware.

The CRC Generator must use the shift register output data to compute the checksum in order for the first bit of checksum to be ready directly after the last bit of data. The CRC Checker must use the shift register input data so its error flag will be ready as soon as the CRC checksum has been read in. Were we to relax this constraint and allow an extra word of waiting time in each field of each sector, more time would be needed to process a sector than exists between sector pulses on the SA4000. For these reasons, a multiplexer is needed on the CRC Generator/Checker input.

The CRC checksum must be appended to the data stream. This requires a multiplexer in the output data stream.

The CRC Checker/Generator must be enabled at exactly the same time relative to each field being read or written. Obviously, the checksum calculated depends on the input bits, hence on the time during the field at which the CRC chip is enabled. It is necessary to check every bit in the data portion of the field, the synchronization word may be ignored. For these reasons, the CRC chip is enabled when SyncWdFound goes true. This is on the NRZClock edge that presents the first data bit to the chip.

Like the CRC chip, the Verify checker should be enabled when the first data bit appears. Thus SyncWdFound is used here too. Since the verify checker compares data bits at the shift register input with memory template bits at the shift register output, the first bit of the memory template must also be present in SerialNRZ.0. This is done by having the microcode pre-load the first data word into the WriteData register and controlling the shift register parallel load input. This signal, PLdSerialNRZ', is held LO until SyncWdFound rises. Thus while the synchronization word is being recognized, every NRZClock transition loads the SerialNRZ register with the first memory template word. Note that the WriteData register and

SerialNRZ shift register contents are cleared by the absence of BTransferEnable. Thus the first memory template word must be sent after the control word starting the verify operation but before the sync word is found. Thus it is necessary that both the control word and the first data word be sent in the same click since the control word will cause the service request to be turned off until the sync word has been found.

SerialNRZ Buffer

This is a shift register as described above. It may be both loaded and read in as a single word. It is cleared between transfers so that the first word of a write will always be 0000...00. That is the VFO synchronization pattern.

WriteData Multiplexer

The CRC checksum must be appended to the data field. This is done using this multiplexer. One half of a dual 4:1 multiplexer was used so the other half could serve as an independent 2:1 multiplexer. This other half is now used to supply a reference clock to the controller's Phase Locked Loop during seek operations.

CRC Generator/Checker

As explained above, a CRC chip both generates and verifies a checksum over the data portion of each field. A multiplexer is needed so the CRC chip may receive its input from the shift register input during read and verify operations and from the shift register output during write operations.

The Generator/Checker's internal shift register is held equal to 111...11 (preset) until the first bit of the data portion of the field being written or checked is ready. Note the checksum of a field of zeros is not zero. This fact allows us to detect an error in a field of zeros which has been cut short. This is a fairly common pattern.

The RawCRCError signal is sampled in the Processor Interface machine. It is valid on the first clock edge after the last checksum bit has been sampled.

Verify Checker

The controller must be able to compare data on the disk with that in memory. This is done by sending the memory data to the controller as the corresponding data is read from disk. These bit streams are aligned so that memory data arrives at the shift register output at the same time the disk data arrives at the input. The corresponding memory and disk bits are compared using the exclusive-or gate shown, the results being recorded in the D flip-flop.

Once this flip-flop is set, the Q' output holds it set. When resetting this flip-flop, the Q' output is immediately raised, relaxing the set input and allowing the Q output to be reset. The BitVerifyErr output is sampled by the Processor Interface on WordBoundry' intervals. The actual VerifyError signal sent in the status word is delayed by one word time. This is done because it should become valid with CRCError. The latter signal is not valid until the CRC checksum has been read, or one word time after the last data word.

File: HSIO51.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of Field and Word State Machine in Serializer/DeSerializer portion of Dandelion Disk Controller

Overview

In order to generate Service Requests to the processor, enable the error checkers and detect the word boundries in the serial data stream, it is necessary to know the current position in both the field and the word being transferred. These quantities are calculated in the circuits shown here.

BitCount.(0..3) records the position in the SerialNRZ shift register of the word being transferred.

SyncWdFound goes true on the first clock tick after the synchronization word or address mark has been received. This enables the error checkers and forces the state machine to switch from synchronization mode to transfer mode.

The PLdSerialNRZ' signal is used to parallel load the SerialNRZ shift register.

WordBoundry' is used to load the ReadData buffer with a word from the disk as well as to sample status and control flags.

Constraints

The process of determining the word boundries in a bit stream has two parts. One must find the first word boundry, then count bits from that point. The BitCount.(0..3) outputs of this machine may be used for both purposes. When searching for a field boundry on the SA4000 drive, they hold the number of valid synchronization word bits found. The end of the synchronization word marks the first word boundry. After this, BitCount.(0..3) is incremented modulo 16 marking subsequent word boundries.

In order to tell the difference between these two modes of updating BitCount, another signal is needed. This is SyncWdFound. When reset, BitCount is used to synchronize. When set, BitCount is always incremented. Since it rises at the beginning of the data field, SyncWdFound also serves to enable the CRC and Verify error checkers.

The field boundries are indicated by address marks, not synchronization words on the SA1000 drive. This is because the SA1000 drive returns no sector pulses from which one could find fields. The address marks contain illegal, but recognizable, MFM patterns so they will not be mistaken for ordinary data. The address marks are recognized by the Input Conditioning circuits. The AddrMkFound signal is set at the beginning of bit 14 of an address mark. The first rising edge of NRZClock occuring while AddrMkFound is true supplies bit 15 to the Field and Word Status Machine. This forces the Word state machine to set BitCount to 15. The value of BitCount automatically increases to 0 on the next NRZClock and SyncWdFound is set, just as if the SA4000 synchronization word had been seen.

Note that two synchronization operations are happening in parallel when the SA1000 is connected. The address mark machine is looking for an address mark while the Word and Field state machine is looking for a sync word. We must be able to guarantee that a sync word is not seen until the address mark has been found. This is complicated by the fact that the head will be traveling over arbirtary data fields looking for an address mark. This problem is simply solved by connecting the derived NRZClock to AdrMkCnt.4 and the NRZInput to AdrMkCnt.2. So long as nothing looking like an address mark goes by, AdrMkCnt.(0..4) oscillates around 0, 1, 2 and 3, so AdrMkCnt.2 is zero. When an address mark is being recognized, AdrMkCnt.2 spends no more than 4 consecutive clock ticks at 1, so cannot look like a 16 bit synchronization word. Thus BitCount.(0..3) will stay less than 4 until AddrMkFound arrives.

During write operations, the microcode supplies the VFO synchronization pattern (several words of zeros), the synchronization word or NRZ version of the address mark, the data and a command used to write the CRC checksum computed by the CRC chip for each field written. BitCount must be incremented modulo 16 even during the synchronization pattern so Service Requests will be generated properly. SyncWdFound, however, should not be set until the synchronization word or address mark is sent out. If it were set early, the CRC Generator would not be properly initialized. This is the reason both the synchroniztion word and address mark have a 1s in bit 15. This lets us set SyncWdFound as soon as BitCount = 15 and CRCInput = 1.

Early versions of the controller would not set SyncWdFound or advance BitCount beyond 15 until a "1" was seen during Read or Verify operations. When connected to an SA1000, a service request is sent whenever BitCount = 15. If a string of data errors caused by the PLL losing phase synchronization with the incoming pulse stream occured at this time, multiple service requests would be sent. This could result in very strange data errors, sometimes in an overrun. In order to force the error indicators to more closely reflect the error cause, the potential delay at BitCount = 15 was eliminated. That is, if BitCount = 15 and the operation is Read or Verify, BitCount always gets 0 and SyncWdFound is always set on the next rising NRZClock edge. Now any data error will correctly be caught by the Verify or CRC checker.

While SyncWdFound is false, the number in BitCount.(0..3) is defined to be the number of sync word bits recognized while finding the first word boundry. While transferring data it is defined to the number of bits shifted into but not yet read from the shift register modulo 16. When the register is full during read operations, the data is read and a new bit is shifted in on the same clock pulse. Thus the number of unread bits ranges from 1 through 16 and BitCount ranges from 1 to 2 to 3 on up to 15 then 0 (0 = 16 MOD 16). The word boundry during read operations is on the 0000 to 0001 transition.

When the shift register is parallel loaded during write or verify operations, BitCount←0000. This is because none of the resulting bits in the register were *shifted* in, all were parallel loaded. The buffer is shifted and bits are written or verified while re-received or verified data is shifted in. BitCount eventually gets up to 1111. On the next clock pulse, the cycle starts over, the first bit of the new word is loaded to replace the last bit of the old word in SerialNRZ.0. During write and verify operations, the word boundry is on the 1111 to 0000 transition.

The SerialNRZ register is loaded when the NRZClock rises and PLdSerialNRZ' is LO. As seen above, this should happen on the 1111 to 0000 transition of BitCount. Hence PLdSerialNRZ' is lowered if the operation is write or verify and BitCount = 1111.

There is no time during a verify operation to request the memory version of the first data word to be verified after recognizing the synchronization word. The first word must be pre-loaded in the SerialNRZ register. To both load it and prevent it from being shifted out while synchronizing, PLdSerialNRZ' is also active while the operation is verify and SyncWdFound is false. The first word is repeatedly loaded while the sync word is being found. Since both the WriteData register and the SerialNRZ shift register are cleared so long as BTransferEnable is LO, the first data word must be sent after the control word that begins the verify operation. This control word causes the hardware to turn off the Service Request until it needs the second word of the field. Hence, the first data word must be sent in the same click as the initial control word.

WordBoundry' is used during all three types of operations. It is used to clock the ReadData register and to clock the Word Status buffer. The former holds data read from the disk, the latter status and control lines that should only change on word boundries. WordBoundry' will always serve its purpose if its trailing, or rising, edge rises on the current boundry between words. During read operations, it is LO while BitCount = 0000 since the next clock edge is a word boundry. During verify or write operations, it is LO while BitCount = 1111 for the same reason. Note that WordBoundry' is produced by its own 74S74. This is a simple solution to a complex problem. Both the status signals, like BitVerifyErr, and WordBoundry' change in response to NRZClock, hence it is hazardous to sample one with the other. For this reason, the Word Status Buffer Register samples the status signals for the Word Status register on drawing HSIO49.sil. The WordStatus Buffer Register samples the signals on the falling edge if NRZClock so they will be stable by the time WordBoundry' changes. Unfortunately, the RawCRCError signal is produced by a very slow chip, the F9401. The error flag may not be stable by the time NRZClock falls (~70 ns after it rises). However, this very slowness allows us to guarantee it will not change within 10 ns after NRZClock rises. If

WordBoundry' can be made to react to NRZClock very quickly, RawCRCError may be sampled directly. This is what has been done. Note RawCRCError goes directly to the Word Status Register on HSIO49.sil and WordBoundry' is produced by a fast 74S74.

When not transmitting data, the hardware state variables should all be reset. This is done by resetting the prom output register when BTransferEnable is LO. This ensures that on initialization of a read or verify operation, no bits of the synchronization word have been recognized. It also says that when beginning a write operation, BitCount = 0000 so the SerialNRZ register is full of data. This happens to be true since the first word written is always 000...00 and the shift regsiter contents are cleared by the absence of BTransferEnable.

Field and Word State Machine

The interesting points about this machine were described in the Constraints section above. It is implemented using two 1024x4 proms and a register. I have standardized on 1024x4 proms because of the extra costs involved stocking multiple types of proms and because 512x8 proms are extremely difficult to obtain.

A listing of the rules used to generate the prom contents may be found in FldWdProm.mesa in [Iris]<Workstation>HSIO>Proms>DDCProms-Rev-A.dm.

File: HSIO52.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of the MFM Encoding, Pre-Compensation and Address Mark Generation circuitry of the Output Conditioning section of the Dandelion Disk Controller.

Overview

This section of the controller is used only when writing data to the SA1000 disk. It contains circuits for encoding the NRZ data stream into MFM, calculating the pre-compensation interval and generating address marks.

The prom machine at the top of the page generates the MFM data to be written. The delay line and multiplexer below implement the pre-compensation delay.

Constraints

The SA1000 drive does not contain a data separator, it is included in the controller. One section of a data separator encodes MFM data, another decodes it. The decoding section is in the Input Conditioning circuits of the controller in our implementation. The encoding section is shown here. Along with encoding the data, a data separator must calculate a pre-compensation interval.

Information is carried on the disk medium as a series of changes in the polarity of magnetic flux. These transitions are written by changing the polarization of the write heads. The transitions may be sensed as they drive a current in the read heads. Flux transitions written on the medium tend to migrate away from each other, limiting the bit density. This tendency may be counteracted by grouping closely spaced transitions even more closely. When these transitions spread out, they will be aligned properly. To do this, transitions on the beginning of a group are written a little late, those on the end of group a little early. In our case, "a little" is 10 ns. This process is called "pre-compensation."

There is no separate clock recorded with the disk data, the data must be encoded so a clock may be recovered from it. A simple scheme would be to record at least one flux transition with each bit to act as the clock. The recording method must also allow a high bit density, i.e. as few flux changes per bit as possible. The MFM (modified frequency modulation) rule gives a reasonable compromise. Under MFM encoding, each NRZ bit is broken into two halves, call them clock and data. A flux transition may be sent in either half. Say we represent a flux transition by "1" in this discussion. An NRZ 1 is always encoded as clock|data = 01, i.e., it has a flux transition in the data half of the bit. An NRZ 0 is encoded as clock|data = 10 unless the previous bit was an NRZ 1. In that case, the 0 is encoded as clock|data = 00. The point of the exception is to eliminate transitions in consecutive bit halves (if there is a flux transition in the data half of a bit, there is no transition in the adjoining clock halves and vice-versa). This rule results in a maximum of one flux transition per bit (for a string of all zeros or all ones) and a minimum of one every two bits (for a string of alternating ones and zeros).

The data must be written on the disk using a stable reference clock. This gives the greatest margin of error for motor rate changes and other variances. In order to test the data separator in the Input Conditioning circuits, the separator must derive a clock from the re-received data. If this clock were also used to send the data, it would be subject to drift, so it may not be used. While performing a write operation, MFMClock and NRZClock are derived from the stable processor clock. When the 51 MHz display bit clock is divided by 6, a 117 ns clock is obtained. This is within 2% of the nominal 115 ns double frequency clock used by the disk, so the 117 ns clock is used. Besides not having to install an extra crystal and stable oscillator, this scheme has another benefit. Transitions are not packed quite so densely on the disk so the error rate should be lowered.

An accurate method must be found for delaying the output data 10 or 20 nanoseconds to implement pre-compensation. This is done here with a tapped delay line and a multiplexer. It is assumed the propagation delays are the same for all the multiplexer's data inputs. Note pre-compensation should only be enabled on

the more densely packed inner tracks. This is where ReduceIW is set, so it is used as an enabling signal.

Encoder, Pre-Compensation and Address Mark Prom

This prom has several functions. The data being sent to the disk must be encoded using the NRZClock, the present and the previous data bits. The pre-compensation interval must be calculated using the present bit, the previous two bits and the next bit. Finally, the pattern existing when the address mark is written must be recognized. This can be done using SyncWdFound, the present bit and the previous three bits.

Note the inputs to the encoder prom must be synchronized with the RefMFMClock. This is because they would arrive too late in the prom cycle to be used if sampled directly.

All these functions may be done using the bit to be encoded, the previous three bits and the next bit. It would be possible to use a 5 bit shift register as the encoder input. Instead, the prom acts as the shift register while using the bits as inputs. The data is shifted in as BNRZWriteData to NRZpWrData.4. It is shifted up through NRZpWrData.0, one shift on each HI to LO transition of NRZClock. One might think LO to HI would be better, but the synchronizing register should allow a full NRZ bit time for its inputs to settle (the CRC chip is slow) so its useful samples occur on the rising edge if NRZClock. Thus the prom machine sees new data from the synchronizing register when NRZClock goes LO.

For a detailed listing of the prom contents, see the Mesa program used to generate the prom. This is stored as EncodingProm.mesa in [Iris]<Workstation>HSIO>Proms>DDCProms-Rev-A.dm. Since 256x4 proms are much easier to obtain than 512x8 proms, this machine has been constructed from the two 256x4 proms, not one 512x8.

The UnCompMFM is produced by examining NRZClock, NRZpWrData.3 and NZpWrData.2. The bit in NRZpWrData.3 is the one being encoded. Since new data appears when NRZClock goes LO, this signals the time to produce a clock as opposed to a data transition. When NRZpWrData.3 is a 1, UnCompMFM is sent as clock|data = 01. When both NRZpWrData.3 and NRZpWrData.2 are 0, UnCompMFM is sent as clock|data = 10. When NRZpWrData.3 = 0 but NRZpWrData.2 (the previous bit) equals 1, UnCompMFM is sent as clock|data = 00. This is the standard MFM encoding rule. There is an exception to this rule used to write address marks. This is explained below.

An address mark may be encoded as long as SyncWdFound is reset. During read and verify operations, the write data is ignored anyway, so the encoding does no harm. The NRZ form of an address mark is 10100*0*01 010*0*0T1. In the MFM version, the zeros in bits 5 and 12 (*italicized*) are written without their clock transitions. It is hoped that this duplication of illegal conditions will prevent normal glitches at the ends of written fields from looking like address marks. The "T" in bit 14 represents a tag bit used to distinguish Label and Data field address marks from Header field address marks. It is set in the Label and Data field marks. Only two types of words are written while SyncWdFound is reset. One is the all zero word used to synchronize the phase-locked loop section of the data separator. The other is the address mark. The bit on which the clock transition is to be omitted is recognized by seeing NRZpWrData.0 is a 1 while NRZpWrData.(1..3) are all zero. Note Bit 15 of the address mark word is on the encoding and Field state machine inputs at the same time. This is during the time Bit 13 is being encoded, so Bit 12 will have already had the address mark written properly. On the next NRZClock edge, SyncWdFound will go HI turning off the address mark encoding.

The pre-compensation intervals are derived from the Shugart controllers. A bit is either sent on time (PreComp = 01), 10 ns early (PreComp = 00) or 10 ns late (PreComp = 10). The interval encoding is given below. The bit in NRZpWrData.3 is being encoded.

NRZpWrData.(1,2,3,4)	PreComp.(0,1)	
0001, 0110, 1110	00	(minimum delay)
0000, 0010, 0100, 0101, 0111, 1001,	01	(normal delay)

1010, 1100, 1101,
1111

0011, 1011, 1000, 10 (maximum delay)

Pre-Compensation Delay and Multiplexer

As explained above, the pre-compensation delay is actually implemented using a delay line and a multiplexer. The pulse is fed into the delay line as the proper tap is chosen. By the time the pulse reaches the first tap used, the multiplexer is set up. Note only the leading edge of the pulse is used by the drive. Also note there is no danger of running pulses together since the MFM encoding rules ensure there are no consecutive pulses.

File: HSIO53.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of Output Buffers in Output Conditioning section of Dandelion Disk Controller.

Overview

This is a fairly simple drawing. Shown are the differential and single-ended drivers used to buffer the disk data and control signals.

Constraints

Shugart specifies that differential drivers must be used for the data lines. The SA1000 disk specifies open-collector differential drivers, the SA4000 specifies active pull-up drivers. Since it is necessary to use the same physical chip for both disks to eliminate the need for jumpers, the active pull-up 75114s are used. The fact that the SA1000 drive uses resistor pull-up while the 75114 is using active pull-up on the other end of the wire is of no consequence.

Either NRZ or MFM data may be sent on the data lines. A multiplexer controlled by the SA1000/SA4000' signal is used to choose between them. It is assumed that this will not affect the MFM pre-Compensation. That is, if a pulse reaches the multiplexer 10 ns late, it will leave the multiplexer 10 ns late.

The specifications for the SA1000 disk require that the differential lines for the write data be in the negative (DiskWriteData + at a lower voltage than DiskWriteData-) state while reading. The SA4000 specification gives no requirement. Thus UWriteEnable may be used to enable the data driver. This signal rises as soon as BWriteEnable but remains on for approximately 5 bit times after BWriteEnable falls. This allows data remaining in the MFM encoder to be written at the end of an SA1000 write operation.

The SA1000 drive needs a clock for the stepping circuitry. It should have 1/16 the frequency of the bit stream. The SA4000 drive requires a clock used to strobe each bit of the incoming NRZ data. The Input Conditioning section of the controller generates both clocks and sends the proper one on DiskOutputClock.

Both drives specify open collector buffer gates for the single ended control signals. These are driven here with either open collector inverters or open collector nand gates.

Only the SA4000 drive requires a ReadGate signal. This is to be sent each time a non-write data transfer is taking place. It must not be active at any time WriteGate is active, the disk will sense this as a WriteFault. Clearly, ReadGate should be the combination of BTransferEnable and some version of WriteEnable which causes it to be active during all non-write type operations. This version of WriteEnable must become active more quickly than DWriteEnable at the beginning of a write so ReadGate is inactive by the time WriteGate appears. This is not difficult since DWriteEnable is delayed 5 bit times from WriteEnable. The version of WriteEnable used must also become inactive more quickly than DWriteEnable but more slowly than BTransferEnable. If it were faster than BTransferEnable, there would be a ReadGate pulse at the end of each write operation, causing a WriteFault. Finally, the version used must not be generated by a register clocked by NRZClock or NRZClock'. The reason for this is a bit obscure. Since the signal is slower than BTransferEnable, there will be a ReadGate pulse at the beginning of each write operation (transfer without write => read). This does not generate a WriteFault because ReadGate should have returned to the inactive state by the time DWriteEnable turns on WriteGate. However, whenever ReadGate is sent, the drive turns off NRZClock while it locks on to the data being read. It had been generating the clock using the SA4000's clock track. The write operation will be delayed until NRZClock reappears to turn off ReadGate and turn on WriteGate via DWriteEnable. The time taken to acquire this clock is normally short but may go up to a few microseconds. This delay at the beginning of write operations has two bad effects. First, a subsequent read of the field may start before the synchronization pattern, causing the disk VCO to decode the rest of the field incorrectly. Second, one may not be able to fit all the data needed for a sector between the sector marks on the disk. The version of WriteEnable chosen is UWriteEnable. It two gate

delays later than BWriteEnable. This makes it safely slower than BTransferEnable, safely faster than DWriteEnable and keeps it independent of NRZClock or NRZClock'.

The WriteGate signal is required by both disk drives. It is active only when a write operation is taking place and an overrun has not yet occured. By using the overrun flag to stop write operations, the damage caused by an incorrectly started or interrupted write operation is limited to a single sector.

File: HSIO54.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of HSIO54.sil, sHSIO60.sil, pHSIO60.sil and pHSIO61.sil; the Analog Phase Locked Loop in the data separator of the Input Conditioning circuits for the Dandelion Disk Controller.

Overview

One task of the data separator used for the SA1000 drive is to synthesize a clock from the received data stream. This is done using an analog phase locked loop. A digital circuit designed to perform the same function was tried, but its error rate was at least 20 times that of the analog version.

The phase locked loop used here is copied from Shugart. Thay used it on their floppy disk controllers and SA4000. It contains a adjustable potentiometer. It seems to be fairly sensitive to this adjustment. A 10 to 15 ns difference in the phase of the clock relative to the average pulse arrival time seems to make a significant difference in the soft error rate (factors of 10 to 100).

The timing elements used are discrete capacitors and inductors. The actual control element is a diode whose capacitance changes as a function of bias voltage. This controls the oscillator's loop capacitance, hence its frequency.

Constraints

The bit stream arriving at the circuit is encoded using the MFM (modified frequency modulation) rules. The disk drive returns a 50 ns pulse each time a flux transition is sensed. Say we divide each bit into two halves, clock and data. We will represent the presence of a transition in a bit half by a 1, the absence by a 0. An NRZ 1 is encoded as clock|data = 01. An NRZ 0 is encoded as clock|data = 10 unless the previous NRZ bit sent was a 1. In that case, the zero is encoded as clock|data = 00.

From these rules, it can be seen that between one half and three halves of a bit may elapse between received pulses. It seems obvious that the oscillator should run at twice the NRZ bit rate so the MFM pulses always arrive on cycle boundries. The frequency of the internal oscillator may only be adjusted when a new pulse arrives. In particular, it might only receive a correction term once every three cycles, so care must be taken to avoid over-correction. The oscillator should be quite stable, yet able to lock on to the data in a new field quickly.

The phase locked loop should be capable of locking on to the signal in 60 bit times. Adding two words for start-up uncertainty gives a 7 word lock-on pattern.

The oscillator may drift out of lock if shown arbitrary data for a reasonable length of time. During seek operations the disk data outputs are arbitrary. During this period, the PLL is shown the reference clock, RefMFMClock.

Phase Locked Loop

The circuit is most easily understood by looking at HSIO54.sil, pHSIO60.sil and pHSIO61.sil. The drawing sHSIO60.sil is a version of the circuit shown on pHSIO60.sil and pHSIO61.sil. Drawing sHSIO60.sil is used for building the stichweld card. Printed circuit cards may have discrete components placed almost arbitrarily while all stichweld discretes must be mounted on platforms.

As explained above, this circuit is copied from Shugart. The version of the circuit used on the SA4000 is explained on page 7 of the *SA4000 Fixed Disk Drive Service Manual*.

The purpose of the PLL is to generate a clock that is exactly 180 degrees out of phase with the average pulse arriving from the disk. Using this clock, the rest of the data separator may decide whether a given clock or data window contains a pulse from the disk, hence may decode the disk data. Data pulses from the disk may jitter, that is some arrive early, some late. One may even see a string of early pulses, so correction should not be too drastic. Having the clock 180 degrees out of phase with the data gives the maximum leeway for both mis-timed pulses and incorrect compensation.

The input pulses are caught in the flip-flops on HSIO54.sil, producing CompareEnable and InputPulse. CompareEnable enables the InputArrived and ClockArrived flip-flops. InputPulse is delayed by 50 ns, then sets InputArrived. The rest of the circuit attempts to adjust the phase and frequency of DrvMFMClock so DrvMFMClock' arrives at the same time as DelInputData. If InputArrived is raised before ClockArrived, the frequency is raised to catch up. If ClockArrived is first, the oscillator frequency is lowered. Whenever both are set, ResetCompare' goes LO, resetting the circuit. CompareEnable is used to ensure a comparison takes place only when a pulse from the disk actually arrives. The delay line gives the clock a reasonable chance to arrive between the time the comparison circuitry is activated and InputArrived is set. Without the delay, it would be impossible to detect an early clock pulse.

We proceed now to pHSIO61.sil. InputArrived and ClockArrived are filtered at the top of the page, producing PumpUp and PumpDown respectively. These are used by the totem-pole circuits on the bottom of the page. Normally, both PumpUp and PumpDown are inactive (PumpUp' and PumpDown' are active). This means the Q103-Q104 totem-pole is turned on. The LM741 and C134 provide an error correction term with a long time constant. They change only very slowly in response to many consecutive bit errors.

When InputArrived or Clock Arrived becomes active, half of the Q103-Q104 totem-pole turns off and half of the Q102-Q105 totem-pole turns on. Say PumpUp goes active. Then Q102 turns on, raising DCError. This is the voltage used to control the oscillator. Similarily, if PumpDown arrives first, DCError is lowered. This is filtered using the resistor-capacitor network involving C132, C131 and R126. When both signals have arrived, the Q102-Q105 totem-pole is fully on. This would tend to send DCError to some value 1/2 way between PumpUpSupply and PumpDownSupply, but, as seen above, as soon as both InputArrived and ClockArrived are set, they are both reset.

The oscillator itself is shown on HSIO60.sil. The main signal path starts with Clk0 at the base of Q101, proceeds through the transistor to Clk1, the nand gate to Clk2, R110 to Clk3, L102 to Clk4 and R111 back to Clk0. The main timing elements of this Colpitts oscillator are L102 ,C118 and the VariCap, CR103. C107, C108 and L101 filter the DC clock supply voltage. The nand gate provides negative feedback and amplification.

When power is applied initially, the oscillator tends to go to a neutral DC state. Since it has negative feedback and a fair amount of filtering, it is stable without oscillating. To start it, one lowers DriveSelect, then raises it again. With DriveSelect lowered, the circuit is forced far off its equlibrium point. When DriveSelect is raised, the transition propagates through the transistor, is inverted by the nand gate, travels through the resistor and finally the inductor. The state of Clk4 is inverted and a new transition follows the old through the circuit. When it reaches Clk4, the cycle repeats.

The capacitance of the VariCap, CR103, is roughly proportional to the voltage across it. The greater the VariCap capacitance, the slower the oscillator goes. Thus when InputArrived caused PumpUp to raise DCError, the voltage difference was lowered and the oscillator frequency increased. The normal bias voltage is set using the potentiometer, P100.

Note that when a pulse train is arriving, changing the potentiometer changes the relative phase of DrvMFMClock to that pulse train. If the DC component of the voltage at ClkAdj3 is raised, the equlibrium value for DCError also rises to maintain the differential across the VariCap. Thus one may have very fine control of the phase of DrvMFMClock relative to the average input pulse.

File: HSIO55.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of the Input buffers and differential receivers in the Input Conditioning section of the Dandelion Disk Controller.

Overview

Like the output buffer drawing, this one is very simple. It shows differentially driven signals being received properly while open-collector signals are terminated and received with Schmitt-trigger gates.

Constraints

Differentially driven signals should be differentially received. The termination resistors for these signals are on a separate page because their mounting differs between the stichweld and PWB versions of the board.

The DiskReadClk receiver only functions when the controller is connected to an SA4000 type drive. When connected to an SA1000, DiskReadClk + is not connected and DiskReadClk- is grounded. DiskReadClk will be stuck at some value which depends on how the receiver views an open line. DiskReadClk is completely ignored when the controller is connected to an SA1000 drive.

Data from the disk may be encoded using either MFM or NRZ rules. The DiskReadData receiver receives either type of data. The multiplexer section of the Input Conditioning circuits must either decode this data or pass it on directly.

Open-collector signals must be pulled-up to some voltage above a TTL high level. This is done here as recommended by Shugart with a 220/330 resistor network. The 220 resistors are connected to 5 volts, the 330 resistors to ground. This gives an approximate 100 ohm termination without excessive power drain.

The single-ended signals are subject to cable noise. Shugart recommends they be received using Schmitt trigger gates as is done here. Many of the unbuffered signals are also available on the KTest port as an aid to diagnostic microcode.

File: HSIO56.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of Miscellaneous Clocks and Multiplexers for Input Conditioning circuits of the Dandelion Disk Controller.

Overview

The circuits shown on this page are generally concerned with the SA1000 portion of the controller. Only the Wrap Around multiplexer and the flip-flop producing SyncSA4Data are used when the SA4000 drive is connected. On this page, the SA1000 versions of the NRZ and MFM clocks are generated.

Constraints

The controller needs a stable time base when writing data to the disk. That data should be written at the correct frequency and should have only pre-compensation jitter. For this purpose, The 25.5 MHz signal, HalfClock, from the display is divided by 3 to produce a 117 ns period clock, RefMFMClock. This is a slightly de-rated version of the 115 ns clock specified by the SA1000 disk.

During write operations, the data written is returned to the Input Conditioning circuits. From here it is either sent back to the shift register (SA4000) or to the Data Separator (SA1000). This allows diagnostic microcode to test the data paths in the controller. A multiplexer is needed to select between the read and write data for input to the shift register. This is the Wrap Around multiplexer.

This multiplexer has another use when the SA1000 drive is connected. While seek operations are being performed, the analog phase locked loop should not be allowed to drift out of lock. The signal RefMFMClock is used to maintain calibration as it specifies the nominal data rate. The control input to the S153 multiplexer is on drawing HSIO50.sil.

The clock supplied by the SA4000 disk is 180 degrees out of phase with the SA4000 read data. This is done to minimize the effects of differing clock and data delays in the cables. The clock period is 140 ns. Unfortunately, 70 ns may not be sufficient for the Field and Word State Machine prom to produce new data. Hence a register is used to sample the received data, producing SyncSA4Data. This means the input data to the prom will change only on clock boundries and the prom will have a full bit time to produce each output.

Because of jitter in the pulse stream, a pulse may arrive at any time within a cycle. It is advantageous to have a pulse detector that does not need to be reset. While a detector is being reset, it cannot sense input pulses and may miss one. A detector meeting these requirements is shown here. The incoming pulse toggles a flip-flop. The output of the flip-flip is sampled by a register producing SyncRcvMFM. As shown on drawing HSIO57.sil, this signal is sampled to produce MFMDetected.0 which is sampled on the next DrvMFMClock to produce MFMDetected.1. When MFMDetected.0 differs from MFMDetected.1, a new pulse has been detected. MFMDetected.0 should not enter a meta-stable state since DrvMFMClock is offset from properly tracked data transitions by exactly one half of a clock period. Thus a pulse may arrive approximately 67.5 ns early or late and still be decoded correctly.

During Read or Verify operations, the controller receives data from the disk, so NRZClock should be derived from that data. During Write operations, a stable reference clock should be used. The multiplexer used to choose between these when controlling an SA1000 is shown at the bottom of the page. When receiving data, the clock is derived from the least significant bit of the address mark state count, AdrMkCnt.4. This changes with each tick of the DrvMFMClock. When sending data, the NRZClock is derived from the reference clock. The divider producing SA1WrNZClock from RefMFMClock is shown on drawing HSIO58.sil.

The multiplexer is controlled by a signal that becomes active as soon as BWriteEnable goes HI and

remains active 5 bit times after it goes LO. We note that AdrMkCnt.4 becomes inactive when BTransferEnable drops. Having the multiplexer choose the reference clock as soon as BWriteEnable goes true allows a write operation to begin without undue delay. DWriteEnable is produced by shifting BWriteEnable through a shift register with NRZClock'. In order to ensure that DWriteEnable goes LO at the end of a write operation, we must guarantee an NRZClock. Thus NRZClock remains active until DWriteEnable drops.

File: HSIO57.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of Data Separator and Address Mark recognizer in Input Conditioning circuits of the Dandelion Disk Controller.

Overview

The circuits shown on this page are concerned with the SA1000 portion of the controller. The NRZ and MFM clocks used when the SA1000 is connected are generated on this drawing. The data pulses are deocded and address marks are recognized here.

Constraints

Address marks must be recognized when an SA1000 is used. They serve three purposes. First, the end of an address mark is used by the Word State machine to find the first word boundry in the field being read. Second, in recognizing the address mark, the prom recognizes the phase of the incoming data, this is reflected in AdrMkCnt.4. Third, a tag in bit 14 of the address mark is used to determine whether the address mark belongs to a Header field or a Label or Data field. The presence of this bit is signalled by a pulse on SA1Sector. This pulse is caught in the flip-flop shown on drawing HSIO58.sil and made available as the SectorFound status bit.

During SA1000 write operations, the data written is returned to the data separator. This allows diagnostic microcode to test the data paths in the controller by comparing the Tag bit of the address mark sent with that of the address mark received and decoded. A multiplexer is needed to select between the read and write data for input to the data separator. This is the Wrap Around multiplexer shown on drawing HSIO56.sil.

As a byproduct of the wrap-around function, the re-received data is decoded and made available to the shift register. Unfortunately, the shift register clock is derived from RefMFMClock and the re-received NRZ Clock is derived from DrvMFMClock. Because the phase of these two clocks cannot be predicted accurately, one cannot guarantee the data clocked back into the shift register will match the data sent. Thus the complete data path is not checked. It was deemed more important to check the data separator and phase locked loop by having it regenerate the clock and data than to check the small data path between the data separator and the shift register.

The address mark recognition circuitry must be disabled when the SA4000 is used. If AddrMkFound were accidently raised during the synchronization process, the beginning of the field would be improperly recognized. When false, the SA1000/SA4000' signal disables the address mark recognition process.

Address Mark Recognizer

This state machine has a counter used to recognize address marks and a state bit telling it whether the address mark has passed. The address mark in MFM code is: 01 00 01 00 10 *00* 10 01 00 01 00 10 *00* 10 tt 01. The italicized zeros represent missing clock transitions. The "tt" is a tag bit used to distinguish Header address marks from Label and Data address marks. This is used by the microcode to find the Header fields. It is assumed that the two fields following a Header field are Label and Data in that order.

A detailed listing of the prom contents may be found in AddrMkProm.mesa in [Iris]<Workstation>HSIO>Proms>DDCProms-Rev-A.dm. Note the circuit uses two 1024x4 proms instead of one 512x8 prom. This was done because 512x8 proms were very difficult to obtain.

The machine starts in state 00000 after having been held reset by the absence of BTransferEnable. The

phase locked loop synchronization pattern is all zeros. These are encoded as 10 10 10 .... Seeing these, the machine cycles through states 0, 1, 2 and 3, going to state 1 when the 0 is seen, state 2 for the next 1, 3 for the next 0 and back to 0 because of the next 1. When the first 01 of the address mark is seen, the machine will either be in state 1 or state 3. If in state 1, the initial 0 causes it to remain in state 1 and the rest of the address mark is recognized properly. If it was in state 3, the initial 0 drives it to state 4 and the next 1 back to state 2. From there, the rest of the address mark is recognized properly. The state in AdrMkCnt advances as each correct bit is seen. If a bad transition is seen or a good one missed, the machine goes back to state 0 1 or 2.

When the count reaches 27 and PulseInWindow = 0, the address mark has been recognized so AddrMkFound is raised on the next clock edge. This tells the Field and Word state machine in the Serializer/DeSerializer to prepare to recognize the end of the Sync word (or in this case the address mark). In the Word state machine BitCount←15 on the next NRZClock edge and 00 on the edge afteer that when SyncWdFound is set. The clock edge that raises AddrMkFound presents Bit 14 to both the address mark recognizer and to the Word Machine.

The tag bit must be sampled at the end of NRZ bit 14 or when AdrMkCnt = 29. If it was set, SA1Sector is set for one cycle. This is caught by the flip-flop on drawing HSIO58.sil and may be sampled by microcode on the SectorFound bit of the KStatus port.

As one might think, the derived NRZ clock is taken directly from AdrMkCnt.4. The NRZ input data is taken from AdrMkCnt.2. After recognizing the address mark and recording the value of the tag bit, the state machine goes through states 0 and 1 each time an NRZ zero is decoded and states 4 and 5 for each NRZ one. Note the potential race between AdrMkCnt.4 and AdrMkCnt.2 is avoided by having only one change at a time. AdrMkCnt.3 was not used to generate the NRZ data for historical reasons. The address mark used to begin with 10, not 01. This was changed to ensure a correctly encoded zero followed each bit containing a deleted clock transition. It would now be permissible to re-program the address mark proms and use AdrMkCnt.3 if there were reasonable need.

File: HSIO58.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of Input Multiplexer in Input Conditioning circuits of Dandelion Disk Controller

Overview

The SA1000 and SA4000 drives differ in data encoding, data rate and the numbers and types of control and status signals needed. The controller given in these drawings serves both drives. The Input Multiplexer serves to standardize the differing signals so they may be used by microcode in relatively similar ways. For example, the SeekComplete signal leaving the Input Multiplexer is active only after the head settling time even though this is not true of the version supplied by the SA4000 drive.

Constraints

As mentioned above, the SeekComplete signal used by the microcode should only become true after the disk heads have settled. This is not true of the SA4000 signal. The controller delays the SA4000 SeekComplete by 29 sector counts, or at least 20 ms, before sending it on. Note the SeekComplete signal may also be delayed until the heads have settled after a DriveSelect. The SA4000 drive may be jumpered so the absence of DriveSelect turns off the holding current to the stepper motor. This may result in a substantial power savings. After DriveSelect is re-asserted, one must wait 20 ms for the heads to settle after re-acquiring the cylinder. One may do this by waiting for SeekComplete to become true. This is most easily done by setting the condition on which service requests are sent to "SeekComplete = 1".

The SectorFound flag has two distinct, though related, meanings. When connected to an SA4000 drive, it is a latched version of the Sector signal suppled by the drive. That drive has a clock track and a counter. It uses these in conjunction with the Index mark to generate periodic Sector pulses. The SA1000 drive has no sector pulses. Bit 14 of each address mark is a tag bit which is made available on the SectorFound line. This bit is set when the address mark belongs to a Label or Data field, reset when the address mark starts a Header field. The bit may be used as error flag to be tested along with the VerifyError and CRCError flags after a Header operation. When set, it means the field found was not a Header field, hence not the start of a sector.

The signal SA1Sector rises at the beginning of Bit 15 of the address mark if the tag in Bit 14 was set. Either this signal or the buffered sector pulse is caught in a status flip-flop for use in generating a Service Request and as a status flag. All status flip-flops are reset by the microcode using the "ClrKFlags" command.

The Index pulse from the SA4000 drive arrives once per disk revolution (~20 ms) and lasts only 1.1 us. The microcode can sample the status flags every 2.05 us, not quickly enough to be sure of catching the Index pulse. Because of this and because busy waiting slows the Emulator task, this signal is also caught in a flip-flop for later use. The SA1000 also supplies an index pulse. Both the SA1000 and SA4000 index pulses mark a particular spot on the disk. Microcode will generally use this to mark the start of sector 0. Note the raw Index' line from the disk is not available on the KTest port. It was deleted to make room for the diagnostic signals used for the Display. The latched version is seen as IndexFound on the KStatus port.

Both the SA1000 and SA4000 drives require clocks from the controller but the clocks are quite different. The SA1000 drive uses the clock to control its stepping circuitry. The SA1000 TimingClock should have 1/16 the frequency of the NRZ bit clock. The SA4000 uses the clock to sample the incoming NRZ WriteData, hence its clock should be matched in frequency and phase with the outgoing data. The SA1000 clock is obtained by dividing the RefMFMClock by 32 since the MFM clock has twice the frequency of the NRZ clock. The SA4000 clock is obtained directly from the ReadClock supplied by the drive. The engineers at Shugart say the SA4000 write clock is only needed to account for cable delays. They assume the clock and data under go the same delay so will still be in phase when they reach the drive. Thus the clock may be used to sample the data reliably.

The NRZClock used by the controller is either derived from the data (SA1000) or received directly from the drive (SA4000), having been derived from the data there. This choice is made here.

Like the NRZClock, the NRZInput data is derived by a data separator either on the controller or on the drive. When derived on the controller, AdrMkCnt.2 reflects the value of the data bit as explained in HSIO57.bravo. Note that when the SA1000 is attached and data is being received, NRZClock rises in the middle of the data bit. This is to avoid a race between the data in AdrMkCnt.2 and the clock in AdrMkCnt.4. In all other cases, NRZClock rises on the edge of the data bit.

File: HSIO59.bravo in [Iris]<Workstation>HSIO>DDC-Q-Rev.DocDm
Contents: Description of sHSIO59.sil and pHSIO59.sil, the disk connectors on stichweld and PWB boards.

Overview

Cable numbering conventions differ between the stichweld and PWB boards, hence the need for two drawings. The drawing sHSIO59.sil shows the stichweld, pHSIO59.sil shows the PWB version.

The SA1000 drive connects to its controller with a 50 conductor and a 20 conductor ribbon cable. The stichweld card has positions for 37 pin connectors. Drawing sHSIO59.sil shows a mapping between the 50 conductor cable and a 37 pin D-type connector and a mapping between a 20 conductor cable and a 16 pin DIP socket connector. No 20 pin DIP socket connectors are available.

An additional connector must be built to connect the SA4000 drive. For a specification of this connector see drawing DDC06.sily. The new connector maps both the 50 and 20 conductor cables, there called HSIO50 and HSIO20, into a single 50 conductor cable, called ExtDk50. The disk drive must be configured so that all status and data lines are available on the single 50 pin cable.

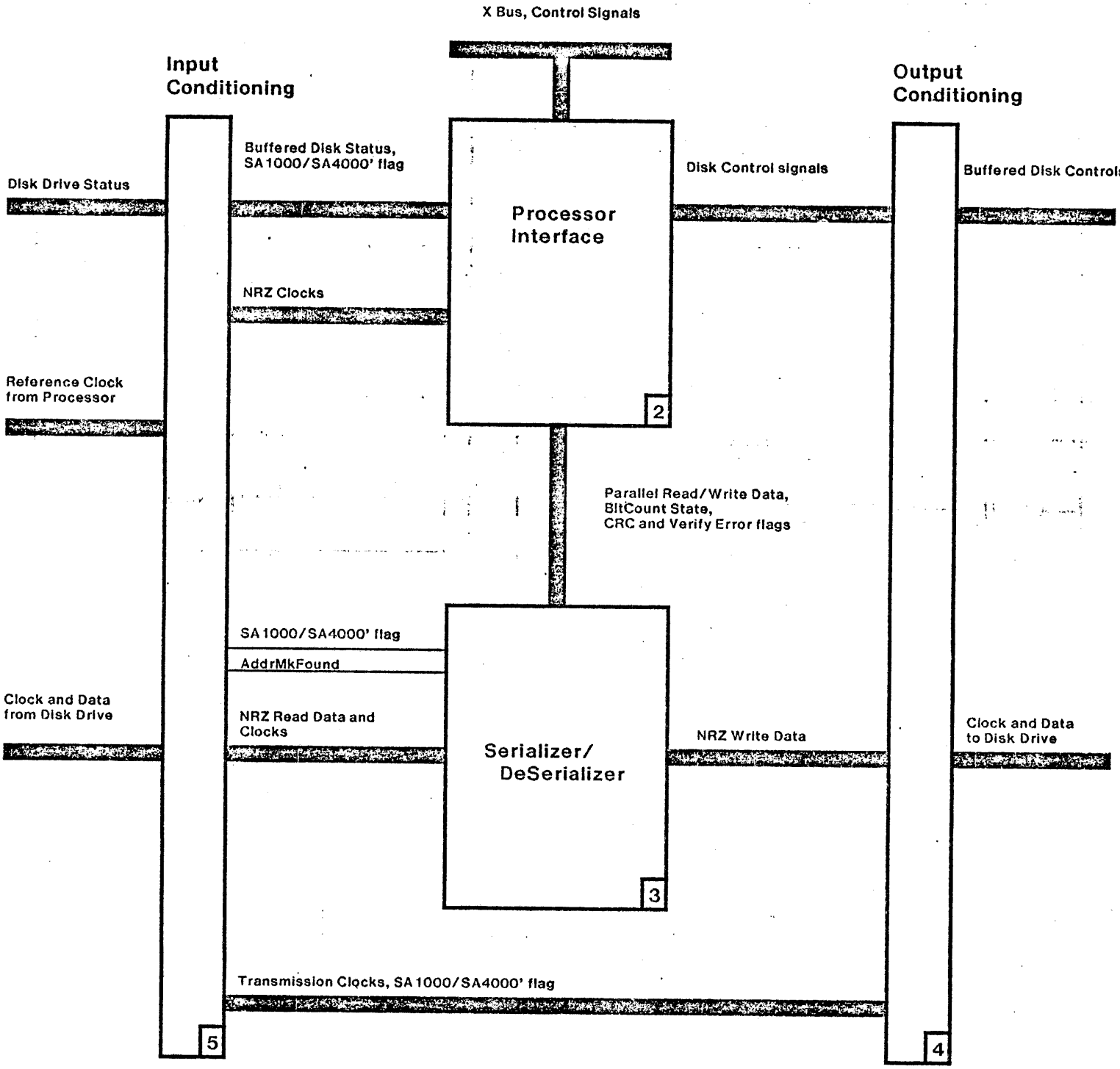
The four 51 ohm termination resistors are also shown on these drawings because the method of specifying them also differs between stichweld and PWB boards.

Constraints

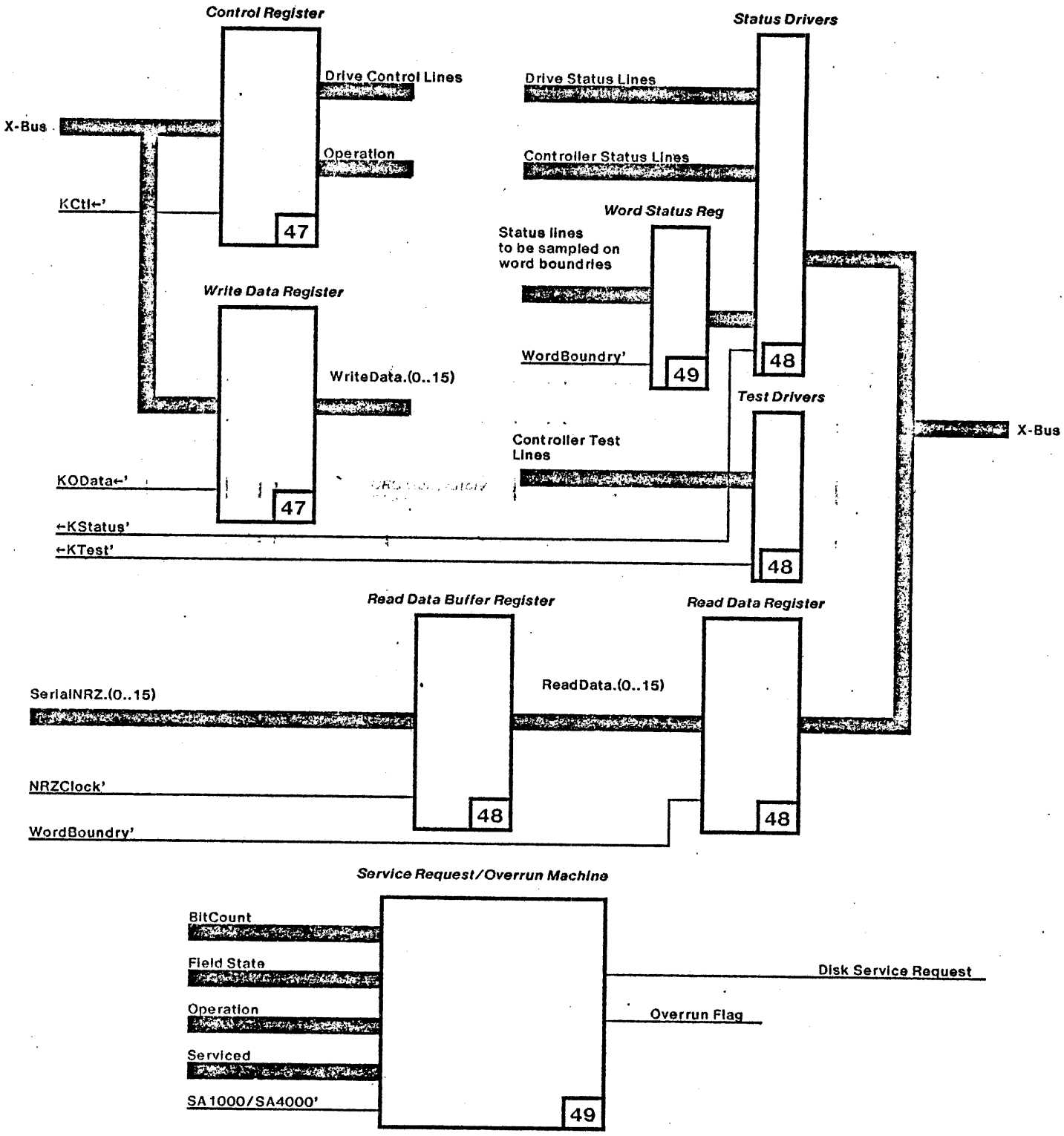
The most obvious constraint in the cabling on the stichewld board is that a 50 conductor cable must be connected to a 37 pin connector. This is done by eliminating spare lines and some ground returns. An effort has been made to keep the number of breaks in consecutive numbering to a minimum. Where there are breaks, they are concentrated in one area so the cablemaker may focus attention there. The ground returns eliminated have come from between signals that seldom change (DriveSelects).

The 20 conductor data cable has been mapped to a 16 pin connector by leaving off the first four lines. These were not used by the drive.

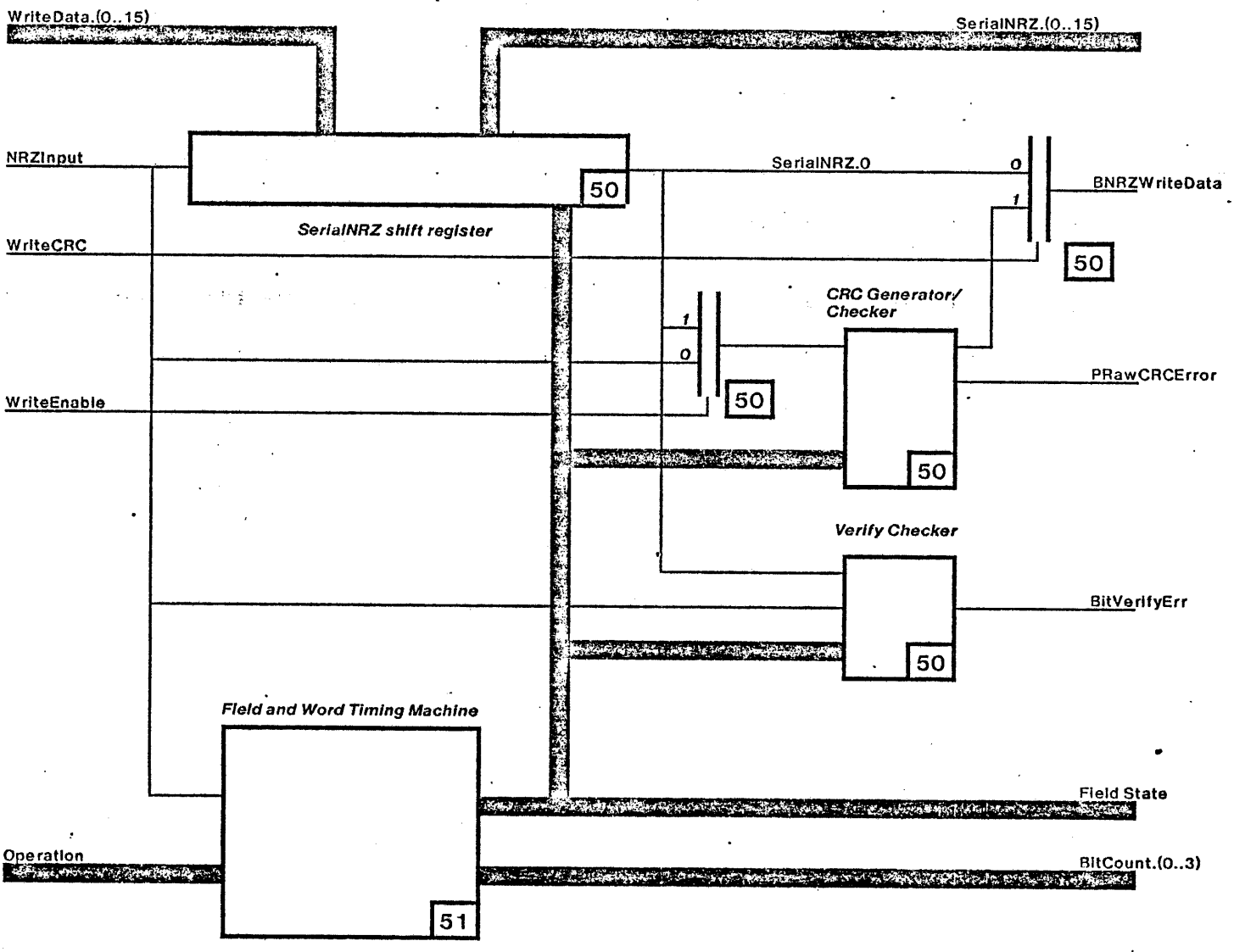
The PWB board has room for the full 50 and 20 pin connectors. Drawing pHSIO59.sil shows how they are connected. Italicized pin numbers denote unused pins.



XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Controller Block Diagram	File: DDC01.sily	Designer: Davies	Rev: Q	Date: 7/25/80	Page: 01

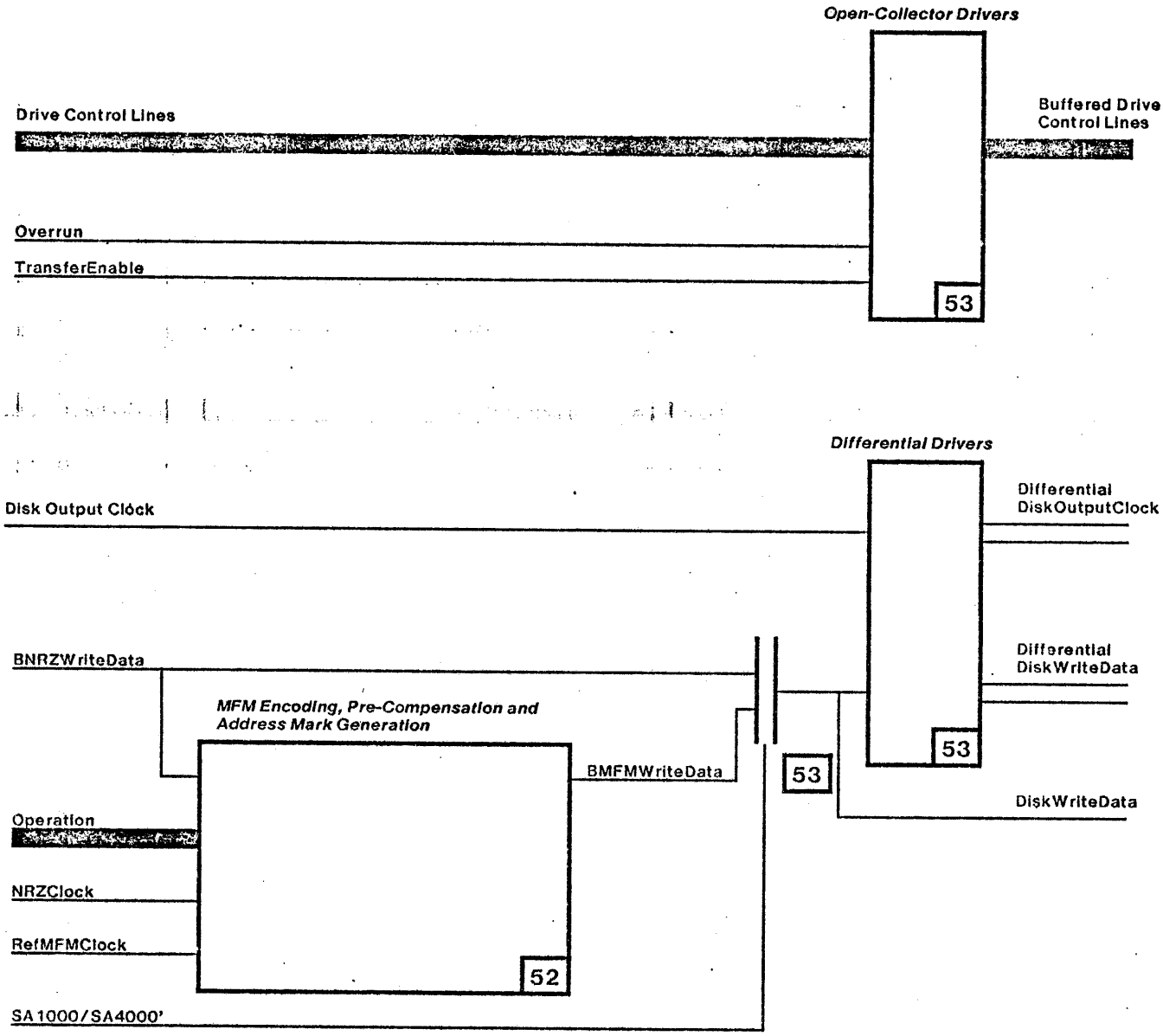


XEROX SDD	Project Dandelion	Dandelion Disk Controller Processor Interface	File DDC02.sily	Designer Davies	Rev Q	Date 7/25/80	Page 02

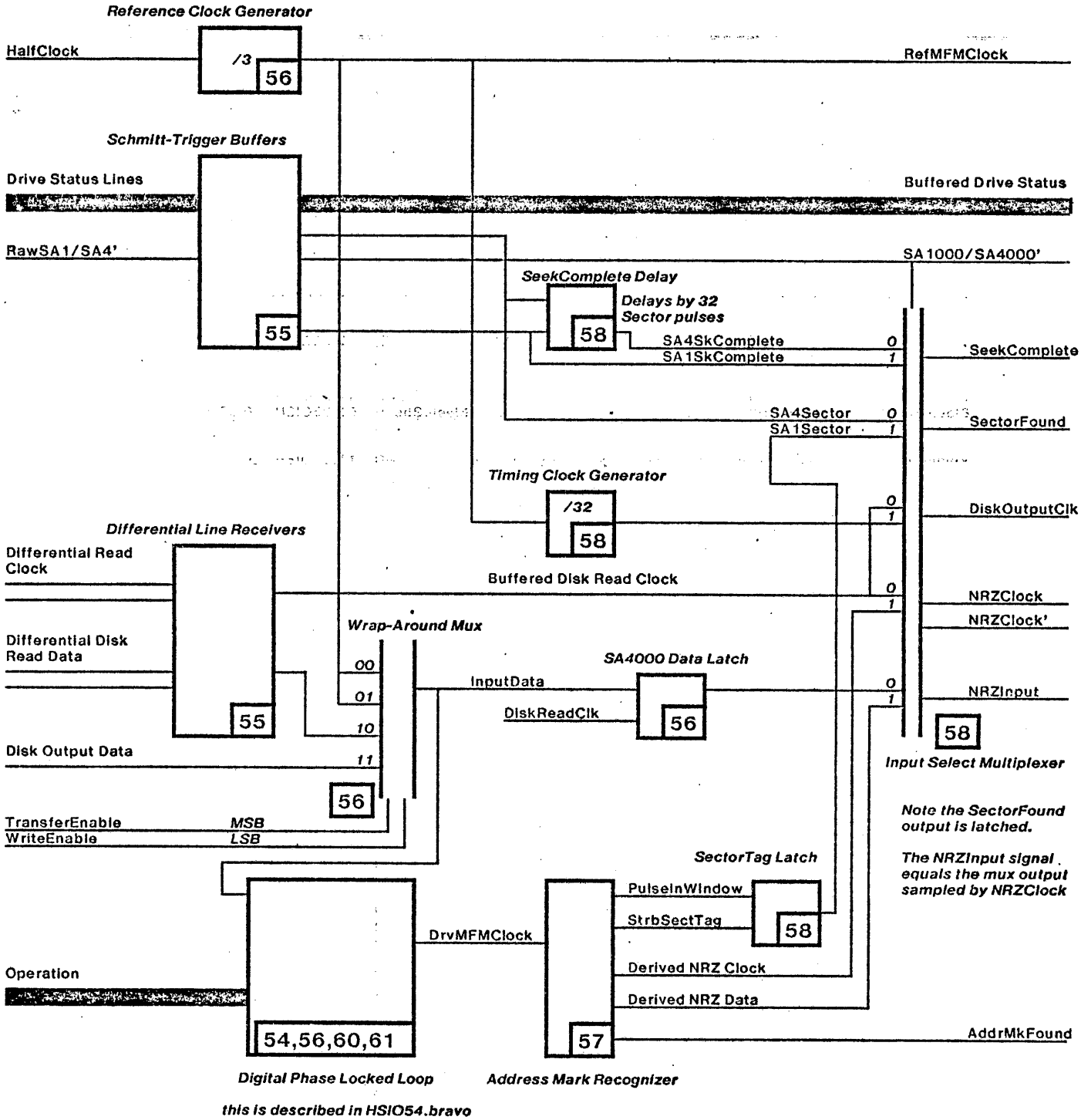


Field State = {SyncWdFound, PLdSerialNRZ', WordBoundry'}

XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Serializer/DeSerializer	DDC03.sily	Davies	Q	7/25/80	03

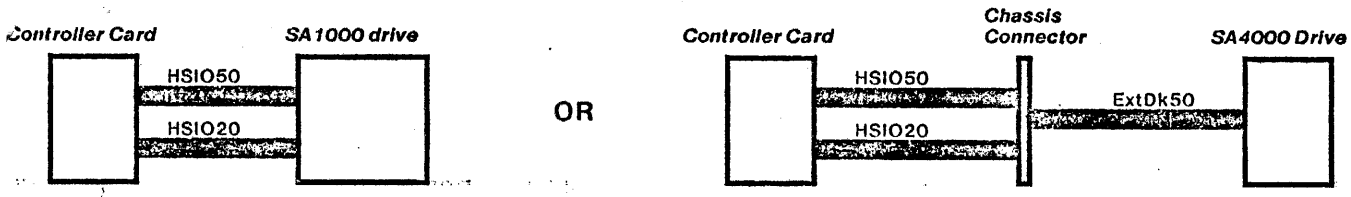


XEROX SDD	Project Dandelion	Dandelion Disk Controller Output Conditioning	File DDC04.sily	Designer Davies	Rev Q	Date 7/25/80	Page 04



XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Input Conditioning	DDC05.sily	Davies	Q	8/6/80	05

The I/O portion of the Dandelion Disk Cabling is arranged as follows:



HSIO50 is a 50 pin ribbon cable, connects directly as J1 connector on SA1000 drive or to chassis connector for SA4000.
HSIO20 is a 20 pin ribbon cable, connects directly as J1 connector on SA1000 drive or to chassis connector for SA4000.
ExtDk50 is a 50 conductor round wire cable, it connects the SA4000 chassis connector to the J1 connector on the SA4000 drive

The connections on the HSIO card are pin-for-pin compatible with those on the SA1000 type drives.

The signals are arranged on the cables as shown below.

Connector | Signal

All Odd numbered pins from HSIO50.1 through HSIO50.49 are connected to signal grounds in both the drive and the connector card.

HSIO50.2	ReduceIW'
HSIO50.4	RawSA1/SA4'
HSIO50.6	Sector'
HSIO50.8	SeekComplete'
HSIO50.10	HeadSelect4'
HSIO50.12	HeadSelect8'
HSIO50.14	HeadSelect1'
HSIO50.16	HeadSelect16'
HSIO50.18	HeadSelect2'
HSIO50.20	Index'
HSIO50.22	Ready'
HSIO50.24	<not used>
HSIO50.26	DriveSelect1'
HSIO50.28	DriveSelect2'
HSIO50.30	DriveSelect3'
HSIO50.32	FaultClear'
HSIO50.34	DirectionIn'
HSIO50.36	Step'
HSIO50.38	ReadGate'
HSIO50.40	WriteGate'
HSIO50.42	Track00'
HSIO50.44	WriteFault'
HSIO50.46	<not used>
HSIO50.48	<not used>
HSIO50.50	<not used>

HSIO20.1	<not used>
HSIO20.2	GND
HSIO20.3	<not used>
HSIO20.4	GND
HSIO20.5	DiskReadClk +
HSIO20.6	DiskReadClk-
HSIO20.7	<not used>
HSIO20.8	GND
HSIO20.9	DiskOutputClk +
HSIO20.10	DiskOutputClk-
HSIO20.11	GND
HSIO20.12	GND
HSIO20.13	DiskWriteData +
HSIO20.14	DiskWriteData-
HSIO20.15	GND
HSIO20.16	GND
HSIO20.17	DiskReadData +
HSIO20.18	DiskReadData-
HSIO20.19	GND
HSIO20.20	GND

ExtDk50	HSIOxx	Signal Name
ExtDk50.1	HSIO50.4 *	GND
ExtDk50.2	HSIO50.14	HeadSelect1'
ExtDk50.3	HSIO50.13	GND
ExtDk50.4	HSIO50.18	HeadSelect2'
ExtDk50.5	HSIO50.17	GND
ExtDk50.6	HSIO50.10	HeadSelect4'
ExtDk50.7	HSIO50.9	GND
ExtDk50.8	HSIO50.12	HeadSelect8'
ExtDk50.9	HSIO50.11	GND
ExtDk50.10	HSIO50.20	Index'
ExtDk50.11	HSIO50.19	GND
ExtDk50.12	HSIO50.22	Ready'
ExtDk50.13	HSIO50.21	GND
ExtDk50.14	HSIO50.6	Sector'
ExtDk50.15	HSIO50.5	GND
ExtDk50.16	HSIO50.26	DriveSelect1'
ExtDk50.17	HSIO50.25	GND
ExtDk50.18	HSIO50.28	DriveSelect2'
ExtDk50.19	HSIO50.27	GND
ExtDk50.20	HSIO50.30	DriveSelect3'
ExtDk50.21	HSIO50.29	GND
ExtDk50.22	HSIO50.8	SeekComplete'
ExtDk50.23	HSIO50.7	GND
ExtDk50.24	HSIO50.34	DirectionIn'
ExtDk50.25	HSIO50.33	GND
ExtDk50.26	HSIO50.36	Step'
ExtDk50.27	HSIO50.35	GND
ExtDk50.28	HSIO50.26	FaultClear'
ExtDk50.29	HSIO50.25	GND
ExtDk50.30	HSIO50.40	WriteGate'
ExtDk50.31	HSIO50.39	GND
ExtDk50.32	HSIO50.42	Track00'
ExtDk50.33	HSIO50.41	GND
ExtDk50.34	HSIO50.44	WriteFault'
ExtDk50.35	HSIO50.43	GND
ExtDk50.36	HSIO50.38	ReadGate'
ExtDk50.37	HSIO50.37	GND
ExtDk50.38	HSIO50.16	HeadSelect16'
ExtDk50.39	HSIO20.13	DiskWriteData +
ExtDk50.40	HSIO20.14	DiskWriteData-
ExtDk50.41	HSIO20.15	GND
ExtDk50.42	HSIO20.10	DiskOutputClk-
ExtDk50.43	HSIO20.9	DiskOutputClk +
ExtDk50.44	HSIO20.11	GND
ExtDk50.45	HSIO20.5	DiskReadClk +
ExtDk50.46	HSIO20.6	DiskReadClk-
ExtDk50.47	HSIO20.4	GND
ExtDk50.48	HSIO20.17	DiskReadData +
ExtDk50.49	HSIO20.18	DiskReadData-
ExtDk50.50	HSIO20.19	GND

* grounding this line indicates the controller is connected to an SA4000 type drive.

High Speed Input/Output Board

Logic Drawings

HSIO Board

1. HSIO0 - this page
2. pHSIO01 - drawings of fuses
3. HSIO02.sily - Display controller parts list
4. HSIO03.sily - Display controller parts list
5. HSIO04.sily - Disk controller parts list
6. HSIO05.sily - Disk controller parts list
7. HSIO06.sily - Stichweld layout
8. HSIO07.sily - proposed PWB layout

Display Controller

1. HSIO22 - 51 MHz Clock Dividers and ECL Terminators
2. HSIO23 - Cycles, Clicks and Display counter
3. HSIO24 - Display Output Machine and Control register
4. HSIO25 - Data FIFO and Border Register
5. HSIO26 - Control FIFO-Data Path
6. HSIO27 - Read Machine; Word Counter & End Conditions
7. HSIO28 - LCAS & LRAS' Generation
8. (p|s)HSIO29 - Discretes, Connectors

Disk Controller

8. HSIO47 - Control and Write Data registers
9. HSIO48 - Status / Test Multiplexer, ReadData Register
10. HSIO49 - Service Request, Overrun and Word Status Buffer
11. HSIO50 - Serializer / DeSerializer
12. HSIO51 - Field/ Word Machine
13. HSIO52 - MFM Encoding, Pre-Compensation and Address Mark Gen.
14. HSIO53 - Disk Output Buffers and Drivers
15. HSIO54 - Logic for Phase Decoder
16. HSIO55 - Disk Input Buffers and Receivers
17. HSIO56 - Miscellaneous Input Clocks and Multiplexing
18. HSIO57 - Data Separator and Address Mark Detection
19. HSIO58 - Input Multiplexer
20. sHSIO59 - Disk Cables Connections for stichweld card
21. pHSIO59 - DiskCables, Terminators for PWB card
22. sHSIO60 - Discrete Phase Decoder Oscillator, Stichweld version
23. pHSIO60 - Discrete Phase Decoder Oscillator, PWB version
24. sHSIO61 - Discrete Phase comparator, Stichweld Version
25. pHSIO61 - Discrete Phase comparator, PWB version

Other Documentation

This file is in:	[Iris]<Workstation>HSIO>HSIO-Rev-I.press	All logic drawings in Press format
	[Iris]<Workstation>HSIO>HSIO-L.dm	All design Automation info about HSIO board
	[Iris]<Workstation>HSIO>HSIO-L.dm	Wirelist for this rev of HSIO board
	[Iris]<Workstation>HSIO>DDC-Rev-A.DocDm	Disk Documentation in SII and Bravo formats
	[Iris]<Workstation>HSIO>DDC-Rev-A.press	All Disk Documentation in Press format
	[IRIS]<Workstation>HSIO>Proms>DDCProms-Rev-A.dm	Disk Prom Programs
	[Iris]<Workstation>HSIO>WSD-Rev-C.DocDm	Display Documentation drawings, Timing Diagrams
	[Iris]<Workstation>HSIO>WSD-Rev-C.press	All Display Logic, Timing diagrams in Press format
	[Iris]<Workstation>HSIO>Proms>DisplayProms-Rev-A.dm	Display Prom Programs

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

MATERIAL LIST

ML	Drawing No.	Rev.
		Q

Rev.	Drawing Title
	Dandelion High Speed I/O Board Clock & Display Controller

Eng. No.

ML

Model No.	Date	Sheet
	7/23/80	1 of 2

Item No.	Drawing Title			Drawing No.	No. Req.	Remarks
	TTL Integrated Circuit		SN74S00		2	
			SN74S02		1	
			SN74S04		3	
			SN74S37		2	
			SN74LS74		1	
			SN74S74		1	
			SN74LS85		2	
			SN74LS139		1	
			SN74LS163		7	
			SN74LS175		1	
			SN74S175		1	
			SN74S225		8	
			SN74S241		3	
			SN74LS273		1	
			SN74S373		2	
	TTL Integrated Circuit		SN74S374		5	
	TTL PROM	Fairchild	93427		1	EndCnt Prom
	TTL PROM	Fairchild	93453		2	Display Prom Vert Prom
	ECL Integrated Circuit		10016		3	
			10102		2	
			10103		1	
			10104		1	
			10124		4	
			10125		3	
			10131		3	
			10141		2	
			10158		1	
			10176		2	
	ECL Integrated Circuit		10231		3	

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

MATERIAL LIST

ML	Drawing No.	Rev.
		Q

Rev. Q

Dwg. No. ML

Drawing Title

Dandelion High Speed I/O Board Clock & Display Controller

Model No.	Date	Sheet
	7/16/80	2 of 2

Item No.	Drawing Title	Drawing No.	No. Req.	Remarks
	TTL Clock 51.04 MHz 14 pin package Motorola K1100 series or equivalent	733W01805	1	
	Terminator Resistor Package 160 ohms to Gnd Allen Bradley 316E161261 260 ohms to -5.2V	703WXXX91	5	Equiv. to 100 ohms to - 2 V.
	Resistor .25 watt 5% 220 ohms	703W30688	2	R1, R2
	Resistor .25 watt 5% 100 ohms	703W29888	3	R3, R4, R5
	Capacitor .01 uF 25 V.	702W07118	2	C10, C11
	Inductor Nytronics SWD-56 5.6 uH 10%	705W00022	2	L10, L11
	Inductor Nytronics SWD-56 1.0 uH 10%	705W00013	1	L12
	Fuse Slow Blow 15 amp	708W11502	1	F001
	Fuse Slow Blow 7 amp	708W11302	1	F002
	Capacitor Bypass 35 uF, 50 V	702W10701	2	C01, C02
	Diode, Protection 1N4148	707W00273	2	CR20, CR21
	0.1 uF bypass capcitors to be supplied 1 per every 2 chip positions			

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

MATERIAL LIST

ML	Drawing No.	Rev.
		Q

Rev. Q

Drawing Title

Dandelion High Speed I/O Board Disk Controller

Model No.	Date	Sheet
	7/16/80	1 Of 3

ML

Item No.	Drawing Title			Drawing No.	No. Req.	Remarks
	TTL Integrated Circuit		SN74S00		2	
			SN74S02		1	
			SN74S04		3	
			SN7406		2	
			SN74S08		1	
			SN74LS10		1	
			SN7414		2	
			SN74S38		2	
			SN74S74		9	
			SN74S86		1	
			SN74S151		2	
			SN74S153		1	
			SN74S175		2	
			SN74199		2	
			SN74S240		4	
			SN74S257		3	
			SN74LS273		14	
			SN74S374		3	
			SN74LS393		2	
			SN75107		1	
	TTL Integrated Circuit		SN75114		1	
	TTL PROM	Fairchild	93427		2	
	TTL PROM	Fairchild	93453		5	
	CRC Chk/Gen.	Fairchild	9401		1	
	Digital Delay Line		DDU-4-5050	744W00002	2	taps at 10, 20, 30, 40, 50 ns
	Operational Amplifier		LM741 (uA741)	733W00021	1	8 pin Mini-Dip
	Connector		10 position	713W12220	1	Display connector
	Connector		20 position	713W10320	1	Disk Data connector
	Connector		50 position	713W10820	1	Disk Control/Status conn.

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

MATERIAL LIST

ML	Drawing No.	Rev.
		Q

Rev. Q

Drawing Title

Dandelion High Speed I/O Board Disk Controller

Model No.

Date 7/16/80

Sheet 2 Of 3

ML

Item No.	Drawing Title	Drawing No.	No. Req.	Remarks
	Resistor pack Beckman 898-5-R220/330	703W30491	2	
	Capcitor 50 V, 5%, 47 pF	102P20275	1	C124
	5%, 100 pF	102P20279	2	C118. C123
	10% 0.0027 uF	702W30527	1	C131
	10% 0.027 uF	702W31727	1	C132
	+80%, -20%, 0.1 uF	702W05218	7	C107, C120-122, C133, C140, C141
	10% 0.47 uF	702W05708	1	C134
	Capcitor 50 V, 10%, Tantalum, 1.0 uF	702W31905	2	C108, C119
	Diode, 0.5 watt, 1N4148	703W00273	1	CR101
	Diode, 0.5 watt, 2.4V Zener, 1N5221B	707W00150	2	CR102, CR104
	Diode, VariCap, MV1404		1	CR103
	Inductor, 10%, 10 uH	705W00025	1	L102
	Inductor, 10%, 22 uH	705W00029	1	L101
	Resistor 0.25 watt 5% 47 ohm	703W29088	5	R10-14
	100 ohm	703W29888	2	R111, R128
	110 ohm	703W29988	1	R126
	150 ohm	703W30288	1	R114
	180 ohm	703W30488	2	R136, R124
	200 ohm	703W30588	4	R115, R116, R133, R135
	240 ohm	703W30788	1	R137
	270 ohm	703Wxxx88	1	R109
	470 ohm	703W31488	1	R110
	510 ohm	703W31588	5	R103, R112, R117, R120, R121
	910 ohm	703W32188	2	R122, R123
	1.0 KOhm	703W32288	9	R127
	1.1 KOhm	702W32388	2	R132, R134
	2.0 KOhm	703W32988	2	R125, R138
	4.7 KOhm	703W33888	1	R105
	Resistor 0.25 watt 5% 200 KOhm	703W37788	1	R113

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

MATERIAL LIST

ML	Drawing No.	Rev.
		Q

Rev.

Dwg. No.

Drawing Title

Dandelion High Speed I/O Board
Disk Controller

Model No.

Date 7/16/80

Sheet 3 of 3

ML

Item No.	Drawing Title	Drawing No.	No. Req.	Remarks
	Potentiometer, 0.75 W, 10% 5.0 KOhm	703W00922	1	R104
	Transistor, 2N5770		3	Q101, Q104, Q105
	Transistor, 2N5771		2	Q102, Q103

1 10 20 30 40 50 (101 150) 51 60 70 80 90 100 (151 200)

Note: All chips inside region enclosed by heavy black line are inserted upside down in sockets.
51 ohm resistors plugged into 2-19 and 3-18 of a1, a2 and into 1-20 of b2.

	a	b	c	d	e	f	g	h	i	
19	S37 CAS,RAS', DCAS',Ctr.	S02 Cycle3,pCk', pWCk',P/F'	S37 LRAS,LCAS, ppClk(2)	Term. A 316E161261	LS374 Border 0-7	LS374 Border 8-15	S374 DData 0-7	S374 DData 8-15	S225 DDataFlf 13-15	19
18	S241 * Cycles, Clicks	10125 CAS,RAS, Wpulse,ppClk	10125 LCas,LRas DCasDly,Dis/P	10231 LRas,LCas	S373 CtlFlfReg 0-7	S373 CtlFlfReg 8-15	S225 Ctl Fifo 6-10	S225 CtlFifo 11-15	S225 DDataFlf 8-12	18
17	LS163 Cycles	10231 RAS,PRas	10231 CAS,ppClk	10158 preLRAS, preLCAS	S225 CtlFifo 1-5	LS273 CtlReg. 8-15	S225 CtlFifo 0	S225 DDataFlf 0-4	S225 DDataFlf 5-7	17
16	LS163 Clicks	10016 Clk Ctr.	10102 ECL1,pRAS Full',Setup	Term. B 316E161261	S241 BufX.[0..7]	LS139 Byte Sel	LS163 DAddr 10-12	LS163 DAddr 13-15	LS85 AComp 13-15	16
15	S74 HalfCk, Vert.	10176 Clk, DCasDly	10016 LRas,LCas	10176 LRAS-LCAS Gen , Tick67'	S241 BufX.[8..15]	S175 OutMachine Byte Sync.	LS163 EndPromCtr	LS85 AComp 10-12	S00 DPReq,RDF F/P',ULCF	15
14		10131 LRAS preLCAS'	10102 preLRAS (2) Full (2)	10103 * pDis/Pr,OCtr. Video,	Term. D 316E161261	10124 CGen, ELin	93427 EndCountProm	LS175 EndCtPrmReg	S00 * DCtl,DCtlFlfo, DBorder,	14
13	10124 C3,Clk,Pblk, Inv.	10131 Setup preDisp/Proc	Term. C 316E161261	10104 RCtrPE,PPLC, Video,LdSR'	10016 OutCtr.	10125 OutMach,LTick	10124 DByte 0-3	S04 DCFlf,DBor,DC DAdr(2),InhRd	LS74 BPBS DPReq	13
12	a S374 *	b LS273	c LS273		e 10141	f 10141	g 10124	h S374		12
	Proc. Sync, DPReq, ClrKFlgs, ClrDPReq,	DiskCntl.[0..7]	DiskCntl.[8..15]		OutSR 0-3	OutSR 4-7	DByte 4-7	DProm Reg.		
11	S240 KStat.[0..7]	S240 KStat.[8..15]	S240 KTest.[0..7]	S240 KTest.[8..15]	S00 pDPReq, Clock Qual.	10131 Video, Blk	Term. E 316E161261	93453 DisplayProm	93453 VertProm	11
10	S74 RefMFMClk	LS10 Serviced', ResetSkComp	LS273 WrDat.[0..7]	LS273 WrDat.[8..15]	S04 MFMCk, NRZCk KCtl, KOData SSrvcd, Clk	OutPlat	LS163 DProm 0-3	LS163 DProm 4-7	S04 * Ctr,HSyn, VSyn,WP' PullUp	10
9	LS273 PRdDat.[0..7]	LS273 PRdDat.[8..15]	S374 RdDat.[0..7]	S374 RdDat.[8..15]	S151 KReq' Select	S08 SA4SeekComp	Protect Plat Buf51MHz	S38 * Gated5MHz	Clock 51.04 MHz	9
8	a F9401 CRC Gen/Check	b N199 SerialNRZ.[0..7]	c N199 SerialNRZ.[8..15]		e LS273 SrvReq Machine	f * LS273 DelTransferEnb	g RDIV16, 220/330 Testability Pullups	h * S04 Display Testability		8
7	LS273 * WordStatReg	S153 * BNRZWrData	S74 VerifyError, ServiceTrap	S257 * CRCIn,InpData SA1NRZClk	F93453 SrvReqProm	F93453 AddrMk Req AdrMkC.[2,4]	10125 * TTLVideo, TTLVideo'	S04 * Disk Testability	*Fake loc of extra RPack used in PWB*	7
6	LS273 WordStatBuf	LS273 EncoderSync	S86 * Srvce, VerErr, PulseTrap	LS273 Field, Word State Machine	LS273 Addr Mark Recog. Mach.	F93453 AddrMk Req AdrMkC.0,1	S74 WordBoundry'			6
5	S74 IndxFound, BTransfEnb	F93427 Encoder Prom NRZWrDt.[0..3]	S257 Input Multiplex.	F93453 Field, Word State Machine BitCnt.[0,2]	F93453 Field, Word State Machine BitCnt.3,WdBd	S74 SyncXferEnb, SyncSA4Data				5
4	F93427 Encoder Prom UnCompMFM	LS273 Encoder Mach	S74 AddrMkFnd, SectorFound	LS393 DelSeekComp	S74 PulseTrap, SyncRcvMFM	S00 ResetComp, DrvMFMClk	FPLAT Phase Decode Oscillator	FPLAT PumpFeed- Back		4
3	DDU-4-5050 PreComp Dly	S151 PreComp Slct	RDIV16 * 220/330 OpnColl term.	S257 * DkWrDat, SeekComplete	S74 InputEnable ClockEnable	DDU-4-5050 DelInputData	FPLAT Oscillator Bias & Control	FPLAT DCError, PumpBias		3
2	N06 HdSlct[1,2,4,8] DkRdData Term	N06 Sky source, Drve Ctl lines	N14 * CRC Ctl, Trk00, SkCompl,Index	N14 Ready, WrFault SA1/SA4',Sect	S04 * WrEnable', IOPReset	S74 InputArrived ClockArrived	FPLAT Phase Comp. Bias & Filter	FPLAT Phase Comp. Bias & Filter		2
1	75107 DkRdDat, DkRdClk Rcvr.	DConn Data Connector	75114 DkWrtData, DkOutClk Xmit	S38 * ReadGate' WriteGate'	S02 AbortWrite', SkCmpClk	LS393 TimingClock	FPLAT Potentiometer			1
	a	b	c	d	e	f	g	h	i	

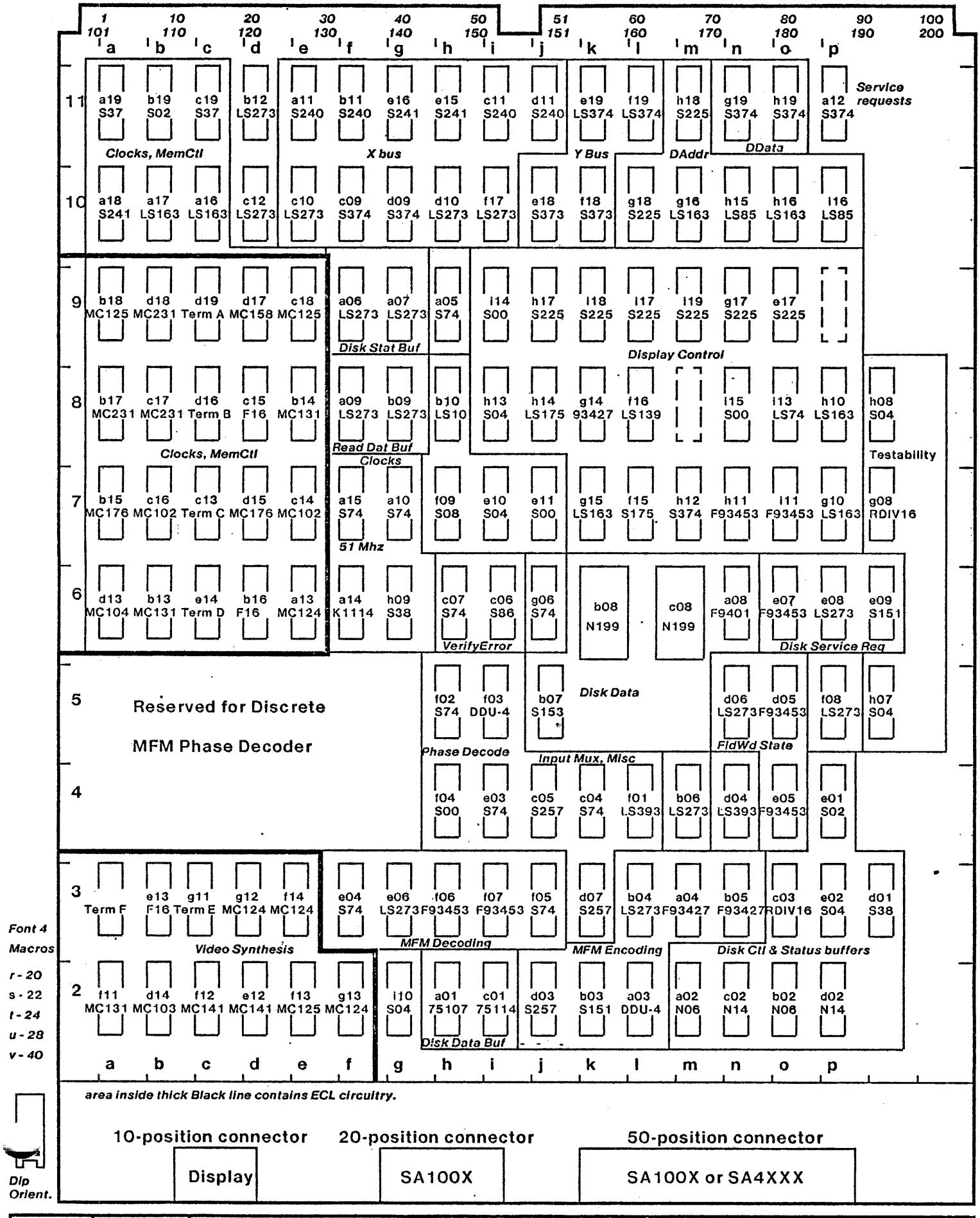
DIP Orient. 1

Area above medium width line used for display, clocks. Below is Disk controller. a12, e11, e10 and g8 are shared.

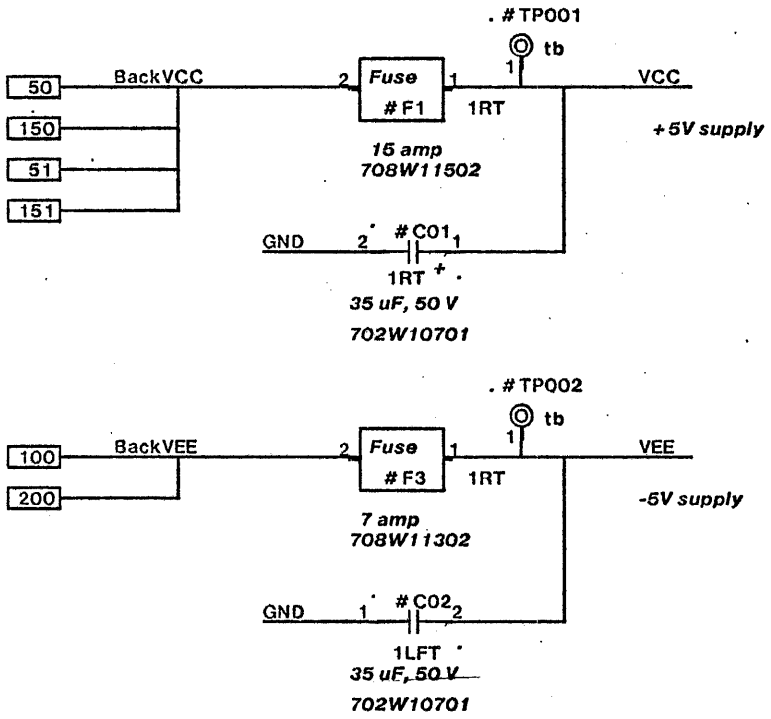
I/O Connector Area (Top)

I/O Connector Area (Bottom)

XEROX EOD	Project: Dandelion	Reference: High Speed I/O Board Layout	File: HSIO07.sily	Designer: Crane, Davies	Rev: Q	Date: 7/23/80



XEROX SDD	Project	Reference	File	Designer	Rev	Date
	Dandelion	HSIO module PC layout detail	HSIO08.sily	Davies	Q	7/16/80



XEPOX SDD	Project		File	Designer	Rev	Date	Page
	Dandelion	Power Supply and Fuses	pHSI001.sil	Ogus, Davies	Q	7/23/80	01

PWA Layout Considerations

1. ECL wiring should be on the chip side of the ground plane in the ECL area. TTL wiring should be placed on the other (back) side of the ground plane in the ECL part of the board.

2. The ECL Clock signal 51MHz should be driven from the middle and terminated on each end with 100 ohms to - 2 volts.

3. The impedance of the traces used for ECL wiring should have an impedance of approximately 100 ohms.

4. The system clock outputs from each board should be checked as part of the manufactu ing process to assure that there is no timing skew between the clocks. This should certainly be done on early production boards. Continuation of tests on later batches can be determined by the results of testing the early batches of boards.

5. Do not change the allocation of MC124 gates. The InhibitRead' signal is used with the common enable on one of the packages.

6. The maximum voltage induced on any ECL trace from another ECL or TTL trace should be 0.1 V.

7. RAS' and CAS should go through gates located in the same MC124 and 74S37 packages.

8. LRAS' and LCAS should go through gates located in the same 10231, 10124, and 74S37 packages.

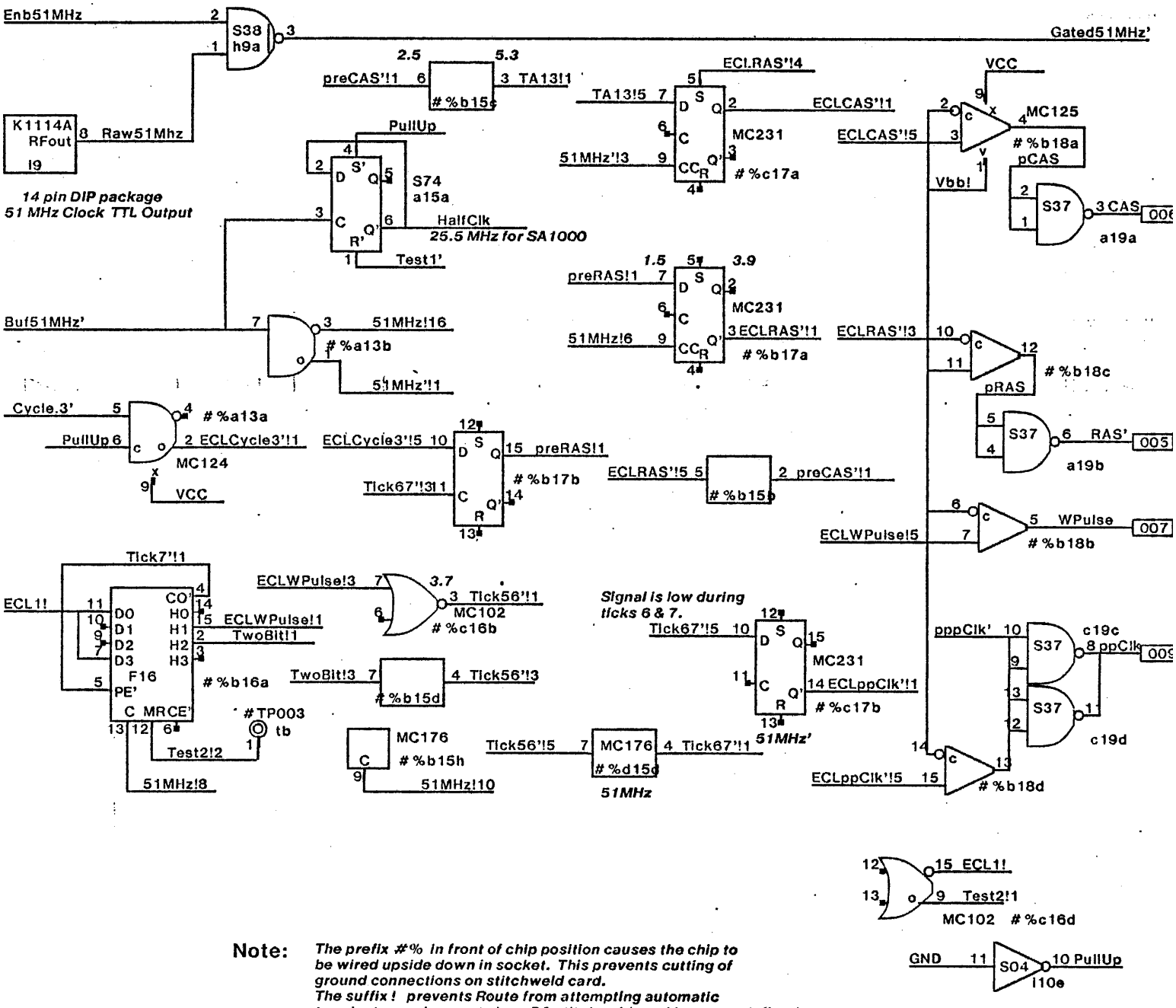
9. Voltages used on this board: +5 V, -5.2V

10. All locations to be provided with IC sockets.

11. lengths of the X, Y, DData and DAddr busses on this board should be minimized.

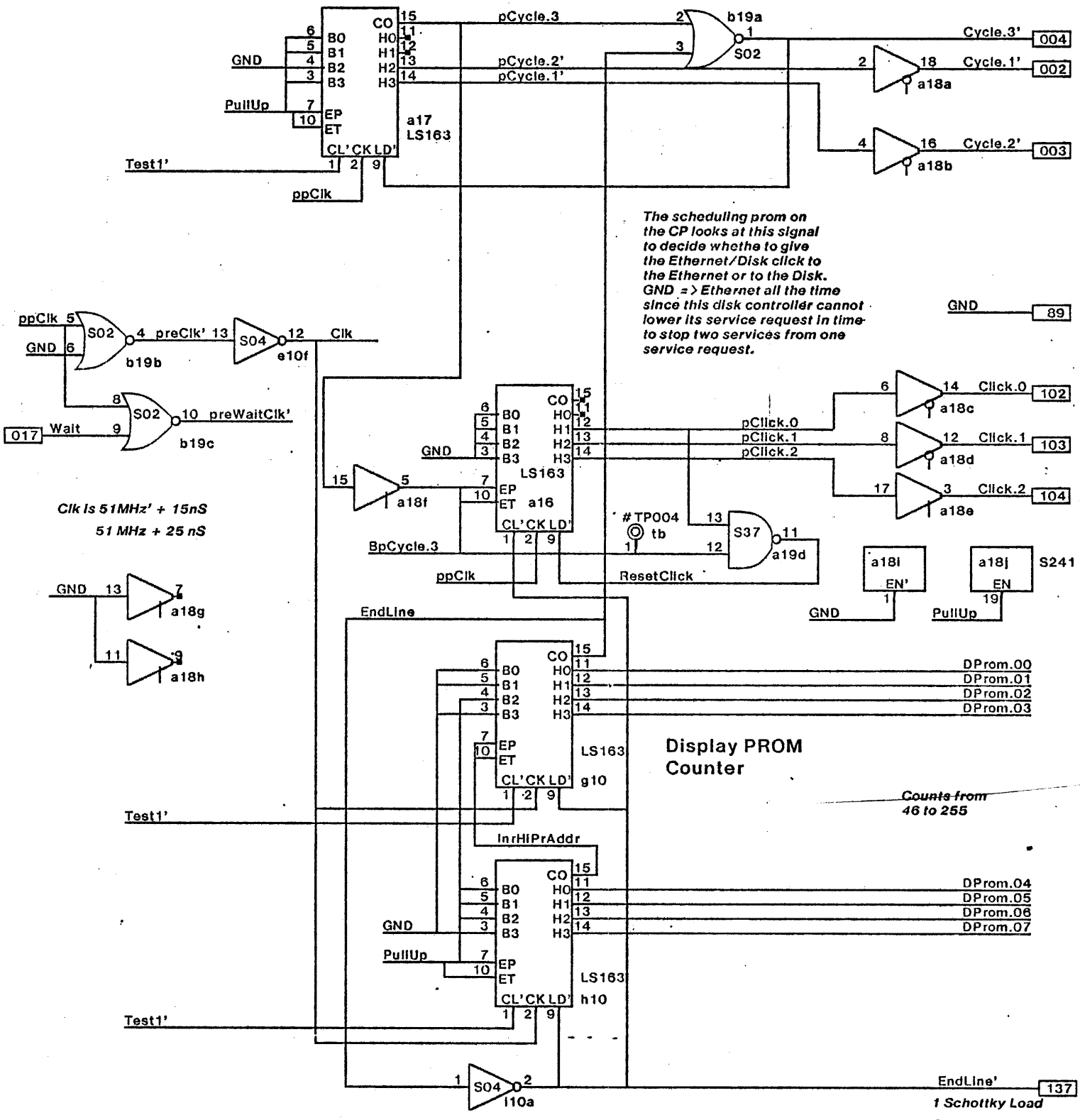
12. Care should be taken to isolate the analog circuits in the phase decoder from noise. All analog grounds should be tied together and connected to the logic ground at one point. Similar care should be taken with the VCC and VEE signals used by the analog circuits.

13. The Open-collector drivers and Schmitt trigger recievers used to buffer the disk cable signal should be placed close to the disk cable connectors.

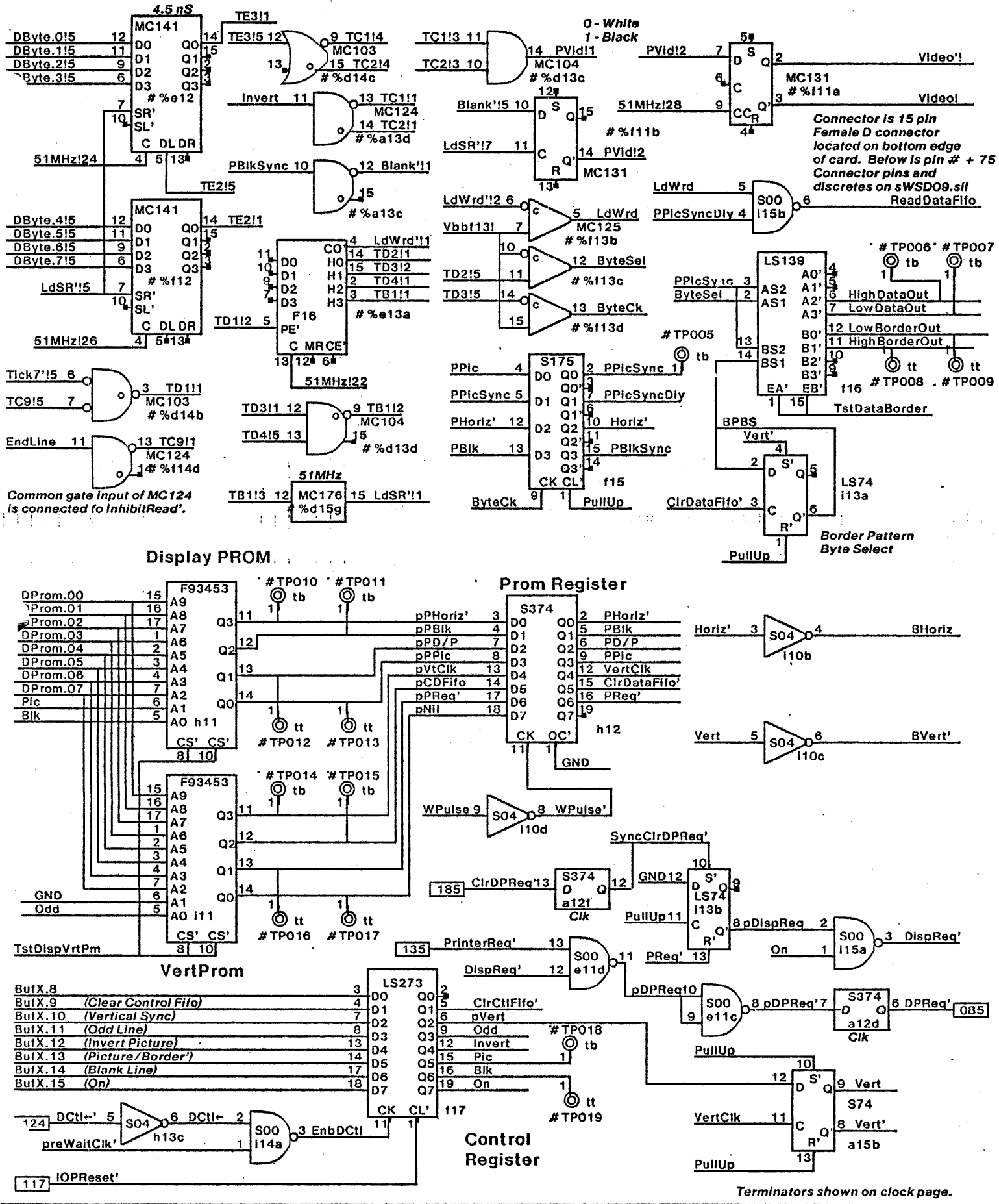


Note: *The prefix #% in front of chip position causes the chip to be wired upside down in socket. This prevents cutting of ground connections on stitchweld card.*
The suffix ! prevents Route from attempting automatic terminator assignment since DO stitchweld card has none defined. Subnet wiring order for a net is done by appending to the net name a ! followed by the wiring sequence number of the node in the net. Automatic terminator assignment is inhibited by use of ! as the last character in the character string of the net. This must occur after the subnet feature if it is also being used.

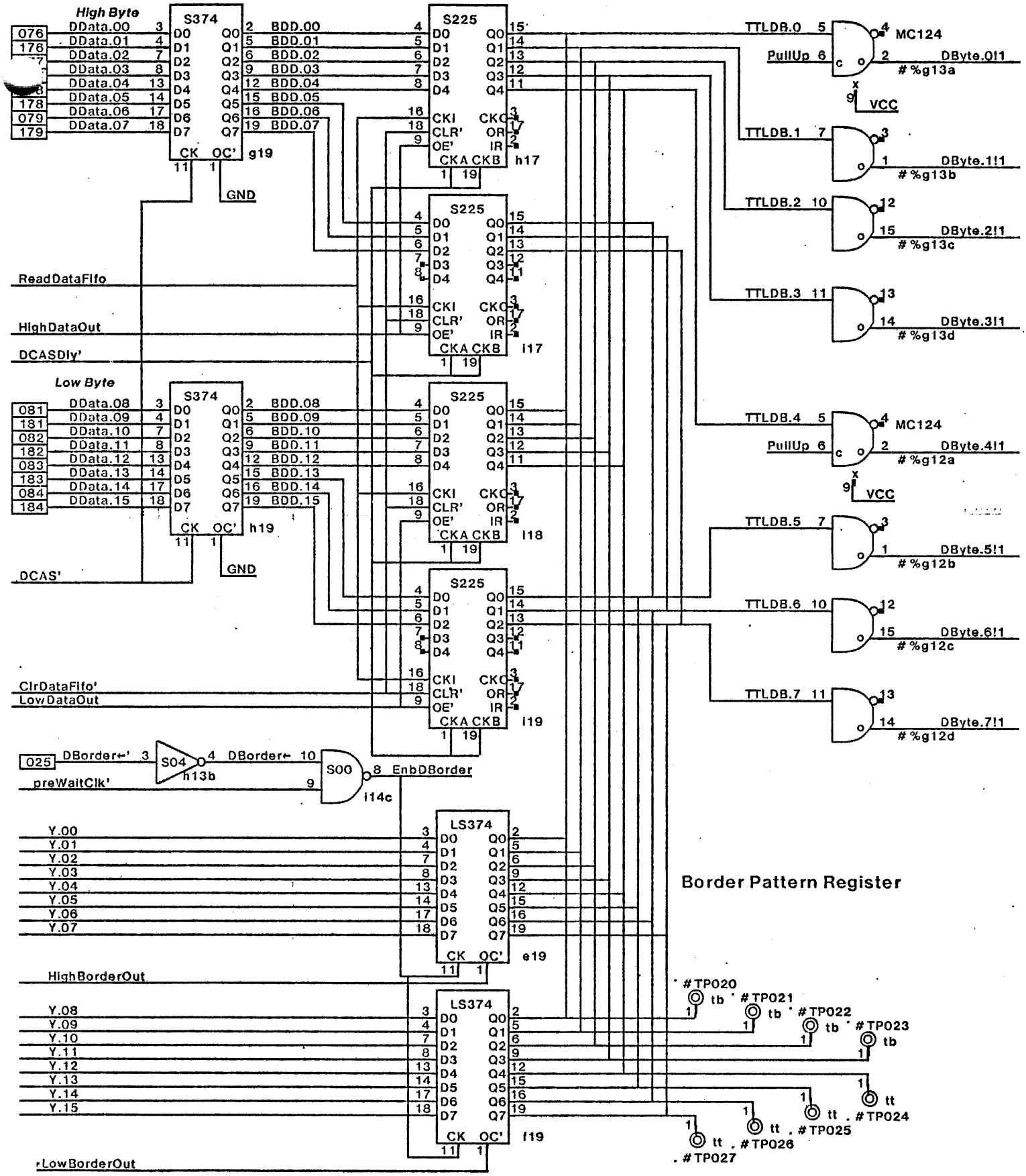
XEROX SDD	Project WS	51 MHz Clock Dividers	File HSI022.sil	Designer Crane	Rev Q	Date 7/14/80	Page 22



XEROX SDD	Project WS	Cycles, Clicks, & Display Counter	File HSIO23.sil	Designer Crane	Rev Q	Date 6/21/80	Page 23

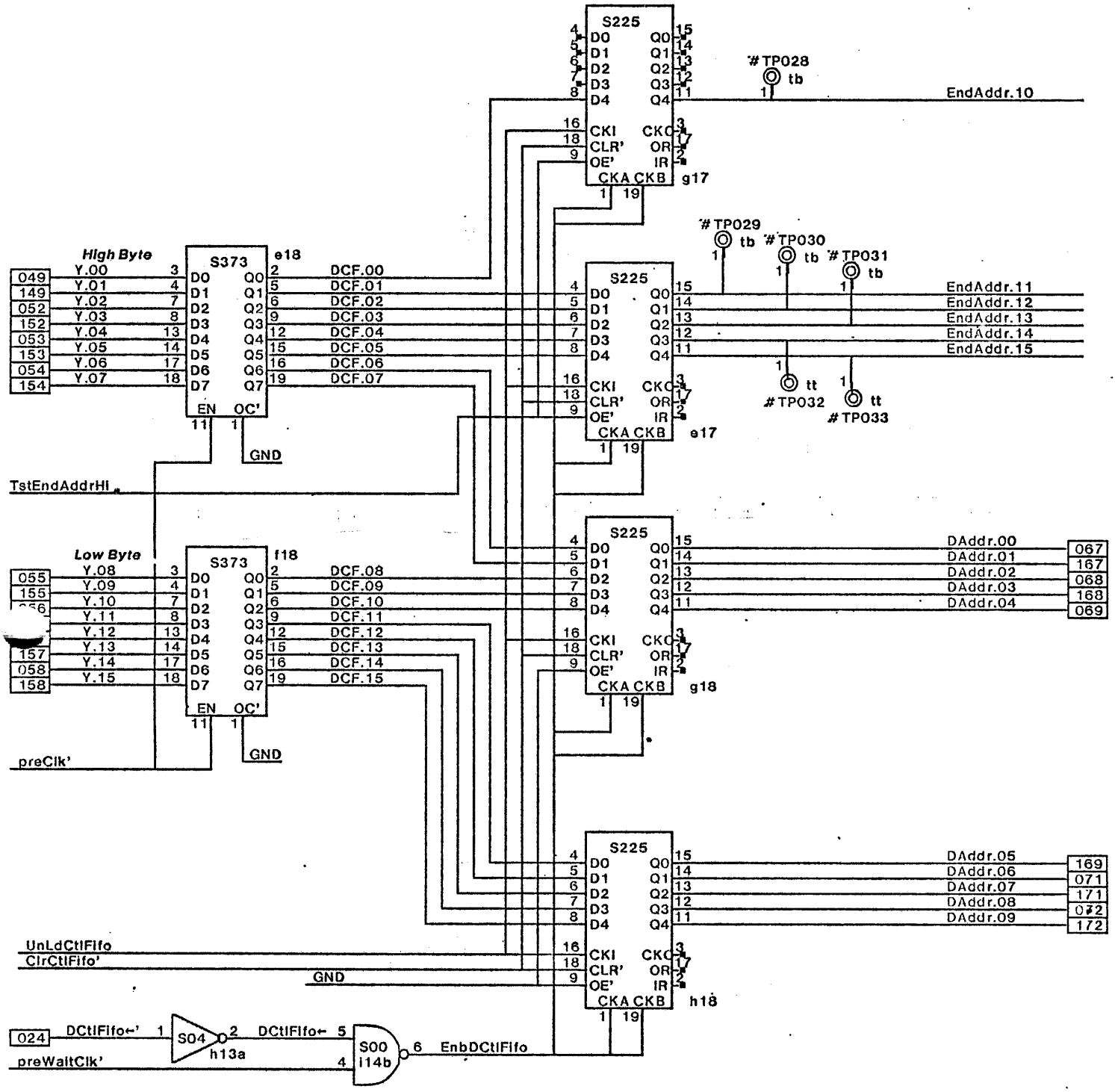


XEROX SDD	Project WS	Display Output Machine and Control Register	File HSIO24.sil	Designer Crane, Davies	Rev Q	Date 6/21/80	Page 24

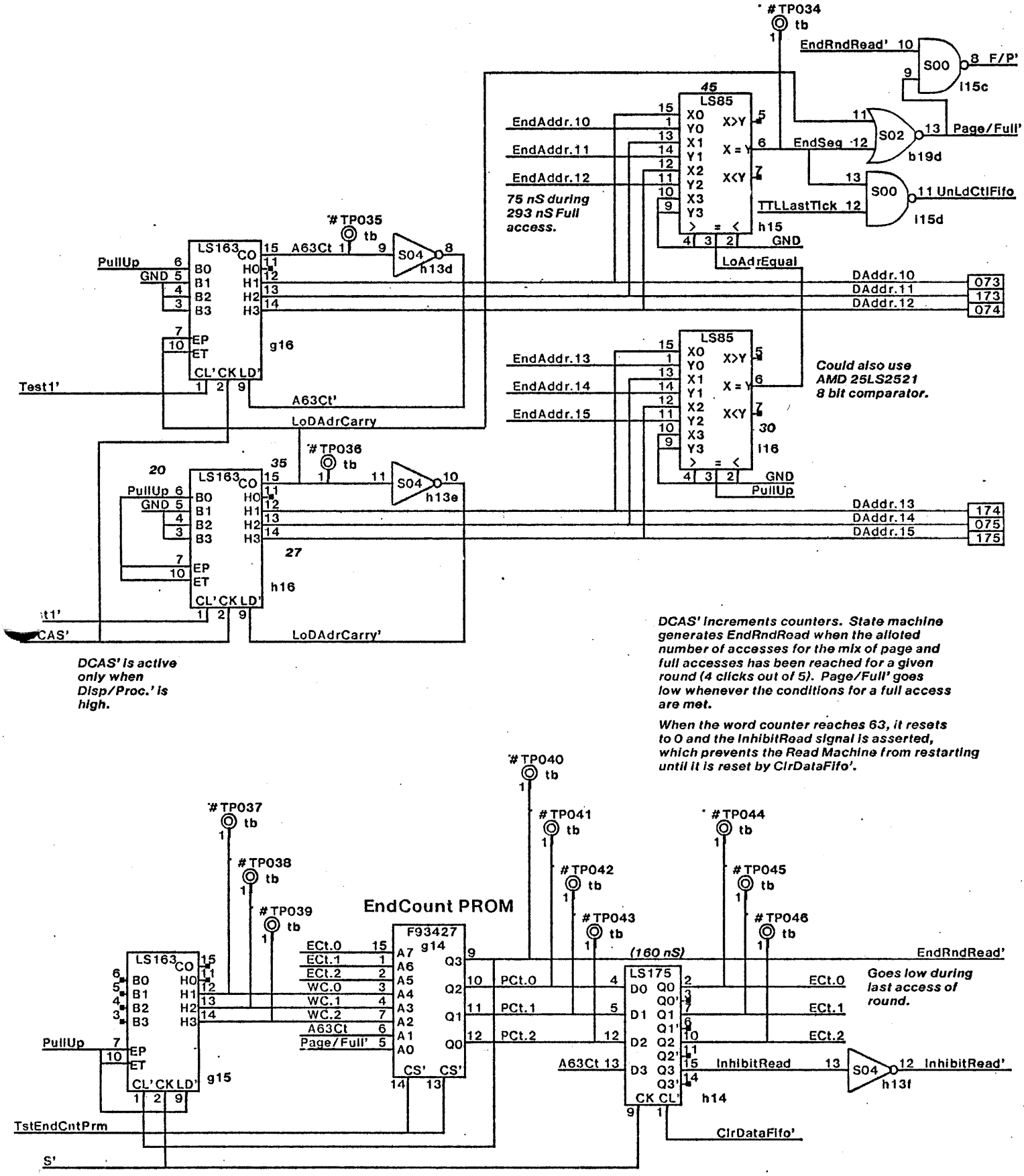


EndLine goes low once per horizontal line.
DCASDly' is DCAS' delayed by 20 nS.
Read signal goes low for 20 nS before low data byte is latched by the shift register.

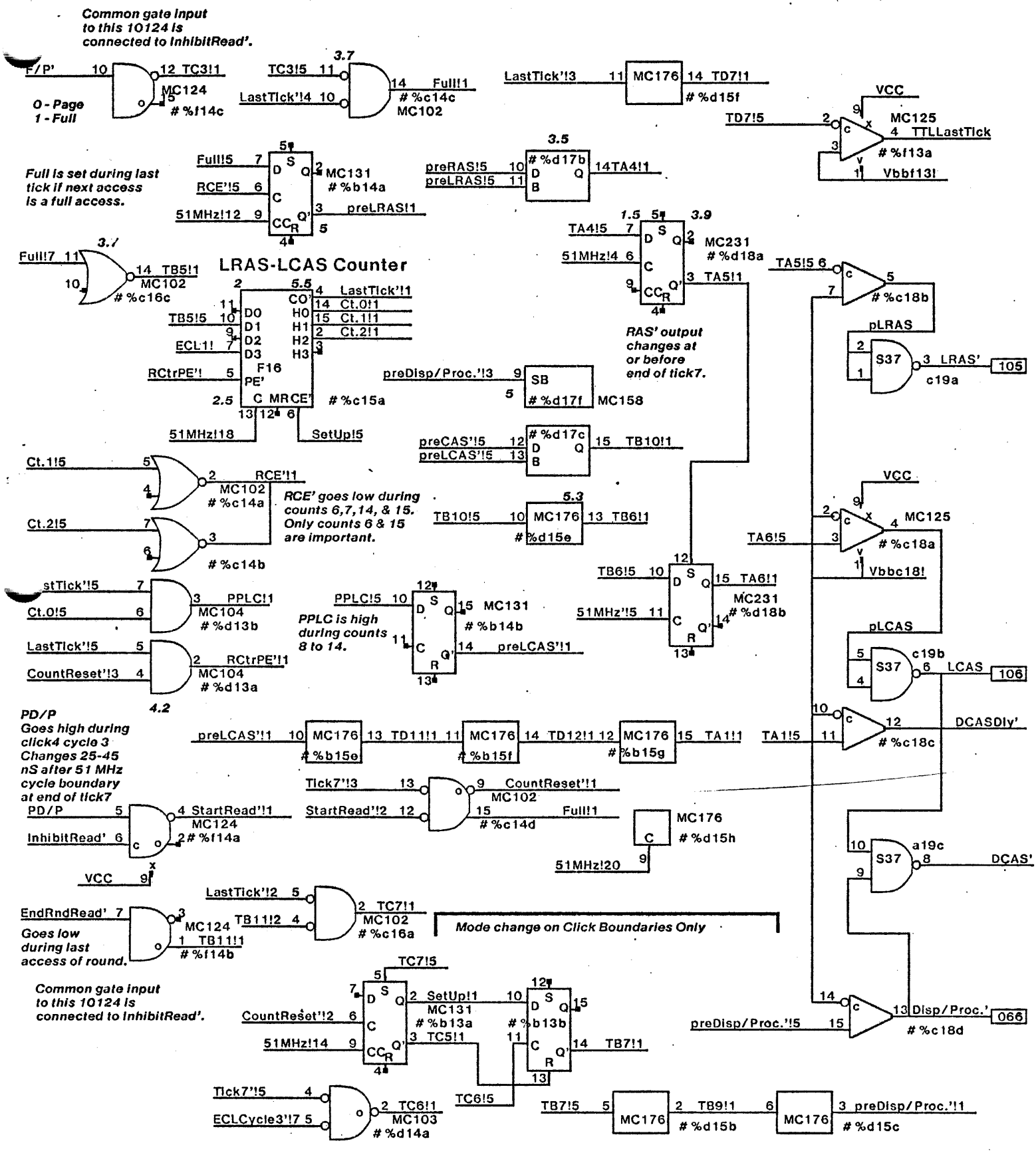
XEROX SDD	Project WS	Data FIFO and Border Register	File HSIO25.sil	Designer Crane	Rev Q	Date 6/21/80	Page 25



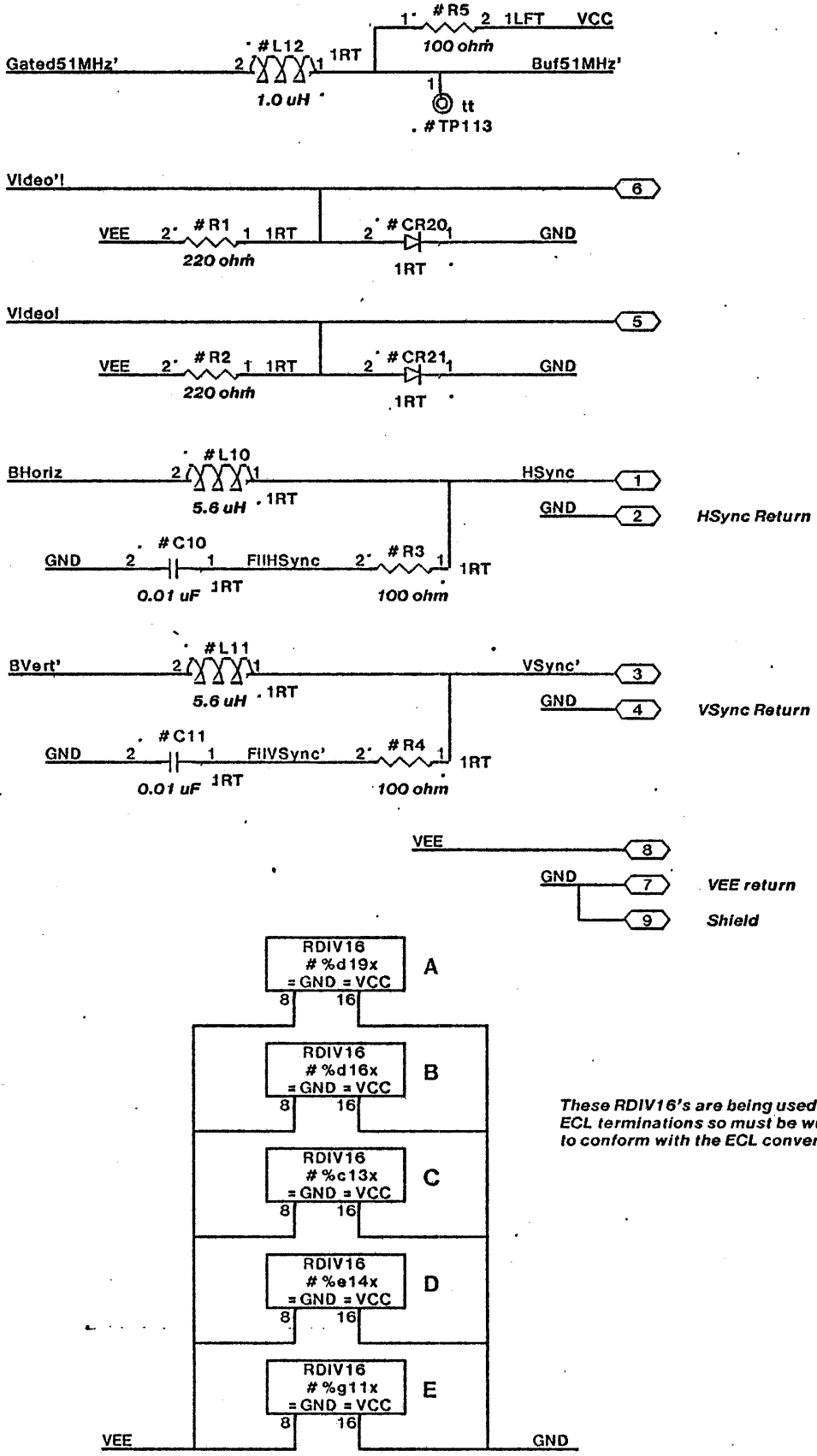
XEROX SDD	Project WS	Control FIFO Data Path	File HSIO26.sil	Designer Crane	Rev Q	Date 6/21/80	Page 26



XEROX SDD	Project WS	Read Machine: Word Counter & End Conditions	File HSIO27.sil	Designer Crane	Rev Q	Date 6/21/80	Page 27



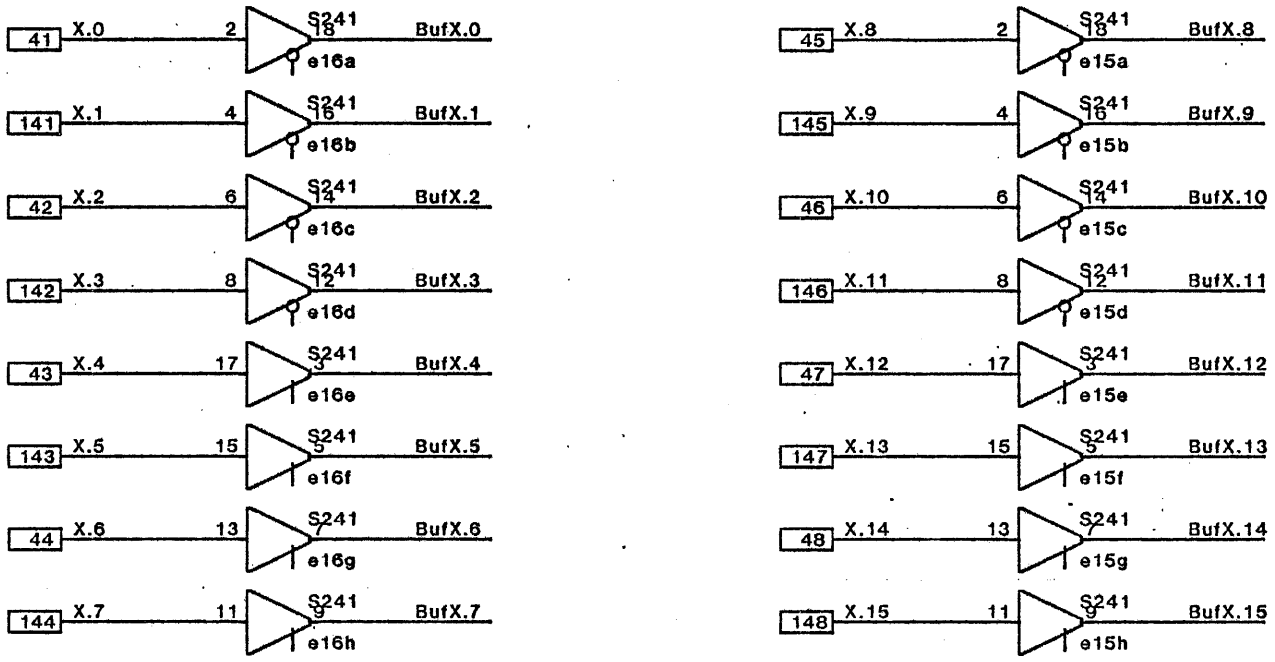
XEROX SDD	Project WS	LCAS & LRAS' Generation	File HSI028.sil	Designer Crane	Rev Q	Date 5/30/80	Page 28

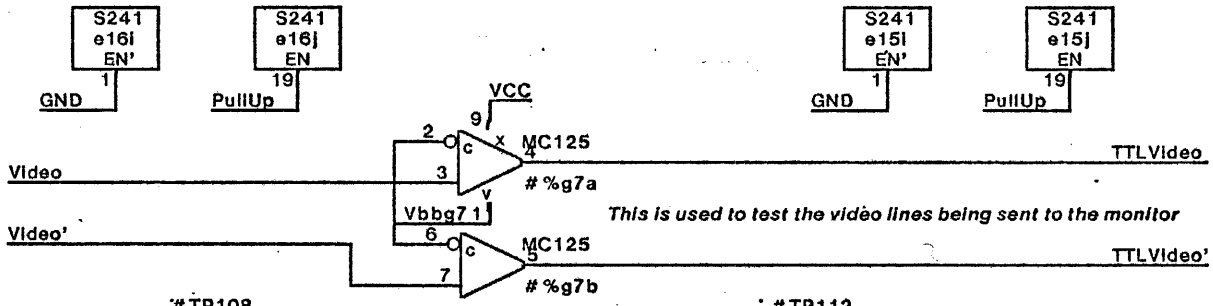


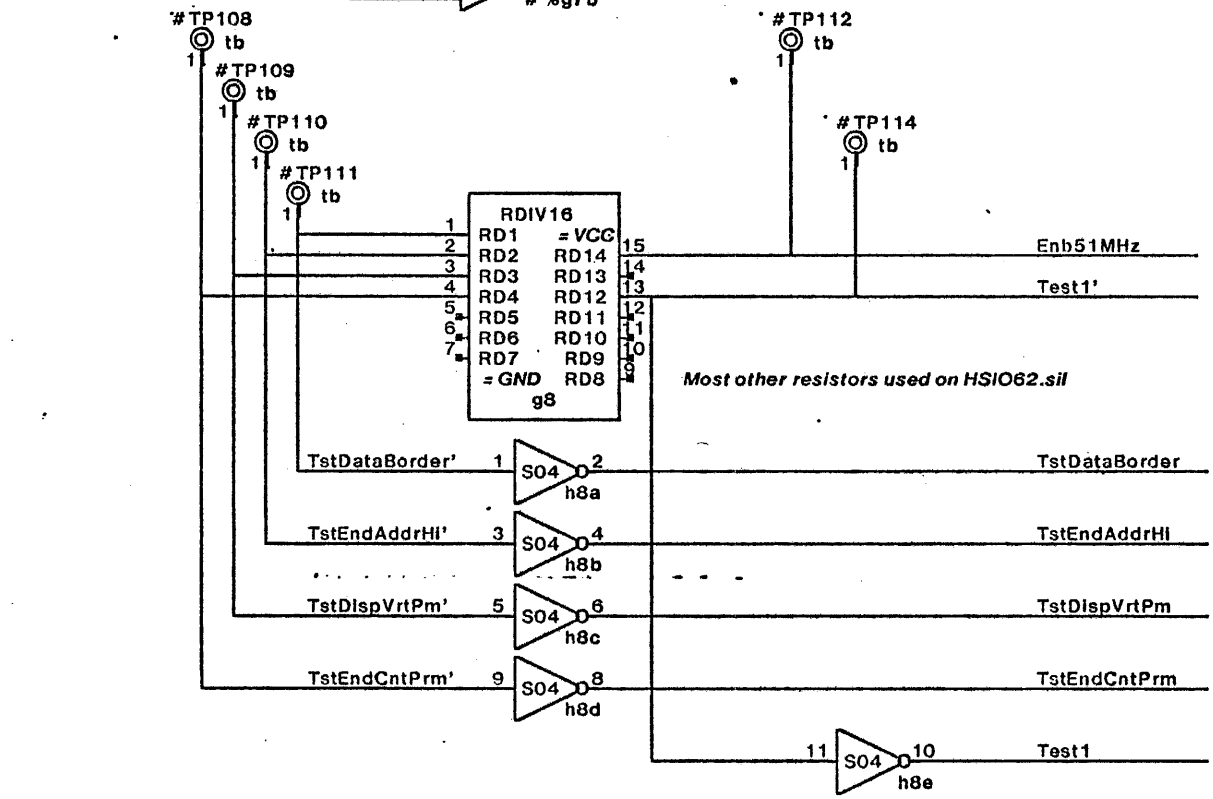
These RDIV16's are being used as ECL terminations so must be wired to conform with the ECL conventions.

XEROX SDD	Project: Dandelion	Dandelion Display Controller: Discretes, Connectors for PWB Controller	File: pHSIO29.sil	Designer: Crane, Davies	Rev: Q	Date: 6/23/80	Page: 29

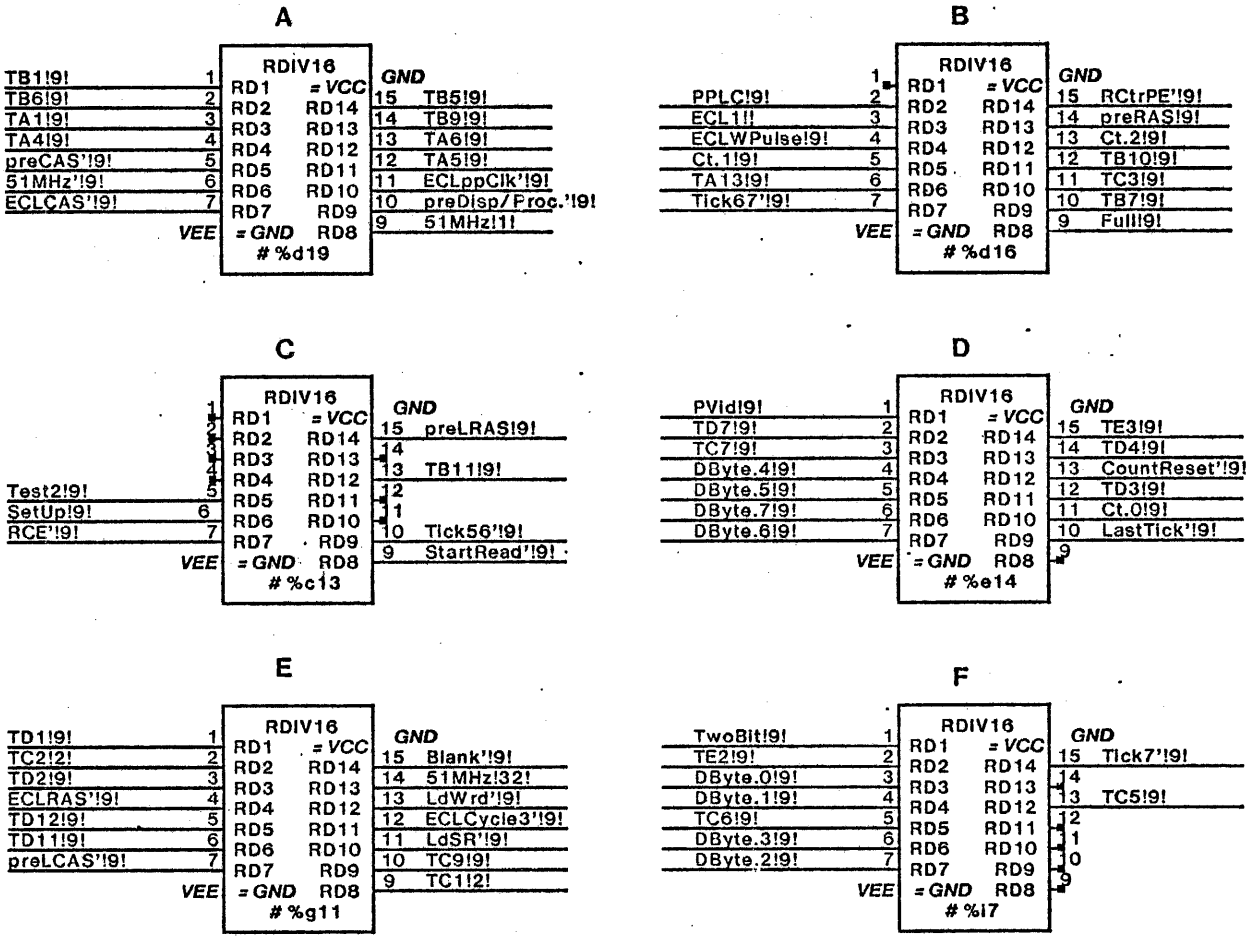
Buffer X bus to reduce loading.







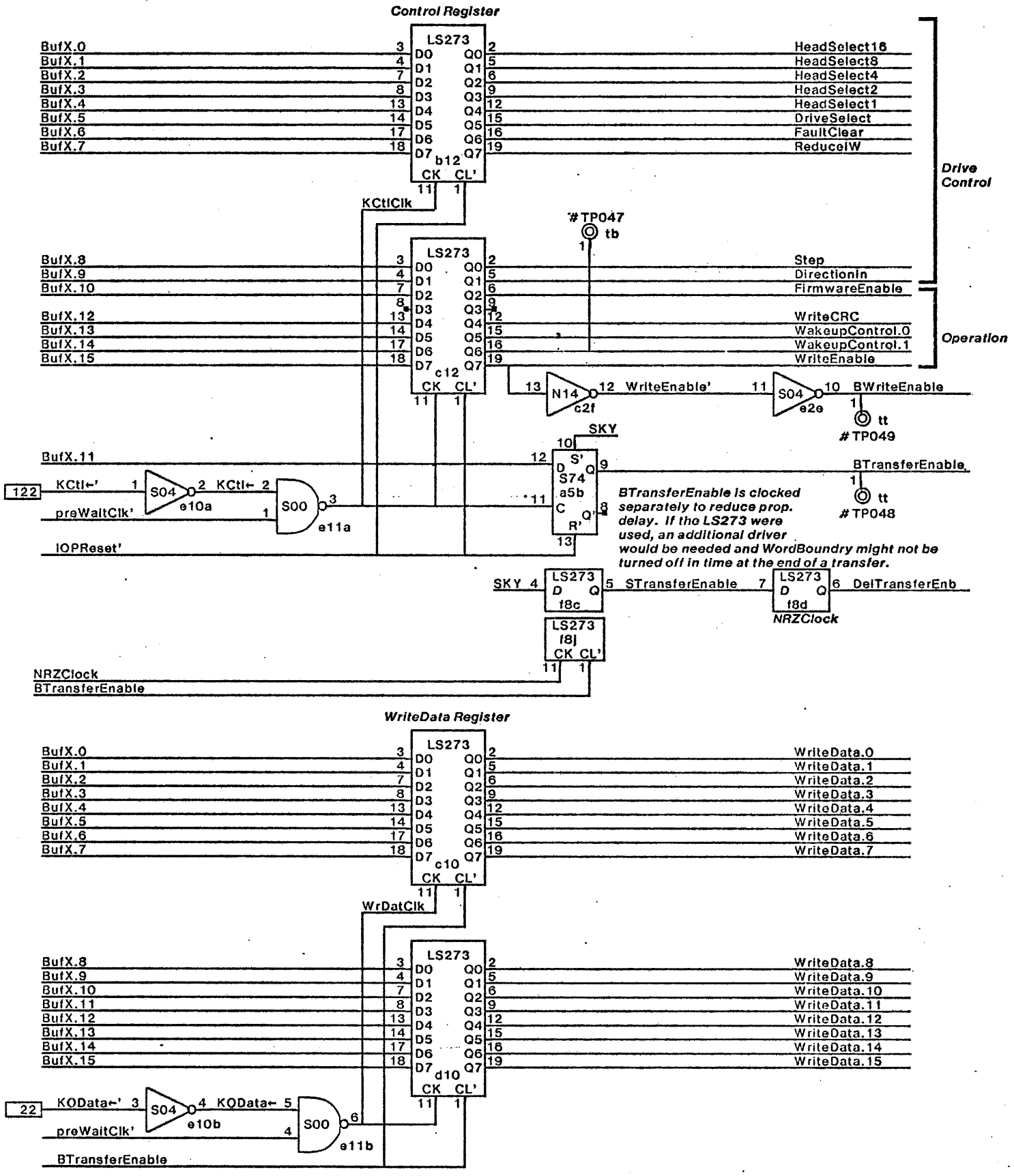
XEROX SDD	Project: Dandelion	Testability Signals	File: HSIO30.sil	Designer: Davies	Rev: Q	Date: 6/26/80	Page: 30



Termination Packages A, B, C, D, E above are 100 ohm termination to -2 V
Allen-Bradley part no. 316E161261

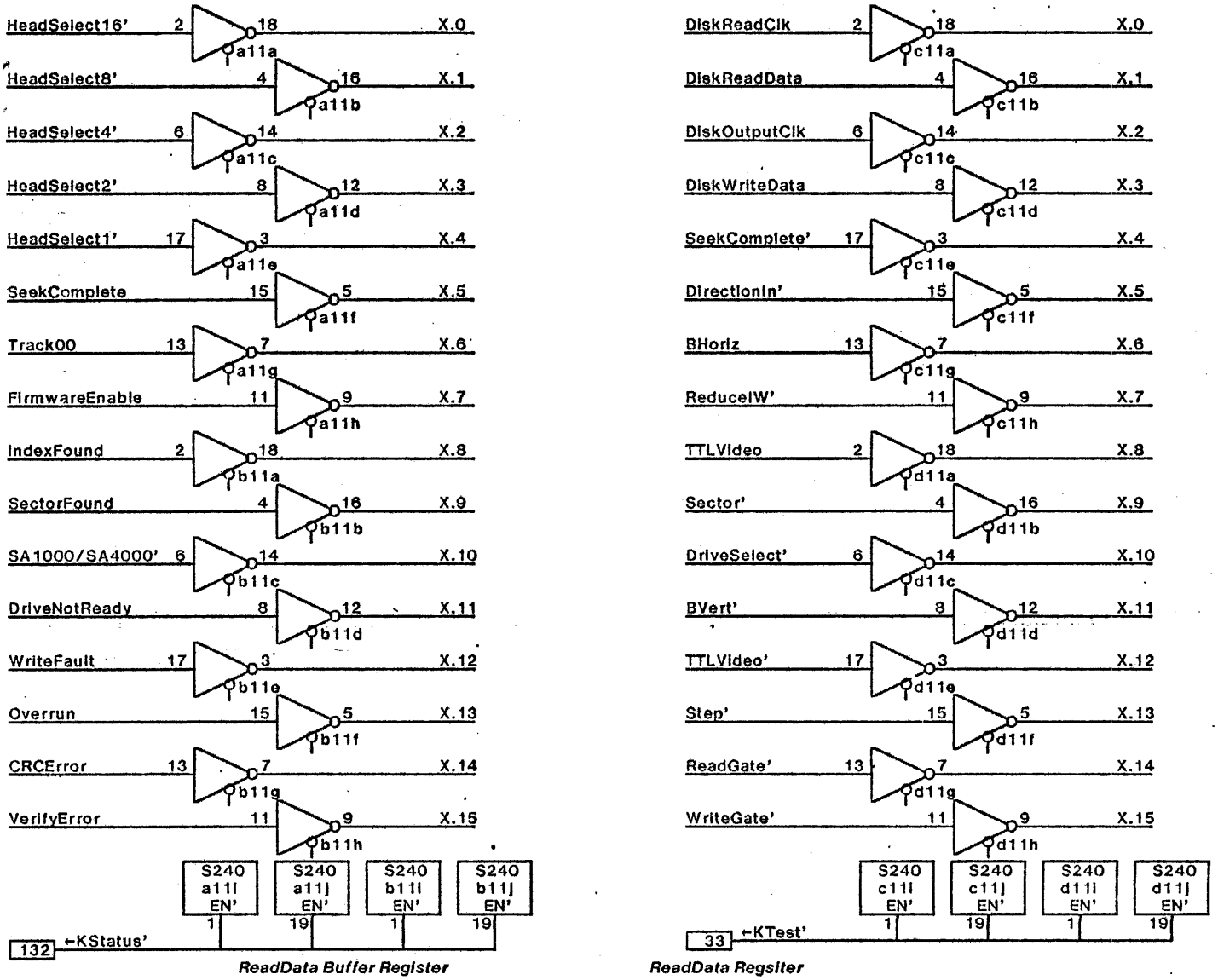
Pin 16 on each termination package is connected to GND and Pin 8 to VEE (-5.2 V). This is done on pWSD09.sil and sWSD09.sil where there is more room. This connection make the termination compatible with normal ECL power rules.

Note: *The prefix #% in front of chip position causes the chip to be wired upside down in socket. This prevents cutting of ground connections on stitchweld card.*
The suffix ! prevents Route from attempting automatic terminator assignment since DO stitchweld card has none defined. Subnet wiring order for a net is done by appending to the net name a ! followed by the wiring sequence number of the node in the net. Automatic terminator assignment is inhibited by use of ! as the last character in the character string of the net. This must occur after the subnet feature if it is also being used.



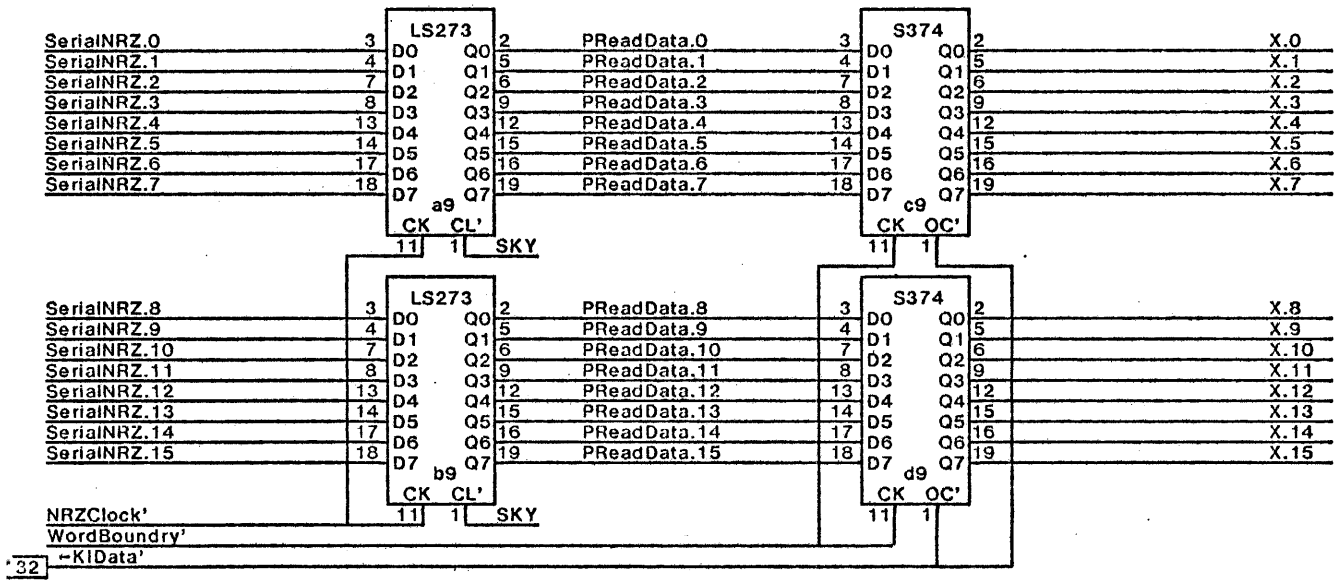
XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Control and Write Data Regs.	File: HSIO47.sil	Designer: Davies	Rev: Q	Date: 7/14/80	Page: 47

Status/Test Multiplexer

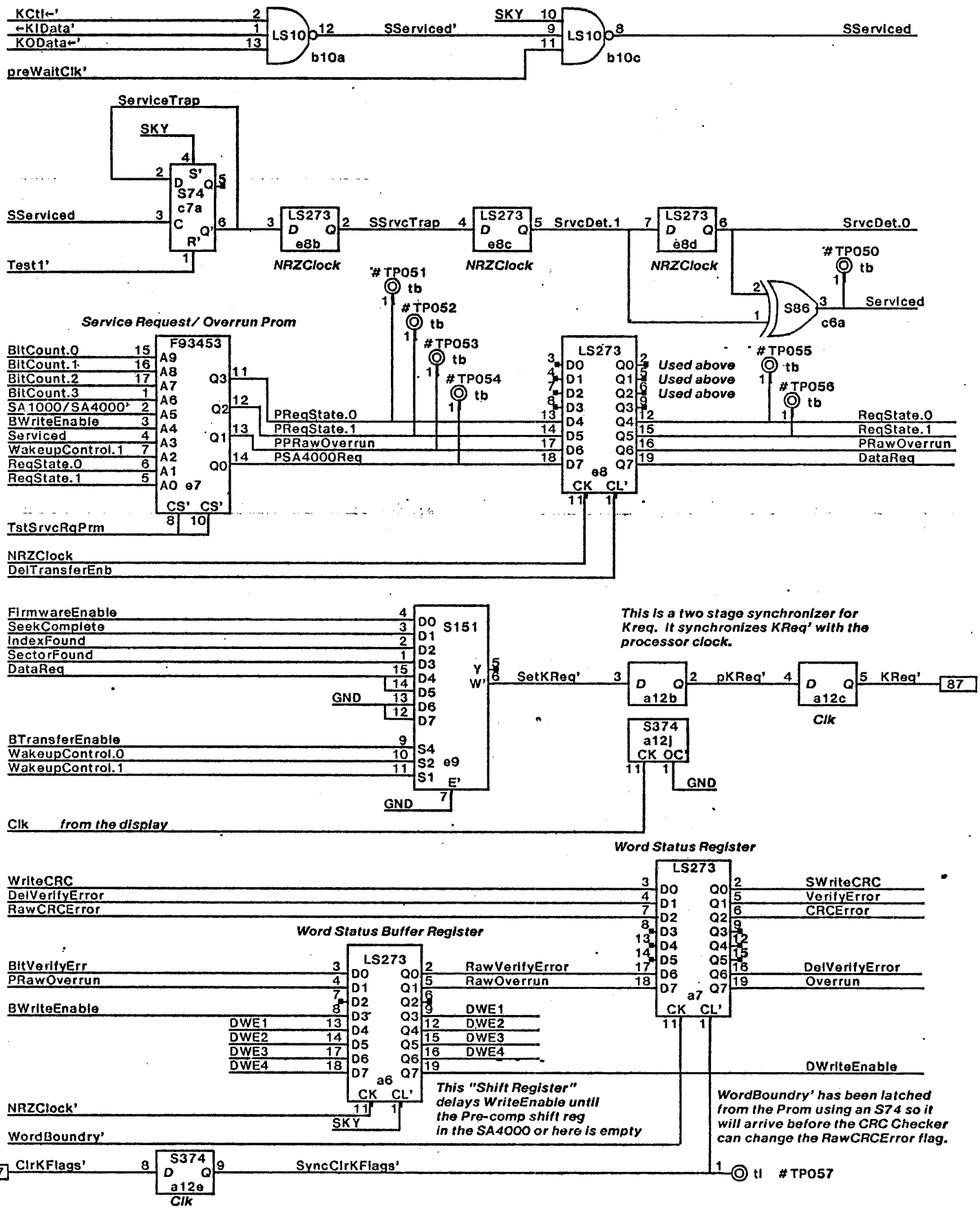


ReadData Buffer Register

ReadData Regsiter

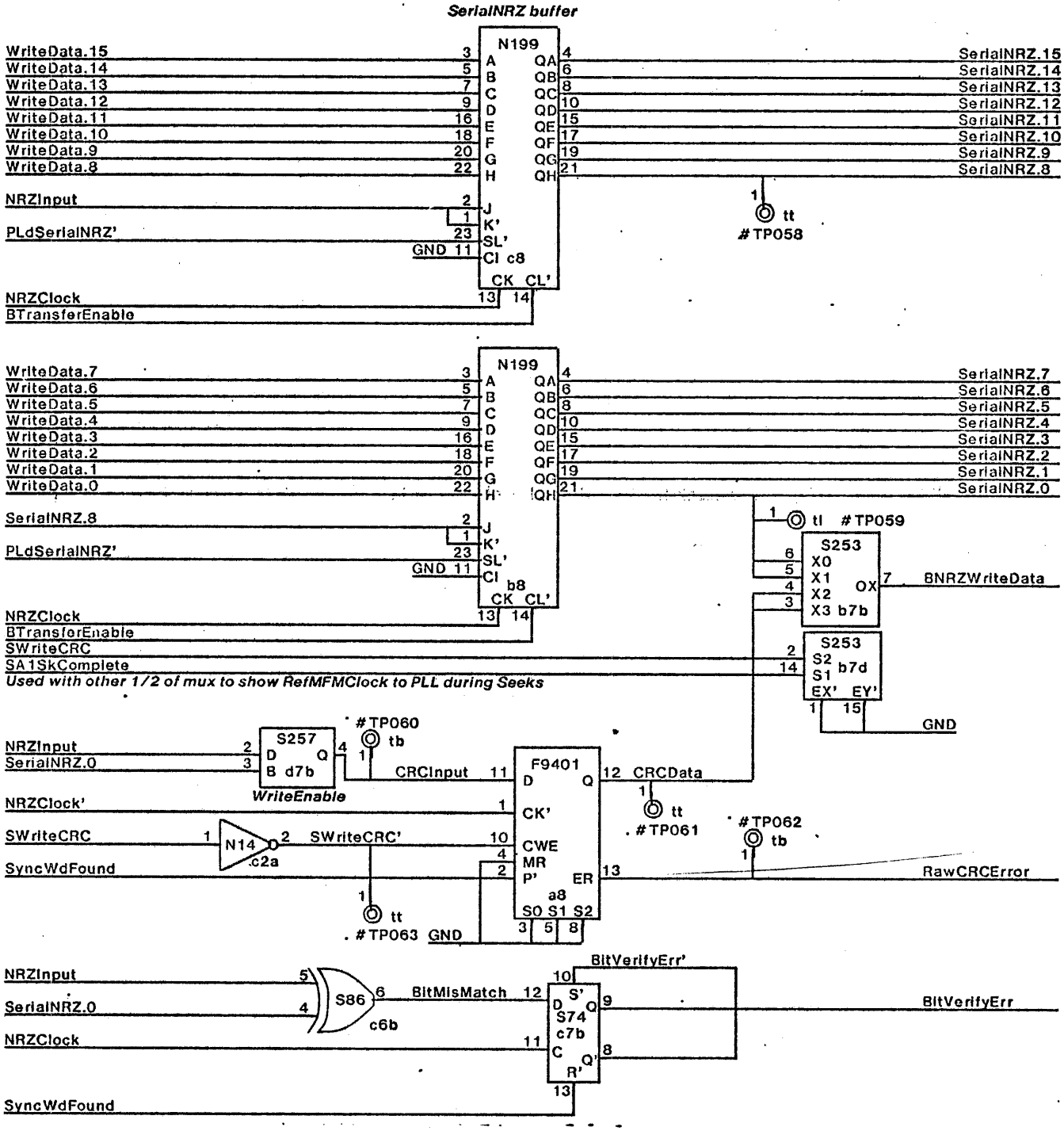


XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Status/Test Mux, ReadData Reg	HSIO48.sil	Davies	Q	6/25/80	48

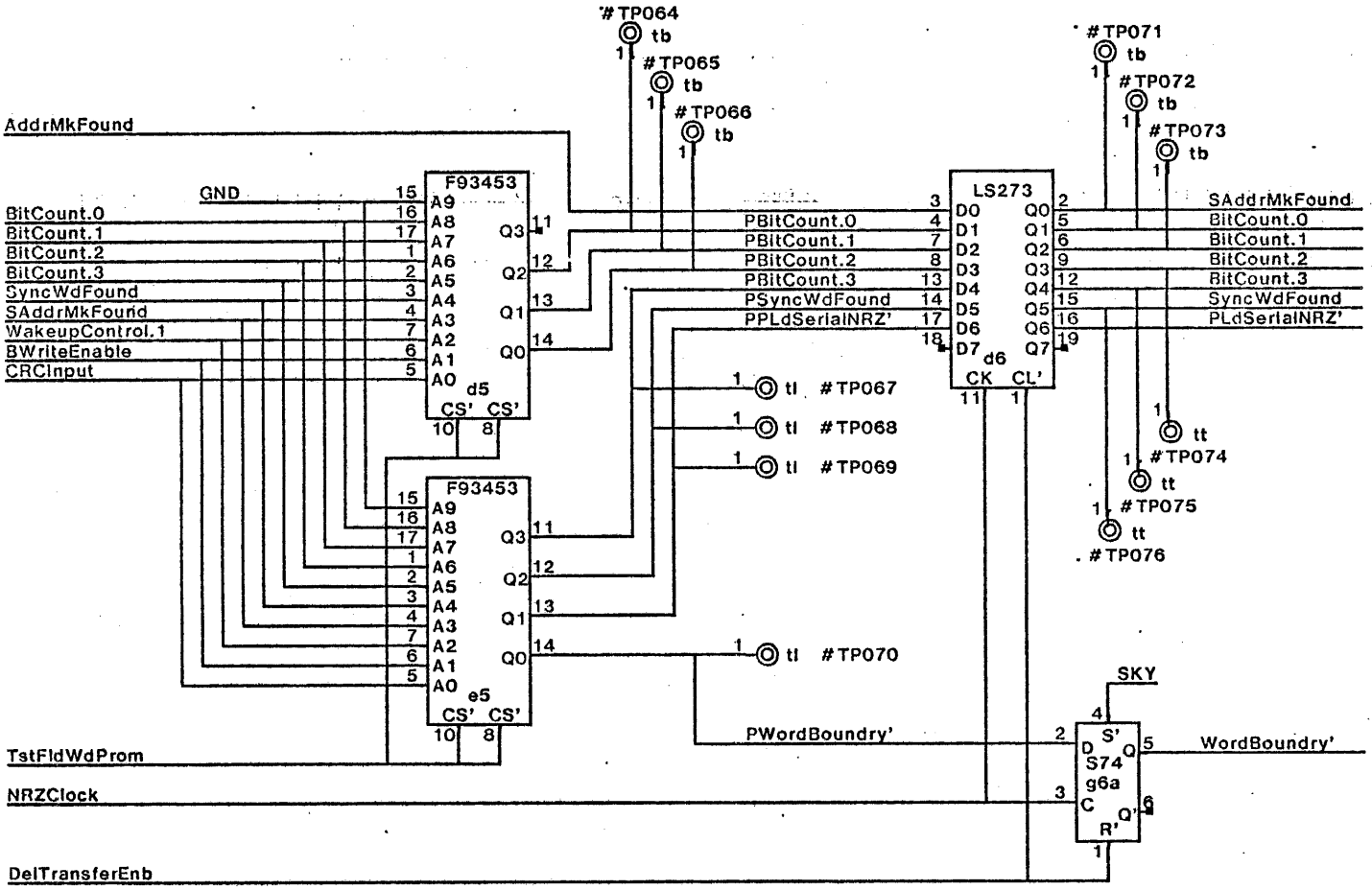


XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Srv. Req, Overrun, Word Stat.	HSI049.sil	Davies	Q	7/14/80	49

SerialNRZ Shift Register and Error Checking

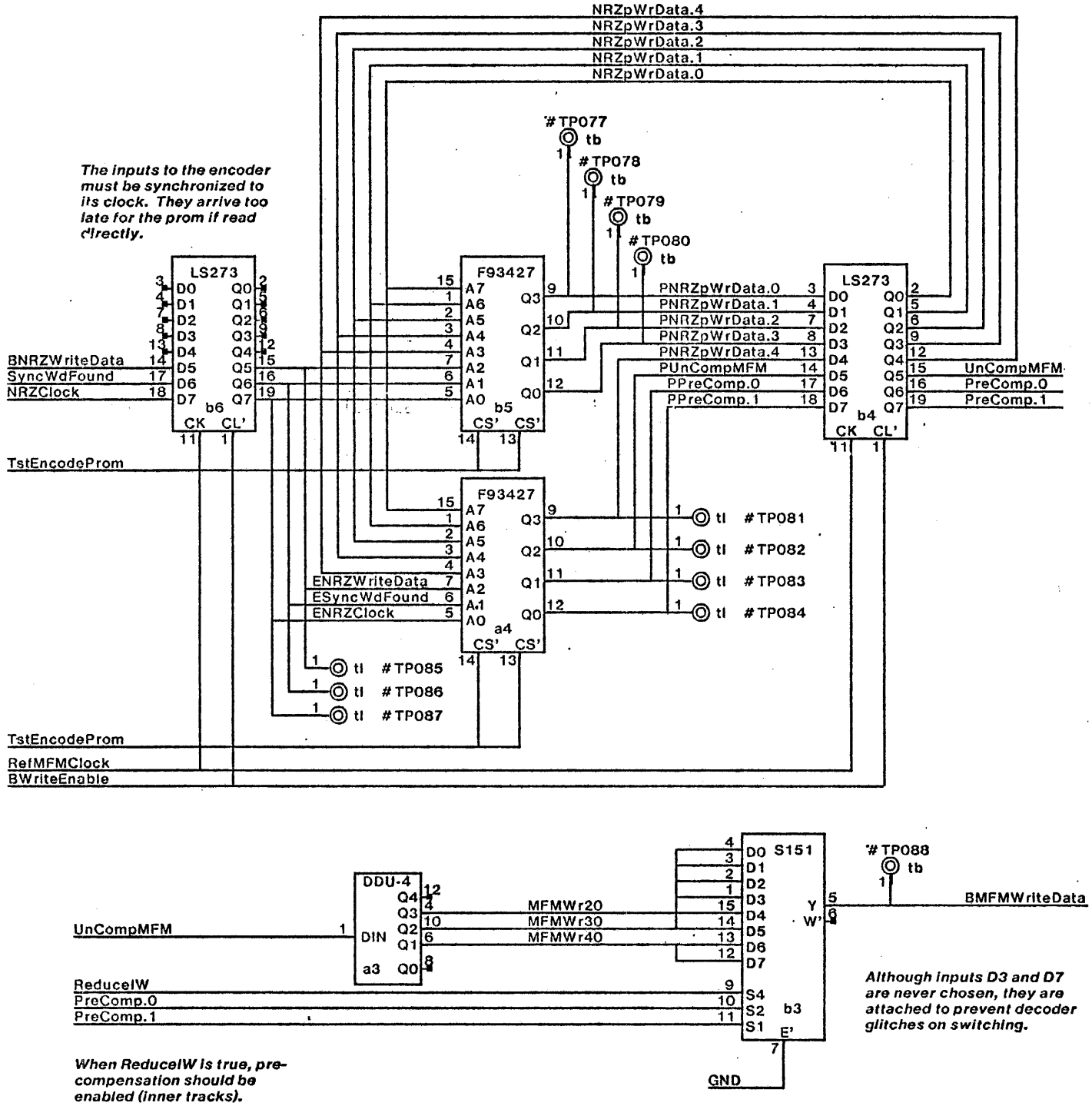


XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Serializer/DeSerializer	File: HSIO50.sil	Designer: Davies	Rev: Q	Date: 7/23/80	Page: 50

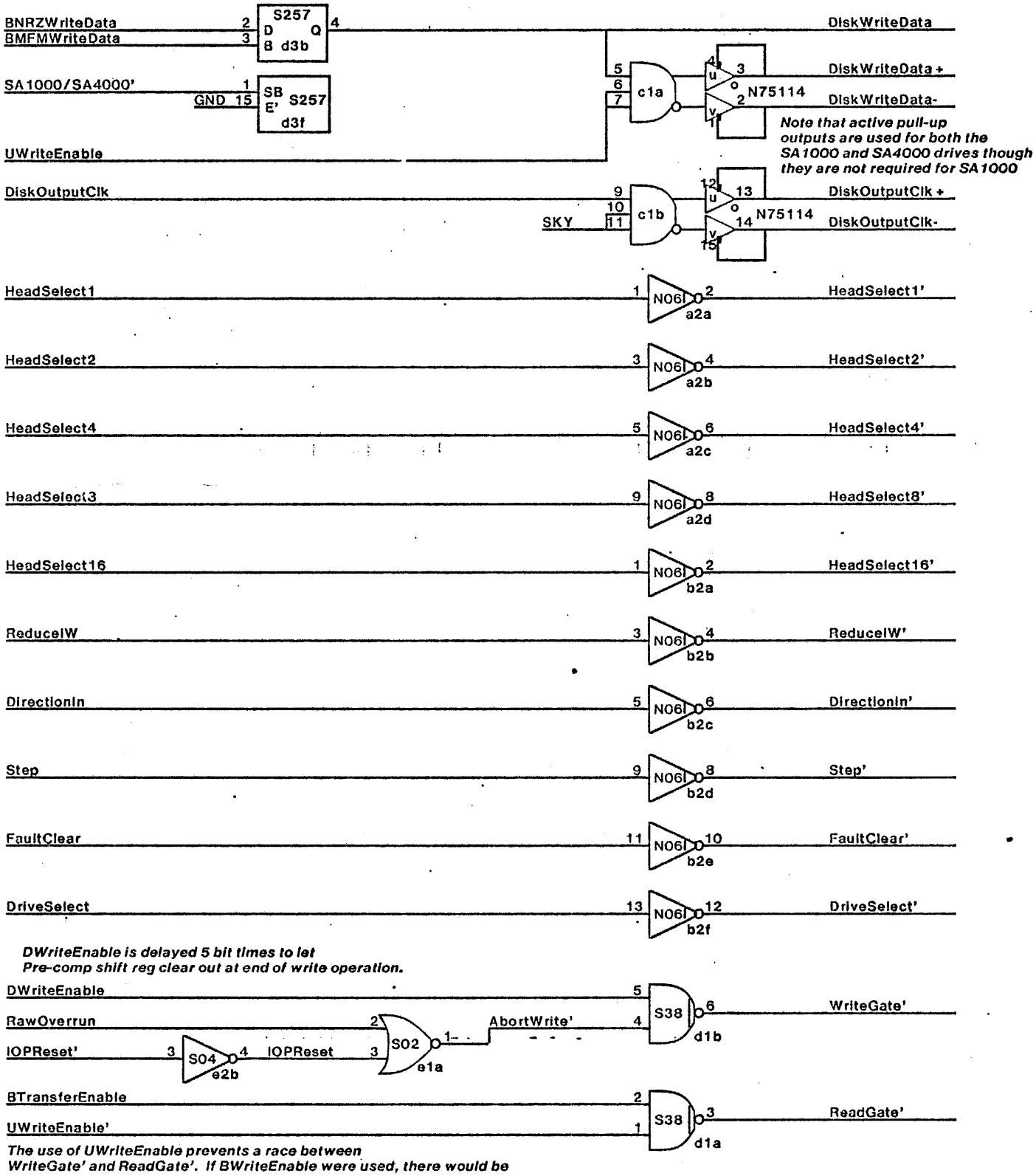


Using an S74 instead of the LS273 speeds up WordBoundry' so it will change before the RawCRCError indicator from the 9401 CRC Checker. This allows us to latch the CRCError signal directly using WordBoundry'. The RawCRCError signal is too slow to latch into the Word Status buffer register using NRZClock'. There is then a race between WordBoundry' and RawCRCError after NRZClock rises. Using the faster S74 here ensures WordBoundry' wins.

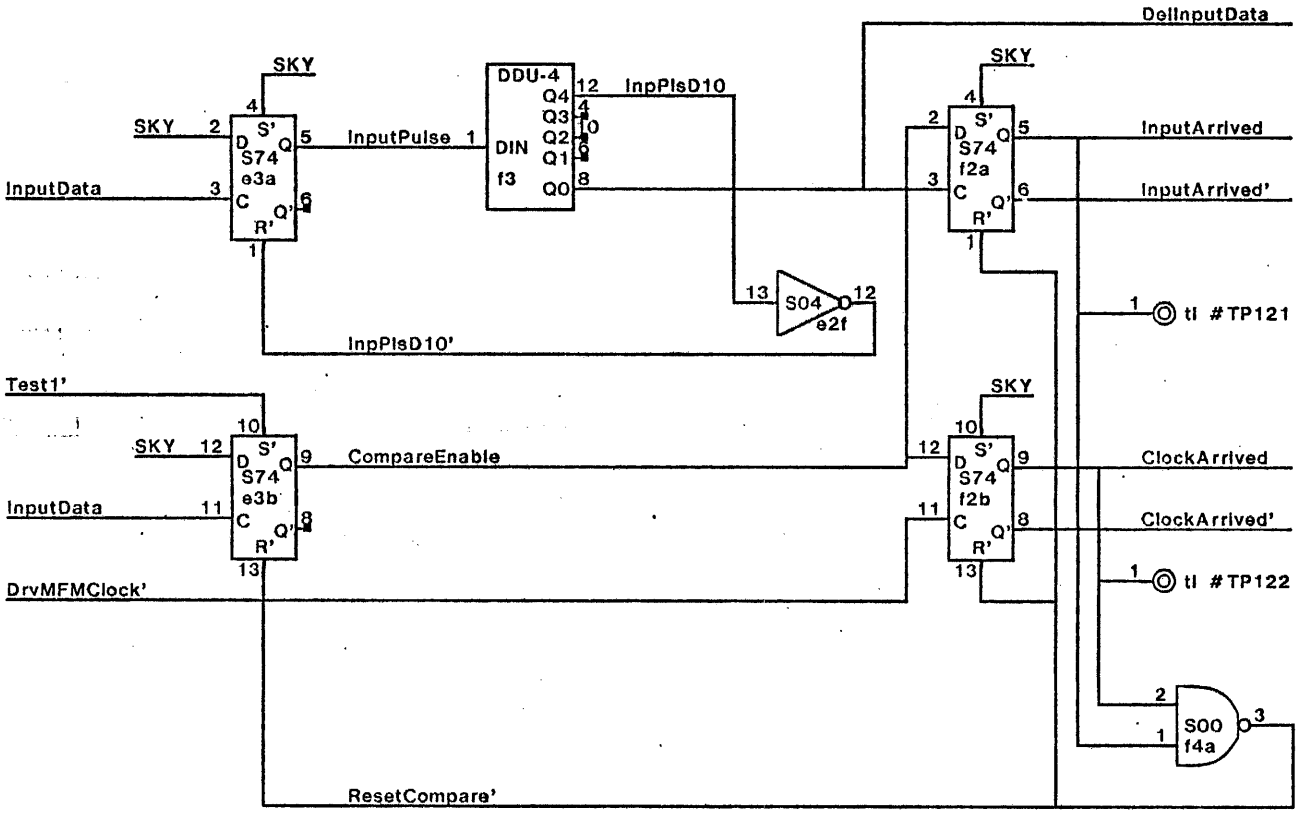
XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Field and Word State Machine	File: HSIO51.sil	Designer: Davies	Rev: Q	Date: 7/14/80	Page: 51



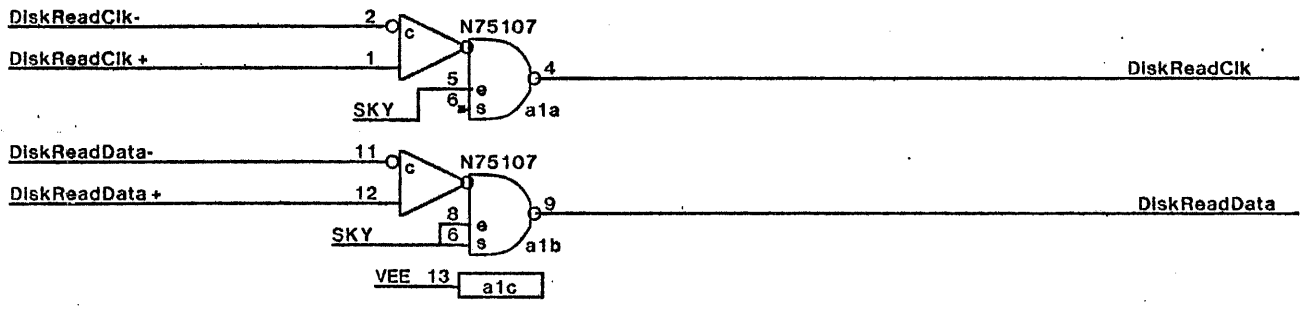
XEROX SDD	Project: Dandelion	Dandelion Disk Controller: MFM Encoding, PreComp, Address Mark Generation	File: HSIO52.sil	Designer: Davies	Rev: Q	Date: 6/21/80	Page: 52



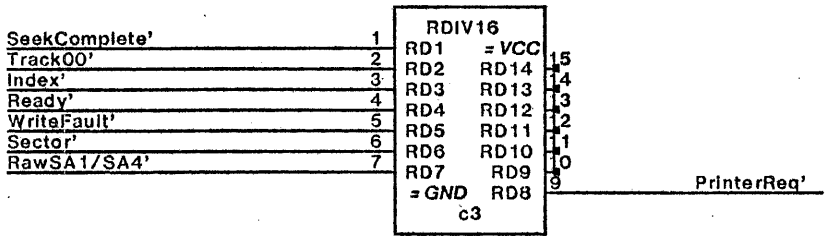
XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Disk Output Buffers, Drivers	HSIO53.sil	Davies	Q	7/23/80	53



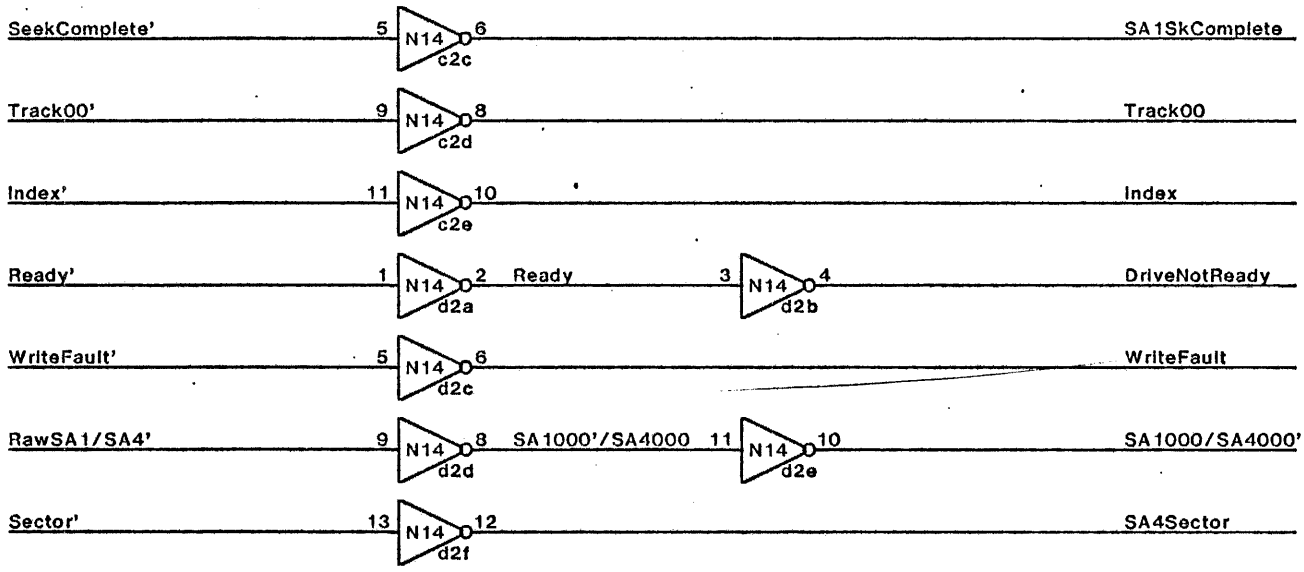
XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Phase Decoder Logic	File: HSIO54.sil	Designer: Davies	Rev: Q	Date: 7/23/80	Page: 54



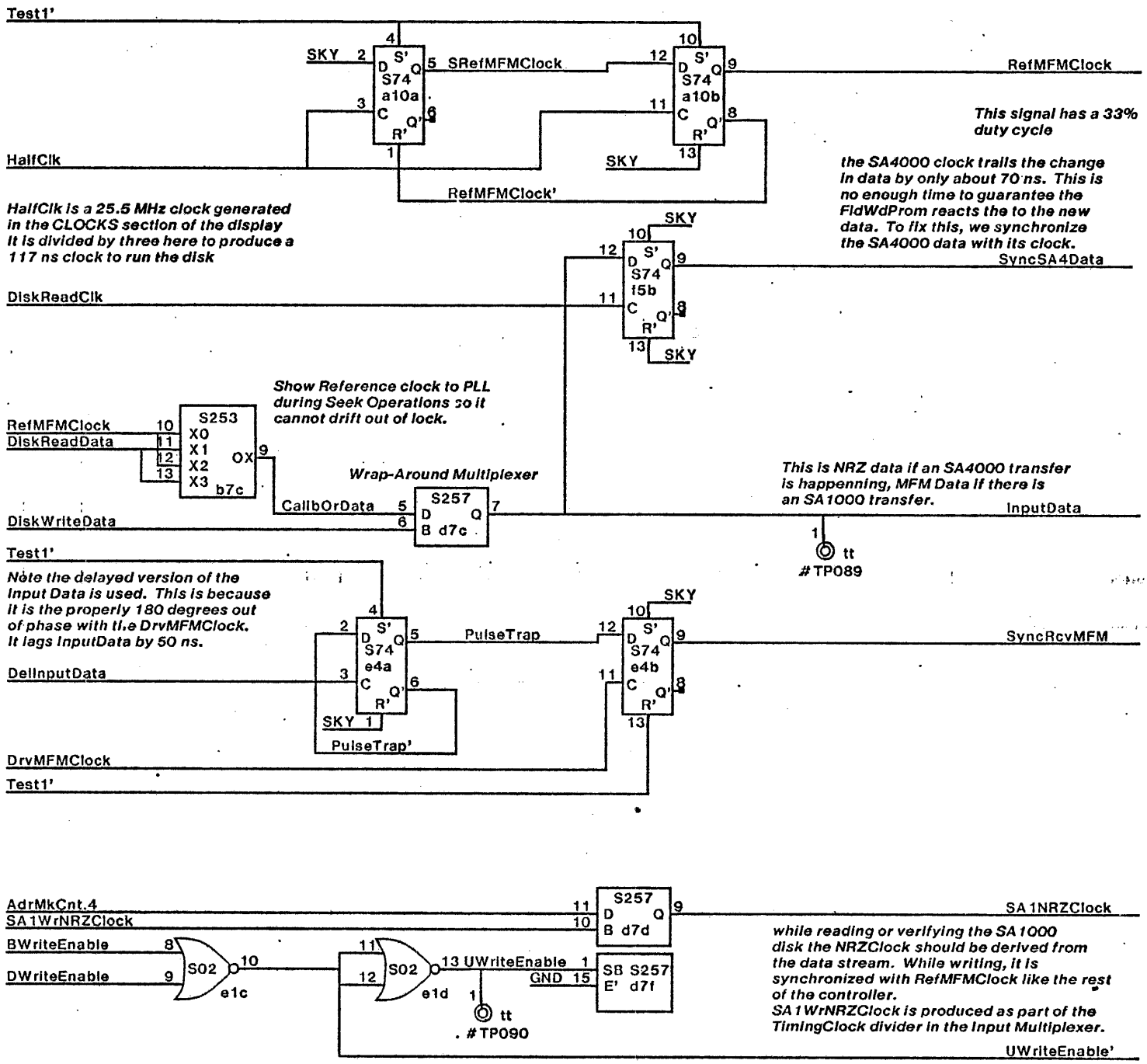
This is a Beckman RPack number 898-5-R220/330.

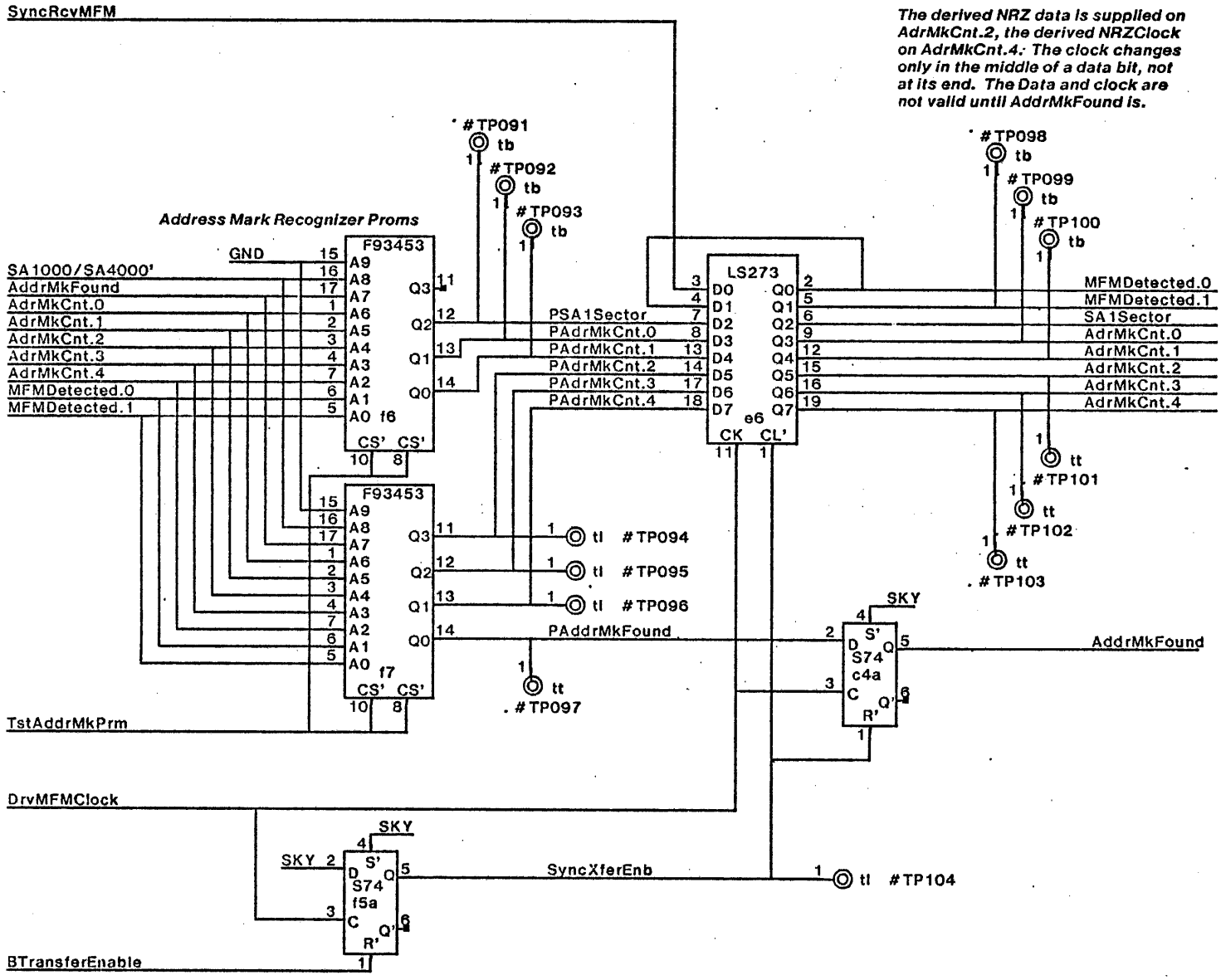


Pull this up so display request will work even if Options board is unplugged.

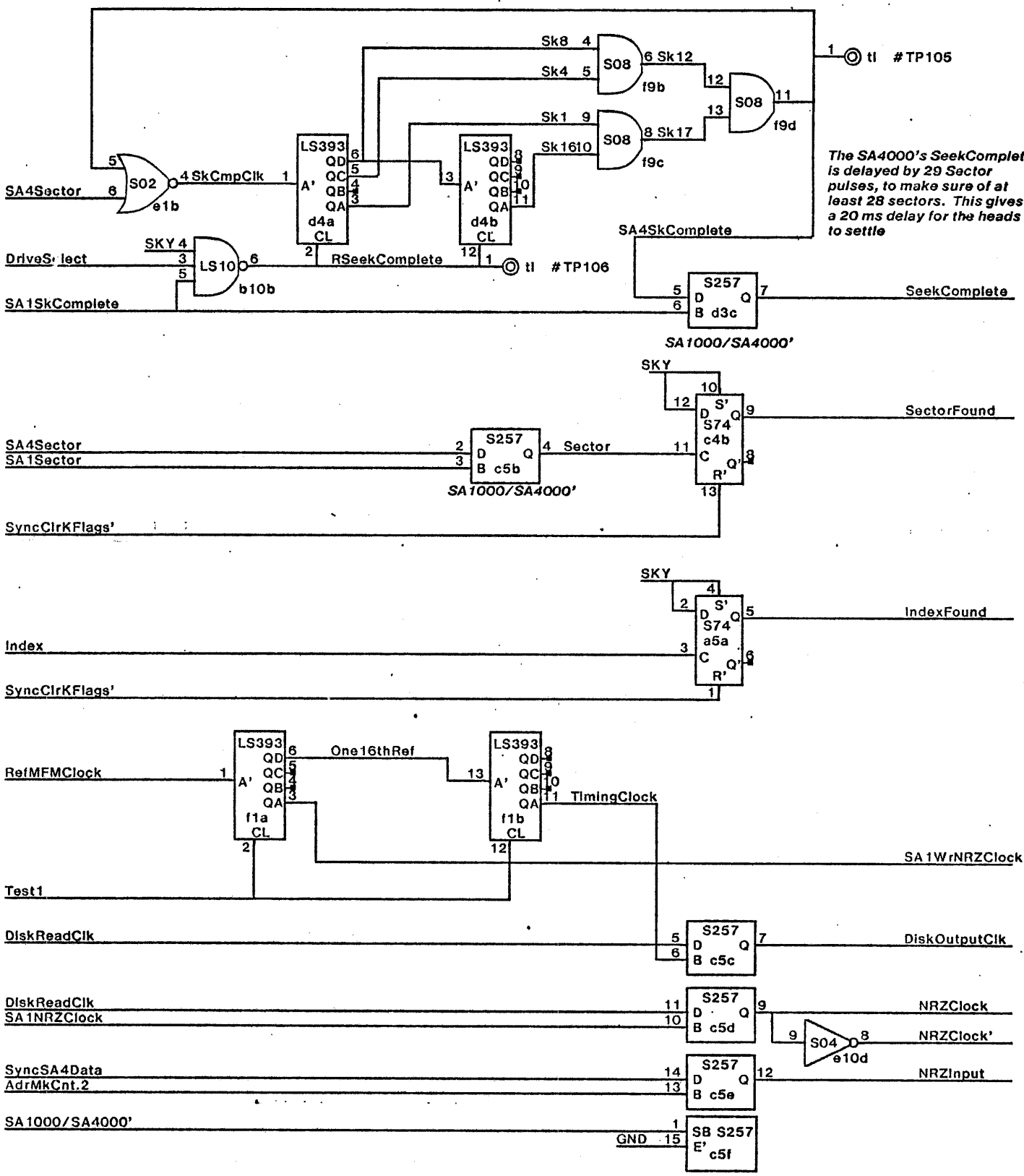


XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Input Buffers and Receivers	File: HSIO55.sil	Designer: Davies	Rev: Q	Date: 5/30/80	Page: 55


Test1'
SKY
S74 a10a
SRefMFMClock
S74 a10b
RefMFMClock
HalfClk
SKY
RefMFMClock'
This signal has a 33% duty cycle
the SA4000 clock trails the change in data by only about 70 ns. This is no enough time to guarantee the FldWdProm reacts the to the new data. To fix this, we synchronize the SA4000 data with its clock.
HalfClk is a 25.5 MHz clock generated in the CLOCKS section of the display It is divided by three here to produce a 117 ns clock to run the disk
SKY
S74 f5b
SyncSA4Data
DiskReadClk
SKY
Show Reference clock to PLL during Seek Operations so it cannot drift out of lock.
RefMFMClock
DiskReadData
S253
X0
X1
X2
X3
OX
b7c
Wrap-Around Multiplexer
CallbOrData
S257
B d7c
DiskWriteData
This is NRZ data if an SA4000 transfer is happenning, MFM Data if there is an SA1000 transfer.
InputData
tt
TP089
Test1'
Note the delayed version of the Input Data is used. This is because it is the properly 180 degrees out of phase with the DrvMFMClock. It lags InputData by 50 ns.
SKY
S74 e4a
PulseTrap
S74 e4b
SyncRcvMFM
DelInputData
SKY
PulseTrap'
DrvMFMClock
Test1'
AdrMkCnt.4
SA1WrNRZClock
S257
B d7d
SA1NRZClock
BWriteEnable
DWriteEnable
S02
e1c
S02
e1d
13 UWriteEnable
GND
SB S257
E' d7f
tt
. # TP090
while reading or verifying the SA1000 disk the NRZClock should be derived from the data stream. While writing, it is synchronized with RefMFMClock like the rest of the controller.
SA1WrNRZClock is produced as part of the TimingClock divider in the Input Multiplexer.
UWriteEnable'
We note that AdrMkCnt.4 ceases to be active when TransferEnable drops. A clock is needed to start a write operation, so SA1NRZClock is set to the always active SA1WrNRZClock as soon as WriteEnable goes active. To ensure the DWriteEnable shift regiter delay is cleared, we keep SA1NRZClock set to SA1WrNRZClock until DWriteEnable goes lo.

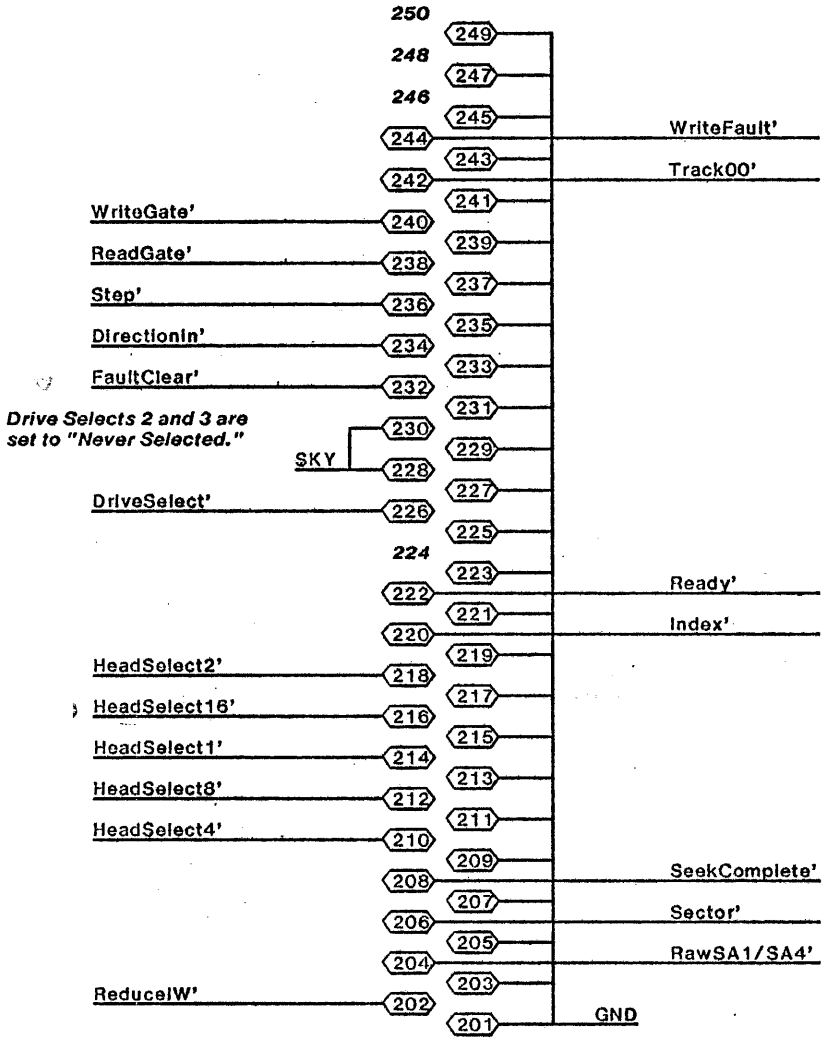


XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Data Sep. and Addr Mk Detect.	File: HSIO57.sil	Designer: Davies	Rev: Q	Date: 6/21/80	Page: 57

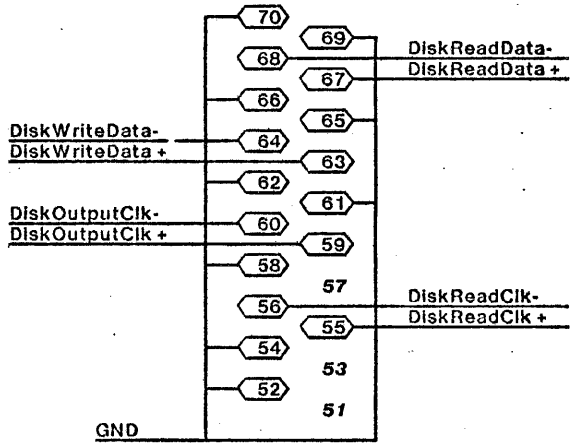


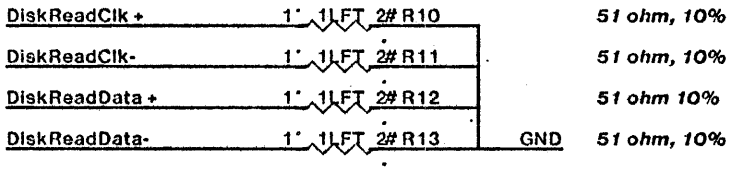
XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Input Multiplexer	File: HS1058.sil	Designer: Davies	Rev: Q	Date: 7/23/80	Page: 58

This 50 pin connector is in location D of the HSIO board.

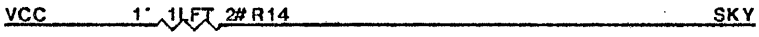


this 20 pin connector occupies position C on the HSIO board.

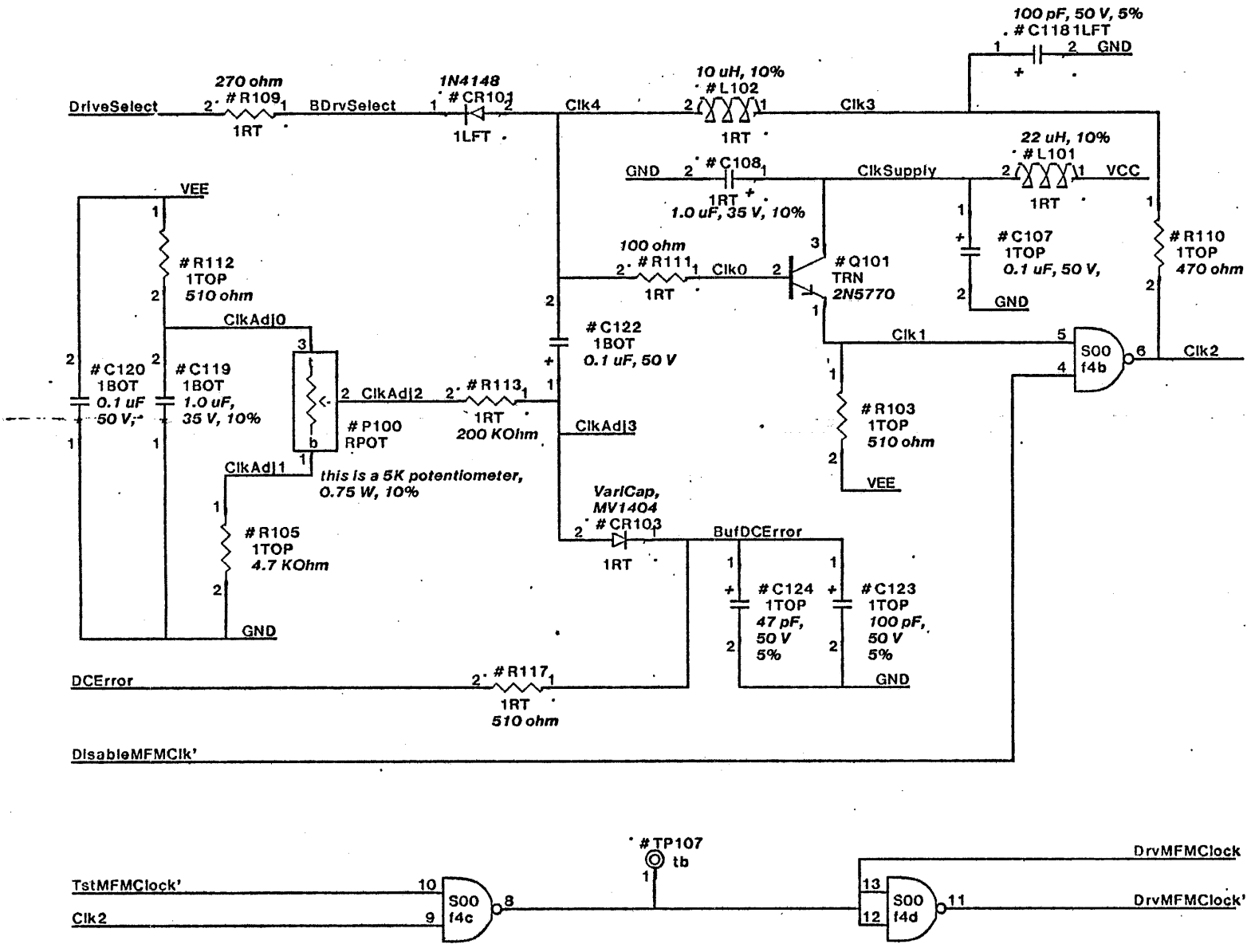




This resistor supplies logical one to the board
It is also 51 ohms

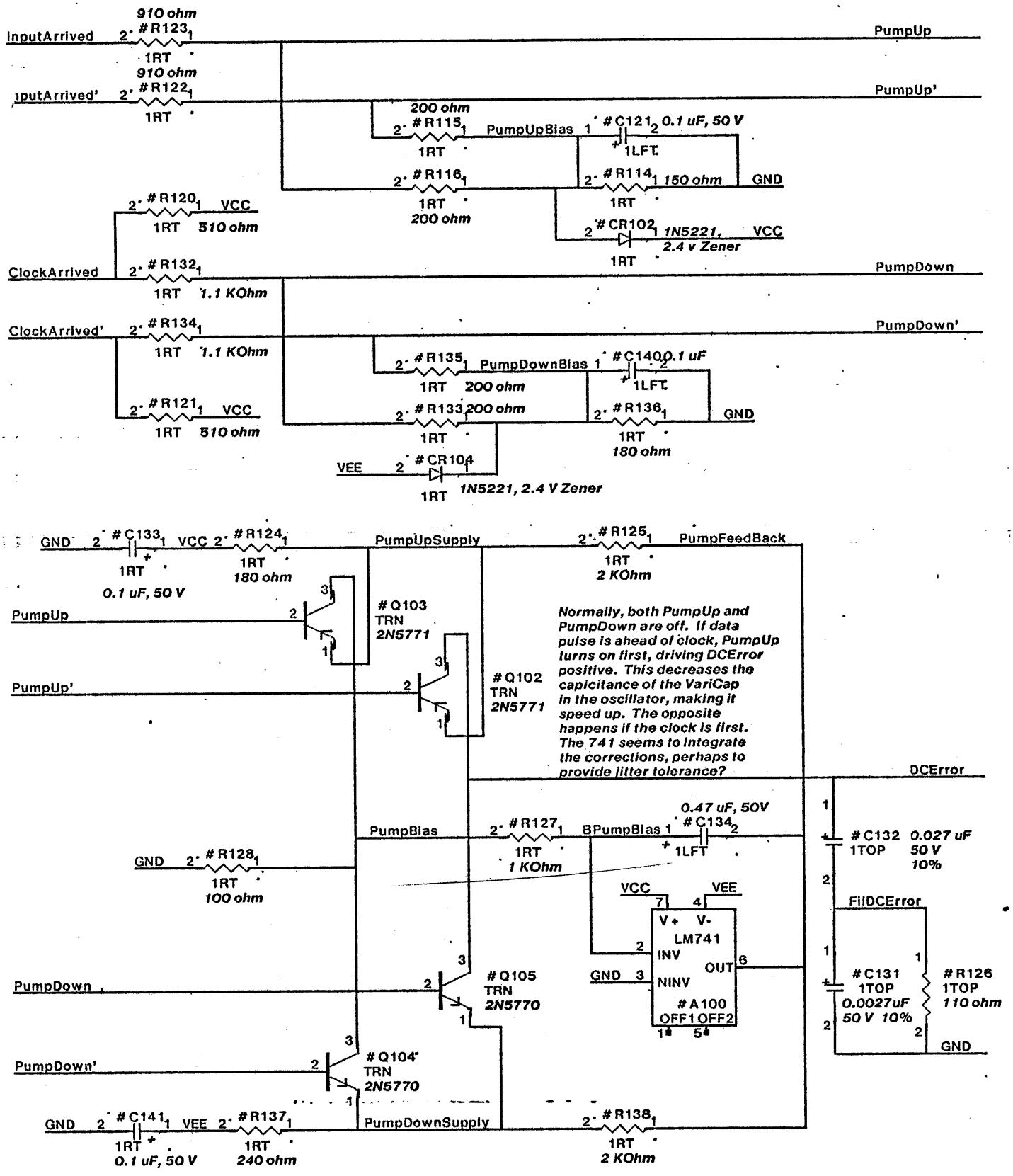


XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Disk Cables and Discretes	pHSIO59.sil	Davies	Q	5/30/80	59



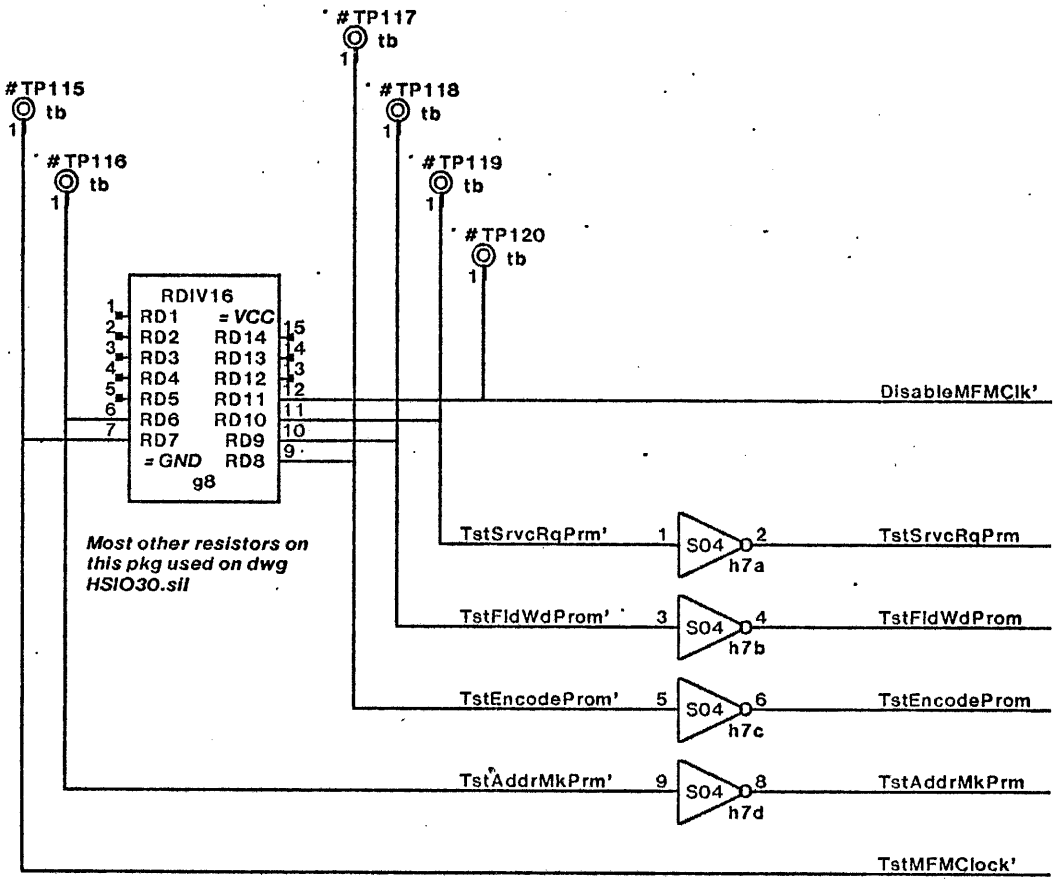
All resistors shown above are 0.25 watt, 5% parts

XEROX SDD	Project	Dandelion Disk Controller	File	Designer	Rev	Date	Page
	Dandelion	Phase Decoder Oscillator	pHSIO60.sil	Davies	Q	7/16/80	60



*All resistors shown are 0.25 watt, 5% parts. All capcitors shown on this page have 10% tolerances.

XEROX SDD	Project: Dandelion	Dandelion Disk Controller: Discrete Phase Comparator	File: pHSI061.sil	Designer: Davies	Rev: Q	Date: 5/30/80	Page: 61


#TP115
tb
#TP116
tb
#TP117
tb
#TP118
tb
#TP119
tb
#TP120
tb
RDIV16
RD1 = VCC
RD2 RD14
RD3 RD13
RD4 RD12
RD5 RD11
RD6 RD10
RD7 RD9
= GND RD8
g8
Most other resistors on this pkg used on dwg HSIO30.sil
DisableMFMClk'
TstSrvcRqPrm'
S04
h7a
TstSrvcRqPrm
TstFldWdProm'
S04
h7b
TstFldWdProm
TstEncodeProm'
S04
h7c
TstEncodeProm
TstAddrMkPrm'
S04
h7d
TstAddrMkPrm
TstMFMClock'