Trident disk for the Alto

by R. D. Bates

June 15, 1979 (revised November 24, 1979)

This document describes the Trident disk controller and microcode interface which is available for placing high performance disk capabilites on Alto computers.

Key words and phrases: Trident, Alto disk, TFS.

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

This memo describes the facility for an 80 megabyte capacity, 9.7 megabit transfer-rate disk for use on Alto computers. The new disk is implemented *in addition* to the standard Alto disk. This simply means that new microcode tasks (one for output and one for input) are used, leaving the standard disk microcode intact. The requirements for implementing this facility are a Trident **T-80** disk ($5500), a disk control board ($800), and an Alto II. Older style Altos can have Trident disks but there controller is not readily avalable.

1. The basic disk system

The disk controller is designed to drive any member of the **Trident** family of disk drives manufactured by Century (now part of Xerox). Currently the T-80 and the T-300 drives have been connected. A summary of the specifications for the **T-80** drive is found in Fig. 1.

The disk system is accessed through a many level addressing scheme. First a particular disk drive is accessed (there may be from 1 to 16 drives attached to a single disk controller). The surface of a particular disk pack is selected by specifying 1 of 5 *heads*, and 1 of up to 815 tracks. Each track is further broken down into sectors. The number of sectors is determined by jumpers within the disk drive which determine the number of words reserved for each sector (3070 words max.). Each sector can be further broken down into blocks, each of which can be either read, written, or checked. Reading or writing within a sector must start with the first block and continue, but one does *not* have to read or write to the end of a sector. A block may not be read after writing any block within that sector. The checking scheme is intended for checking the header and label, and will inhibit writing later blocks within the sector should a compare error occur.

The size and number of blocks within any sector is determined by parameters from the Alto program which are given to the microcode for each sector. Thus the sector formatting does not have to be the same for the whole disk. All the necessary delays for turning on read or write circuitry are created within the disk controller with a single PROM which is programmed with the appropriate delays. The choice of the number of blocks, and their size, should be made (unless you wish to defeat the "sector overflow" check) such that all blocks are within a given sector area on the disk.

Due to the high data density used in this system, a disk pack certified for use by these disk drives does not have to be completely perfect. A disk pack is suitable if it has no more than three bad areas on any of the five surfaces; where a bad area is defined as one which could potentially cause read errors of no more than 11 data bits in length. In order to correct such errors, as well as other very occasional read errors, a scheme of error detection and correction has been implemented in the disk controller which will detect (with very high probability) errors of any length, and correct any burst error of 11 bits or less. Warning! If a burst error of more than 11 bits occur there is a significant possibility that the error correction algorithm will *incorrectly* correct the error and thus double the number of bad bits.

T-80 SPECIFICATIONS AND CHARACTERISTICS

CAPACITY

82.1 million 8-bit bytes - unformatted (312 for T-300)

TRANSFER RATE

9.67 megabits per second
16 bit word every 1.65 mj.

ACCESS TIME

Track to Track Positioning - 6 milliseconds maximum
Average Positioning - 30 milliseconds
Full Stroke Positioning - 55 milliseconds maximum
Average Latency - 8.3 milliseconds

ROTATIONAL SPEED

3600 revolutions per minute (16.66 milliseconds per revolution)

PACK START/STOP TIME

Start Time - 20 seconds
Stop Time (with dynamic braking) - 20 seconds

SECTOR LENGTH SELECTION

12 bit increments through jumpers on sector board

DENSITIES

Track Density - 370 tracks per inch
Recording Density - 6060 bits per inch maximum

DISK PACK CHARACTERISTICS

Disk Pack - IBM 3336-type components
Recording Surfaces - 5 plus 1 servo surface
Tracks per surface - 815

OPERATING METHODS

Recording Method - Modified Frequency Modulation
Positioning Method - Linear Motor; Track Following Servo

MECHANICAL SPECIFICATIONS

Size - 17.8" wide x 10.5" high x 32" deep
Weight - 230 pounds

ERROR RATE

Recoverable: 1 error in 10^{10} bits
Non-recoverable: 1 error in 10^{13} bits
Positioning: 1 error in 10^{6} seeks

CONTROLS AND INDICATORS

Ready Indicator
Fault Indicator
Start/Stop Switch
Read Only Switch
Degate Switch (takes disk off-line for testing)

FIGURE 1

2. Available software

A file system equivalent to that used on the standard Diablo 31 disk drives is available. This system is described in the Alto Operating System Reference Manual. In addition, other programs associated with or for use on the Trident disk system are as follows:

[MAXC]<ALTO>TFS.dm
: This file package is equivalent to the "BFS" file system in use on the Alto which are used for implementing file oriented access the disk. This also contains the microcode necessary for driving the disk controller, including a small routine called during error recovery. Documentation for the TFS package (and the TFU program, mentioned below) is contained in the Alto Subsystems manual and in <AltoDocs>TFS.tty.

[MAXC]<ALTO>TRIEX.run (also available as a network boot file)
: An exerciser program used for basic debugging and to check general disk reliability.

[MAXC]<ALTO>TFU.run
: A program for creating and manipulating a file system on a Trident disk, for certifying new disk packs for operation, and for exercising the file system at a high level.

It should be mentioned that writing software to drive the Trident controller directly has proven to be quite difficult, especially in the areas of initialization error recovery. If you choose to run the Trident controller directly (through the interface described in section 4 of this document), you should first examine carefully the code in the TFS package.

3. A "standard" configuration

The following is a description of the sector formatting which is most often used for general file storage on this disk system. If you wish to look at other configurations, you can go to the section on Format Specifications for full details.

Each track (10,080 words) is divided into 9 sectors of 1120 words each. Each sector is then divided into 3 blocks in much the same way as the standard Alto disk. The first block is a Header, and contains 2 words for identifying the disk address of that particular sector. The Header is normally read and checked by the controller in order to verify that the disk drive is actually where the controller thinks it is. The second block is the Label, and contains 10 words for storing useful file dependent information, such as name, type, address of next sector, etc.. The last block is 1024 words of actual file data.

Using the above format on a T-80 drive yields a total formatted capacity of 37,552,200 words (1024 words x 9 sectors x 815 cylinders X x surfaces). Using this format, the average transfer rate per disk revolution is 8.85 megabits per second.

4. Command blocks

The disk controller contains a "Run-Enable" flip-flop (initially turned off) which can be controlled by the emulator task through the execution of the SIO (StartIO) instruction. An SIO with bit 10

set will cause Run-Enable to be set. An SIO with bit 11 set to one will cause Run-Enable to be reset. Note that issuing an SIO with bit 10 set will wake up the microcode once and thus may report status in the absence of sector pulses from a disk. This facility is important since sector pulses are present only when a drive is running with the heads loaded.

The Alto program communicates with the disk controller via a four-word block of main memory at location KBLK (currently 640 octal), specified as follows:

KBLK	Pointer to first disk command block
KBLK+1	number of currently selected drive
KBLK+2	cylinder position used for the last command
KBLK+3	Status of current drive at last sector pulse
KBLK+4	Error which caused last transfer to be aborted

The microcode program is woken up at every sector pulse by the hardware. The program then updates the status entry, and checks to see if the command pointer is non-zero. If a command is present then processing begins, otherwise the microcode blocks until the next sector pulse. The command table pointed to by KBLK is a variable length block which completely specifies a transfer for a given sector. The block has the following format:

DCB	Cylinder select
DCB+1	left byte - Head select (values 0 through 4)
DCB+1	right byte - Sector count (values 0 through 15)
DCB+2	Drive select
DCB+3	Pointer to next command block
DCB+4	Disk Command Seal (octal 122645)
DCB+5	R/W command
DCB+6	Word count
DCB+7	Memory block pointer
DCB+8	Error Correcting Code 0
DCB+9	Error Correcting Code 1
DCB+10	Status
DCB+11-	
DCB+16	Possible repeats of DCB+5 through DCB+10
-	-
-	-
DCB+?	All zero’s R/W command word to terminate
DCB+?+1	Interrupt word

The disk command block is made up of a five word introduction followed by a variable number (0 to n) of six word block descriptors. The introduction contains a description of the sector location, and the block descriptors describe the transfer required for each block within the sector. Processing terminates when a R/W command within a block descriptor is found to be zero; at this time, the following word is ORed into NWW to generate an interrupt on the channels corresponding to one bits in that word.

A detailed description of the entries in these two tables follows.

KBLK -Command Block Address-

This word is tested by the microcode every sector time or upon the completion of a disk command block. If the value is zero, then the current disk status will be placed in KBLK+3 and the microcode will block until the next sector pulse. If the value is non-zero, then command block proccessing will start at the address pointed to by KBLK. The microcode will update KBLK with the address of the command block it is currently working on. However, the microcode may advance to the next command block before all data transfer activity for the previous one has completed; consequently, checking for KBLK to be zero is *not* a safe way to determine whether the disk is idle.

KBLK+1 -Drive number

This word is updated by the microcode to reflect the drive that is currently selected. The software can force a new drive select by storing 100000B + drive number into this cell.

KBLK+2 -Cylinder Position-

This word is updated by the microcode to indicate the cylinder address last issued to the disk drive. The cylinder address of any successive disk transfer is compared to this value to determine whether a seek command should be issued to the drive. Unnecessary seek commands will cause no problems but will take a full sector time to be completed. The software can force a seek to take place by storing -1 into this cell.

KBLK+3 -Sector Status-

This word is updated by the microcode to indicate the disk status at the last sector pulse. Note that this word in NOT updated when the microcode is busy processing a disk command block.

Bit 0 -Seek Incomplete-

Indicates that the disk drive has not correctly positioned the heads within the last 700ms. A RE-ZERO command must be issued to the drive in order to clear this error.

Bit 1 -Head Overflow-

Indicates that the head address given to the disk drive is invalid (i.e. greater than 4).

Bit 2 -Device Check-

One of the following errors occurred.

a) Head select or Cylinder select or Write commands and disk not ready
b) An illegal cylinder address.
c) Offset active and cylinder select command.
d) Read-Only and Write.
e) Certain errors during writing, such as more than one head selected, no transitions of encoded data or heads more than 80 micro-inches off cylinder.

A RE-ZERO command may be required to clear this error.

Bit 3 -Not Selected-

The selected drive is in "off-line" test mode or the selected drive is not powered up.

Bit 4 -Not On-Line-

The drive is in test mode or the heads are not loaded.

Bit 5 -Not Ready-
There is a cylinder seek in progress or the heads are not loaded.

Bit 6 -Sector Overflow-
The controller detected that a write command was active when the next sector pulse occurred. The controller will inhibit writing as soon as this error is detected. This error implies either a hardware malfunction or a discrepancy between the format of the drive and the format the program thinks the drive has.

Bit 7 -Output Late-
The 16 word output buffer within the disk controller became empty while either a read or a write command was in progress. The controller will inhibit any writing until a Device Check Reset command is issued.

Bit 8 -Input Late-
The 16 word input buffer within the disk controller became full. This error will cause words read into memory to be left shifted by the number of words lost, and the error correcting code to be non-zero.

Bit 9 -Compare Error-
The data read during a "Read and Compare" operation did not match the data read off the disk. The controller will inhibit any writing for the remainder of the sector if this error is detected.

bit 10 -Read Only-
The "Read-Only" switch on the disk drive in on.

Bit 11 -Offset-
The cylinder position is currently offset. This is a mode used for recovery of bad data.

Bit 12 -
Bit 15 -Sector Count-
A value from 0 to count-1, where count is the number of sectors implemented in the disk drive. This value, you might have noticed, restricts the number of sectors to 16 or less. The value returned here is the sector count for the *next* sector on the disk.

KBLK+4 -Command Abort Errors-
This word is set by the microcode if, while processing a command, it finds some extraordinary error. In this circumstance, the status word in the DCB will not be written (though it may have been written in a previous DCB), but the reason for aborting will be reported here.

Bit 0 -
Bit 3 -not used-

Bit 5 -Seek Incomplete-
The last DCB was aborted because the seek was not completed (within 64 sector times).

Bit 6 -

Bit 9 -not used-

Bit 10 -Output Late-
The command chain was aborted because an Output Late error occurred in some previous command. Because commands as well as data are sent to the controller through the output data path, an Output Late command may cause the controller hardware and microcode to get out of sync, so further processing is aborted to prevent issuing illegal commands to the controller. This condition must be reset by issuing a Device Check Reset command.

Bit 11 -Invalid Seal-
The last DCB was aborted because word DCB+4 was not equal to 122645 octal.

Bit 12 -
Bit 13 -not used-

Bit 14 -Aborted-
The last DCB was aborted because of one of the errors mentioned here.

Bit 15 -Invalid Sector-
The last DCB was aborted because the hardware sector counter never equaled the specified sector number (within 64 sector times).

DCB -Cylinder Select-
This word will cause the disk drive to position its heads over the indicated cylinder if the cylinder address is different than the previously selected cylinder. Cylinder positioning causes the disk drive to go non-ready for 6 ms. cylinder-to-cylinder and 55 ms. full seeks. (Typical seek times appear to be around 3 ms. cylinder-to-cylinder and 50 ms. full seek)

Bit 0 -
Bit 3 -all zeros-

Bit 4 -
Bit 15 -Cylinder Select-
This field may take on any value, but be sure that it represents a valid address for the disk drive being used. The microcode will update KBLK+2 with this value.

DCB+1 -Head and Sector Select-
This will select the particular head (or surface) for the next command as well as the sector to be used in the transfer.

Bit 0 -Off Track-
This bit may be activated during a read in order to attempt to recover bad data. When activated, this bit will cause the cylinder positioning mechanism to move 80 micro-inches off track.

Bit 1 -Direction-
This bit determines the direction of off track positioning if bit 0 is set.

Bit 2 -

Bit 4 -not used-

Bit 5 -

Bit 7 -Head Select-
The actual head specification.

Bit 8 -
Bit 11 -not used-

Bit 12 -

Bit 15 -Sector Number-
The actual sector specification. Be sure that you specify a valid sector number. There is no test in the microcode for invalid sector numbers, and the microcode will loop forever looking for an invalid sector! (’Life is hard!’).

DCB+2 -Drive Select-
This will specify the disk drive to be involved in the upcoming command.

Bit 0 -
Bit 11 -must be all zeros-

Bit 12 -
Bit 15 -Drive number-
This value is used by the disk controller for selecting the appropriate unit. The microcode will update KBLK+1 with this value.

DCB+3 -Next Command Pointer-
This word is read at the end of a disk command, and its contents are placed in KBLK.

DCB+4 -Disk Command Seal-
This word is tested to see that it equals 122645 octal. If this test fails, then KBLK is set to zero and processing is terminated. This word is overwritten with status information during disk processing so that a DCB table cannot be used twice by the controller without program intervention.

DCB+5 -R/W Command-
This word is first checked for zero. If so, then the sector transfer processing is complete, and the microcode will either go on to the next command or terminate. If the word is non zero, the microcode will then check to see that the cylinder positioning is not active, and wait if it is.

Next the sector count is fetched from DCB+1, and the microcode enters a loop waiting for the sector count in KBLK+3 to equal the desired sector. Once this is found, a "wait for next sector" command is issued and then the block transfer commands are sent to the output FIFO for processing.

If the first command in a DCB is nonzero but has neither the Read nor the Write bit set, the microcode will reset the controller, clearing the Sector Overflow, Output Late, and Compare Error conditions (which inhibit writing on the disk). This is typically done using a Device

Check Reset or Re-Zero command, and should be done only after waiting for the disk to be idle.

The control bits of the read/write command are as follows:

Bit 0 -
Bit 3 -not used-

Bit 4 -Check Data-
This bit is examined during a read command to see whether incoming data should be compared against that data which already exists in the Alto memory. In this mode, the first n non-zero words from the Alto memory are passed to the disk controller for comparison during reading, where n is at least 2, and not greater that the word count for that block. The disk controller will perform a normal read (placing all but the first 2 data words read into Alto memory), and in addition it will set an error bit if the check words are different. This check bit will inhibit any further write operations within the sector.

Bit 5 -not used-

Bit 6 -Strobe late-
This is used in conjunction with read in an attempt to recover bad data.

Bit 7 -Strobe Early-
Same as bit 6. I don't know what happens if both bits are set.

Bit 8 -Write-
Indicates that the transfer is to be a write.

Bit 9 -Read-
Indicates that the transfer is to be a read.

Bit 10 -not used-
Bit 11 -Head Address Reset-
Please don't use this bit, since it will force head 0 to be selected.

Bit 12 -Device Check Reset-
This bit is used to clear any error stored by the device check circuitry. This bit is activated by the microcode and should not be set here except to recover from a Device Check, Compare Error, Output Late, or Sector Overflow. The read, write, and check bits should not be activated when this bit is set.

Bit 13 -Head Select-
This bit causes the head select electronics to activate the selected head. This bit should always be set for a read or a write command.

Bit 14 -Re-Zero-
This is a special control function which is used to cause the disk drive to completely retract its cylinder positioning arms and then reposition them on cylinder 0. This bit should only be activated when a head positioning error has been detected. The read,

write, and check bits should not be activated when this bit is set.

Bit 15 -Advance Head Address-
Please don't use this bit either, since it will increment the head address selected by 1.

DCB+6 -Word Count-
This word specifies the number of words in the block to be read or written. This number does not include the two words of check sum at the end of each block.

DCB+7 -Memory block Pointer-
This is a pointer to the appropriate memory block in Alto memory.

DCB+8 -
DCB+9 -Error Correcting Code-
These two words are updated at the end of a read with the contents of the error code shift registers. If no read error has occurred then both these words will be identicaly zero. If they are non-zero, then an error has occurred, and error correction may be applied. The error recovery process runs as a BCPL procedure, and a small microcode routine, which are capable of determining the error bits in less than 5 ms.

DCB+10 -Status-
This word contains the disk status at the end of processing the current block.

Bit 0 -
Bit 10 -Disk & Controller errors-
The status bits in this field are the same as those described for KBLK+3.

Bit 11 -ECC error-
Indicates that one of the two ECC words was found to be non-zero.

Bit 12 -
Bit 15 -constant-
This field is set to 1 by the microcode. Note that this means a 1 is returned in the entire status word for a transfer with no errors.

5. Format Specifications

Various delays must be provided at the beginning of each sector block in order to allow for electrical and mechanical tolerances within the disk drive. For the purpose of defining a new disk format, one simply needs a summary of "words lost" for each block.

1) total words per disk revolution =	10,080
2) words lost for the 1st block =	32
3) words lost for successive blocks =	14
4) required gap at end of sector =	14

The number given for "words lost" for each block includes; 2 words of error detection and correction (32 bits of ECC code) which are always added at the end of the data written, 1 word of

trailing zeros (to assure that all data is sent before the write electronics is turned off), 1 word for the disk controller to execute the read/write command, and the number of delay words allotted by the disk controller as required by the disk drive for mechanical and electronic delays.

Using these numbers on the standard Alto disk format yields the following numbers. The number of words available per sector is 10,080/9 = 1120. The total words lost for disk formatting is 32 for the first block, 28 for the second and third block, and 14 words at the end of the sector, for a total of 74. Subtracting 74 from 1120 gives us 1046 words remaining for the three fields. This will give us 2 words for Header, 10 words for Label, and 1024 words for Data, with 10 words unused.

6. Error Detection and Correction

The following section describes (as best I can) the provisions implemented for error correction for the Trident disk drives. This capability is done as a mixture of disk controller hardware (for ECC generation and checking) and system-software/microcode (for error recovery).

The error detection and correction scheme implemented in the disk controller is a compromise of capability, speed, and cost. The number of check bits generated is 32 (2 words are easy to keep track of), and the total controller chip count has been held down to fit along with the Ethernet controller, on a single Dorado printed circuit board. The basic capabilities and restrictions are summarized below.

1) Correction of *single* error burst of length not to exceed eleven data bits. (Example: for the data "000 1100101", the data "000 0101101" contains a single burst error of length 4.)

2) Capable of correcting record lengths of up to 2684 words. The error detection and correction code implemented will detect errors in arbitrary length records, but not enough information is generated for error correction if the sector length exceeds 2684 words.

3) Simple error detection. 2 words are returned by the hardware, which if both words are zero, indicates a successful read.

4) Error correction in less than one revolution of the disk drive. The error correction procedure is well suited for implementation in a mixture of BCPL and microcode. The bulk of the processing, which is in microcode, will take a maximum of 4.5 ms. worth of microinstructions. This is sufficiently fast to allow for reading a sector, finding that an error occurred, attempting to correct it, and if not correctable, initiating a new read of the same sector on the next revolution.

5) Not all uncorrectable errors will be detected as such. The probability of an uncorrectable error being generated is exceedingly small. It requires two bad spots on the disk surface within one sector (the pack is bad - throw it out!), an electronic error in a sector with a bad spot, or two electronic errors within one sector. Given that such an error has occured, it can, with a probability of say 20 percent, result in an error pattern and displacement which is seemingly valid. This will result in leaving

the error bits un-corrected and, in addition, changing some bits which were in fact correct. This means that for high data security, a check code should be generated and imbedded as part of the data file before writing on the disk.

The following is a more detailed description of the error correcting code used, and the procedure used for data recovery.

The error correcting code (ECC) generated is referred to as a Fire Code (see *Error-Correcting Codes* by Peterson), and is capable of correcting any single *burst error* of up to 11 bits in length (that is a scattering of error bits within the bit stream, all of which fit within an 11 bit span). The code calls for dividing the outgoing data stream by a polynomial of the form:

$$P(X) = P_1(X)(X^m + 1)$$

Where $P_1(X)$ is an irreducible polynomial of degree n (n = burst length) and m is $\geq 2*n - 1$. For this particular application the polynomials chosen are:

$$P(X) = (X^{11} + X^2 + 1)(X^{21} + 1)$$

During a write operation, the two polynomials are multiplied together and implemented by hardware in the form:

$$P(X) = X^{32} + X^{23} + x^{21} + X^{11} + X^2 + 1$$

The data stream is premultiplied by X^{32} to make room for the 2 word ECC and then reduced modulo P(X). This is accomplished by the normal feedback shift register technique with the difference that to perform premultiplication, the output of the register is exclusive-or'd with the incoming data and then fed back. After all data bits have been shifted out, the contents of the ECC shift registers are appended to the disk block.

During the read operation, the feedback shift register is reconfigured such that the two original polynomials are implemented separately. The incoming data stream, including the 2 appended words of ECC, is independently reduced modulo $P_0(X)$ and $P_1(X)$, where

$$P_0(X) = X^{21} + 1$$

$$P_1(X) = X^{11} + X^2 + 1$$

After reading in all words off the disk, the contents of the two polynomial shift registers are read into the Alto. If the data is recovered without error, then reducing it modulo $P_0(X)$ and $P_1(X)$ results in the registers containing all zeros.

If the data contains an error, then the two registers will be non-zero. If one but not both registers is non-zero, then the error is non-recoverable.

Given that one finds an error, then a procedure is undertaken which determines the pattern of bits which are in error, and the displacement of this pattern from the end of the record. I am simply going to present the magic equation to be solved, and some magic constants to be used for solving this equation. Much of the polynomial implementation and the equations, which use the "Chinese

Remainder Theorem" are discussed in technical reports from CALCOMP (Calcomp Technical Report TR-1035-04, by Wesley Gee and David George) and XEROX (Xerox XDS preliminary report "Error Correction Code for the R.M. Subsystem, by Greg Tsilikas, March 28, 1972.).

The basic equation is:

$$D = Q*LCM - (A_0*M_0*S_0 + A_1*M_1*S_1)$$

Where:

D = displacement from the end of the record.
Q = smallest integer to make D positive
A_i = a constant such that $A_i*M_i = 1$ modulus E_i
E_i = modulus of the polynomial
$M_i = LCM/E_i$
LCM = least common multiple of E_0 and E_1
S_i = number of shift operations to the appropriate polynomial remainders as described below.

The values of E_0 and E_1 were found by programing the procedure outlined in the CALCOMP report, and yielded the following result:

$$E_0 = 21 \qquad E_1 = 2047$$

The least common multiple (LCM) of E_0 and E_1 is simply the product of E_0 and E_1 since the two numbers have no factors in common. Thus the LCM, which is also the record length which can be corrected, is 42,987 bits, or 2684 words.

Knowing LCM and E_0 and E_1, the values of M_0 and M_1 are easily found to be

$$M_0 = 2047 \qquad M_1 = 21$$

The values of A_0 and A_1 are next determined using a trial and error approach that I put in a small program. The results can easily be confirmed, and are given below:

$$A_0 = 19 \qquad A_1 = 195$$

All of the above values derived so far are constants determined for the particular polynomials chosen. The values of S_0 and S_1 are determined in the software from the error patterns returned at the end of a disk transfer.

S_0 is first determined through a BCPL procedure by the following steps.

1) The remainder from dividing the input data by $X^{21} + 1$ is found in the first ECC word, bits 11 through 15, and the second ECC word, such that first word bit 11 is the most significant bit and second word bit 15 is the least significant bit.

2) First test the remainder for zero, and if so quit since the error is non-recoverable.

3) Test the low order 10 bits for all zeros, and if not then perform a left circular shift on the 21 bits. When the low order 10 bits are all zeros, the error pattern is in the upper 11

bits of the word, and S_0 is the number of times the circular shift was performed.

4) If the low order 10 bits don't become all zeros within 21 shifts (1 full cycle) an un-correctable error has occurred.

S_1 is then determined through a microcode procedure by the following steps.

1) The remainder from dividing the input data by $X^{11} + X^2 + 1$ is found in the second ECC word, bits 0 through 10.

2) First test the remainder for zero, and if so quit since the error is non-recoverable.

3) Test this number to see if it is equal to the error pattern determined in step 3 of S_0, and if not reduce this number modulo $X^{11} + X^2 + 1$ (left shift and XOR feedback). When the contents of this word equals the error pattern (it is guaranteed to happen with 0 to 2047 shifts), the value of S_1 is determined as the number of shifts performed (In the hardware implementation of switching from the write polynomial to the read polynomials, it was easier to implement a polynomial that premultiplied by X^{11}. This means that the remainder returned by the hardware already has had 11 shifts performed. To compensate, when S_1 has been determined by the above procedure, you must add 11 to the value, and subtract 2047 if the result is greater than 2047).

The basic equation for the displacement now looks like

$$D = Q*42{,}987 - 19*2047*S_0 - 195*21*S_1$$

Where:

$0 \leq S_0 \leq 21$
$0 \leq S_1 \leq 2047$

Notice that the straight-forward solution to this equation can not be done with single precision arithmetic on the Dorado. In order to avoid double precision arithmetic, I have used the following manipulation of the equation.

$$D = Q*2047*21 - 19*2047*S_0 - 4095*S_1$$
$$D = Q*2047*21 - 19*2047*S_0 - 2*2047*S_1 - S_1$$
$$D' = Q*21 - 19*S_0 - 2*S_1$$

where:

$0 \leq D' \leq 20$
$D = 2047*D' - S_1$ (add 42,987 if D' = 0)

7. Hardware implementation

Due to the high data transfer rates of the Trident disk (1.65 ms per word), the disk controller implementation includes two independent 16 word "First In - First Out" (FIFO) registers. One FIFO is used to buffer all output information, both control and data, and the other FIFO buffers all input data. A status input instruction has been implemented which is not buffered through the

FIFO’s.

The two FIFO’s are completely independent of one another, and are, in fact, serviced by two separate microcode tasks. With the standard allocation of task assignments in the Alto, the most suitable tasks available for the new disk are task 17, the highest priority task, for the input FIFO and task 3 for the output FIFO. With this implementation, a disk read can be performed without concern over the microcode activity of other tasks. The output task, while being of low priority, will have enough room in the FIFO to store all commands for reading the sector. The input task, which will have all the activity, will have the highest priority. During this disk activity, the standard Alto disk must be idle for its own good, while the display will still function, but with possible break-up of the display during the transfer. During a disk write, all task activity above the disk (i.e. everything but the emulator) must be minimized in order to guarantee adequate service to the disk. If the output FIFO is allowed to become empty, the "write late" flag will be set and the write command to the disk will be inhibited for the remainder of the transfer.

The fullness or emptiness of the appropriate FIFO generates a task wake-up request if there is enough room for at least 4 words to be transferred. This is sufficient to allow a double word memory access to take place in a minimum micro-instruction loop of 6 instructions per double word read/write.

All disk formatting delay constants are implemented with a 32 X 8 PROM. An Alto program is available which takes the required delay values from the user and computes the appropriate values to be put in the PROM. These constants do not involve the number of blocks per sector or the number of data words per block - these being determined from values passed to the microcode in the disk command table.

For applications requiring a single disk drive on an Alto, the hardware implementation involves a single Alto PC card which is plugged into a "processor" slot in the Alto. In the situation requiring more than one drive (with a maximum of 8) an additional Trident Multiplexor card is required. This card contains data and clock repeaters, disk select decoding and latch, and a sector count register. (It should be mentioned that the controller will not work unless drive 0 is connected and has AC power turned on; however, it is not necessary that drive 0 be on-line.)