CORVUS SYSTEMS

CORVUS DISK SYSTEM TECHNICAL REFERENCE MANUAL

PART NO. : 7100-03289

DOCUMENT NO. : CCC/10-99/1.0

RELEASE DATE : November, 1982

CORVUS DISK SYSTEM
TECHNICAL REFERENCE MANUAL

NOVEMBER 19, 1982

{This page left intentionally blank}

CORVUS DISK SYSTEM
TECHNICAL REFERENCE MANUAL

TABLE OF CONTENTS

{This page left intentionally blank}

1. DISK HARDWARE INTERFACE

1.1 General

All cable assignments are TTL.

1.2 Cable wire assignments

NAME	ORIGINATOR	FLAT CABLE WIRE
Data Bit 0	bi-directitonal	25
Data Bit 1	bi-directitonal	26
Data Bit 2	bi-directitonal	23
Data Bit 3	bi-directitonal	24
Data Bit 4	bi-directitonal	21
Data Bit 5	bi-directitonal	22
Data Bit 6	bi-directitonal	19
Data Bit 7	bi-directitonal	20
DIRC (bus dir)	drive	9
READY	drive	27
-STROBE	computer	29
-RESET	drive	31
+5 volts	drive	3,4,34
Ground drive		6,8,10,17,28,30,32
Unused	----	1,2,5,7,11-16,18,33

1.3 Cable timing

1.3.1 General case

Command initionation and computer to drive data transfer.

```
READY   ----------------+        +-------------------+        +-
                        |        |                   |        |
                        +--------+                   +--------+

-STROBE ----------+  +-------------------------+  +--------------
                  |  |                         |  |
                  +--+                         +--+

                /------------\               /------------\
DATA  ---------<              >-------------<              >----
                \------------/               \------------/

     ------------------------------------------------------------

DIRC
```

The drive indicates its readiness to accept a command by raising the READY line. The computer then puts a command byte to the data lines and pulses -STROBE (the command byte is to be latched by the drive on the rising edge of -STROBE). Upon seeing the -STROBE pulse, the drive drops the READY line as an acknowledgement to the computer. When ready for the next command byte the drive again raises the READY line.

At the end of the command sequence, the drive will keep the READY line low until the desired operation has been performed. Upon completion of the operation, the drive will lower the DIRC line, raise the READY line and then allow the computer to read data and status information. Note that all commands consist of a write phase (during which command and data information is sent to the drive), followed by a read phase (during which status and data information is received from the drive).

Drive to computer data transfer.

```
            +---------+          +---------+                    +-
            |         |          |         |                    |
READY   ----+         +----------+         +----------//-------+

-STROBE ------+  +-----------------+  +---------------//------
              |  |                 |  |
              +--+                 +--+

            /------------\       /------------\
DATA  -----<              >-----<              >------//----
            \------------/       \------------/

DIRC ----+                                                 +----
         |                                                 |
         +--------------------------------------------//---+
```

The drive starts a computer read sequence by lowering the DIRC line. The drive then puts a byte to the data lines and raises the ready line. The computer then pulses the -STROBE line, capturing the data on the rising edge. The drive then lowers the READY line until the next data byte is ready to send. After the last byte is transferred, the drive raises the DIRC line prior to raising the READY line.

1.3.2 Special conditions

There are two special conditions which deviate from the general cable timing information presented and must be accounted for by the computer/disk controller or by the computer/disk handler.

Case 1 -- READY line glitch after the last byte of command.

After the last command byte is received by the drive, the READY line will go high (for 20 uSEC. or less). Since this occurs prior to the completion of the command operation, it must be ignored. Since the glitch occurs while the DIRC line is high, it is easy to detect either in hardware (by gating) or in software (by the procedure shown below in Pascal pseudo-code).

```
REPEAT UNTIL (DIRC = LOW) AND (READY = HIGH );
```

Case 2 -- DIRC line glitches after last byte of Mirror command.

After the last command byte of a Mirror command is received, the DIRC line will repeatedly alternate between high and low (while the drive talks to the Mirror). Since these changes occur while the READY line is low, they are easy to detect either in hardware (by gating) or in software (by the procedure shown below in Pascal pseudo-code).

```
REPEAT UNTILL (READY = HIGH) AND ( DIRC = LOW);
```

Note that the two glitch cases are resolved with a single fix.

1.4 Cable connector description

17 x 2 female connector on cable, red stripe on cable is pin 1.

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| 1| 3| 5| 7| 9|11|13|15|17|19|21|23|25|27|29|31|33|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| 2| 4| 6| 8|10|12|14|16|18|20|22|24|26|28|30|32|34|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Pin 1 is normally designated by a square pin on the cicuit side of the interface card.

{This page left intentionally blank}

2. Disk Controller

2.1 System area

The first 2 cylinders on all drives are allocated as a system area, the second cylinder being a backup copy of the first. There are no spare tracks allowed in this region; all blocks must be good. The usage for the blocks within a cylinder are shown below.

Block 0 =	Boot Block.
Block 1 =	Disk parameter block. Spare track table (see 2.5.4) Interleave information. Step time Virtual drive track offset table (see 2.5.5).
Block 2 =	Diagnostic block.
Block 3 =	Constellation parameter block (see 2.5.3).
Blocks 4 through 5 =	Dispatcher code.
Blocks 6 through 7 =	Pipes and semaphores (see 2.5.3).
Blocks 8 through 17 =	Mirror controller code.
Blocks 18 and 19 =	LSI-11 controller code.
Blocks 20 and 21 =	Pipes controller code.
Blocks 22 through 39 =	Reserved for future use.
Blocks 40 through 59 =	Reserved for boot command.
Blocks 60 through remainder of cylinder =	Unused.

The paragraphs that follow provide brief descriptions of the content of each of the system area regions.

Boot block -- Contains Z-80 code.

Disk parameter block -- Contains disk related information as shown below:

```
+----------------------+
|  spare track table   |
|  (see 2.5.4)         |
+----------------------+
|  interleave factor   |
|  (default = 9)       |
+----------------------+
|  unused              |
+----------------------+
|  VDO table           |
|  (see 2.5.5)         |
+----------------------+
|  LSI-11 VDO table    |
|  (see 2.5.5)         |
+----------------------+
|  unused              |
+----------------------+
```

Diagnostic block -- This area contains code used by the Z-80 (in the controller) during diagnostic mode commands (format, verify, etc).

Constellation parameter block -- Contains multiplexer polling parameters and the pipe area definition, as shown below:

```
+----------------------+
|  multiplexer poll    |
|  parameters          |
+----------------------+
|  pipe area define    |
|  (see 2.5.3)         |
+----------------------+
|  unused              |
+----------------------+
```

Dispatcher code -- This area contains code used by the Z-80 (in the controller) during normal mode commands.

Pipes and semaphores -- This block contains code for the dispatcher and support utilities for pipes and semaphores, and also contains the semaphore table.

```
+----------------------+--+
| dispatcher code      |  |
| for pipes and        |  +-block 6
| semaphores           |  |
+----------------------+--+

+----------------------+--+
| semaphore table      |  |
| (see 2.5.2)          |  |
+----------------------+--+-block 7
| pipe and semaphore   |  |
| utilities code       |  |
+----------------------+--+
```

Mirror controller code -- xx.

LSI-11 controller code -- xx.

Pipes controller code -- xx.

Reserved area -- xx.

Boot extension -- Blocks 40 through 43 are currently used to support the Apple.

2.2 User area

The user area always starts at the third cylinder. The user area can be viewed as logical or physical sectors.

Logical sector numbers range from 0 to the size of the drive. The sizes are:

    11220 for the 6 Mbyte drive.
    21220 for the 10 Mbyte drive.
    38460 for the 20 Mbyte drive.

Physical sector numbers are given as head, cylinder, sector #.

The algorithm for converting logical sector numbers to physical sector numbers would be as shown below, if it were not for the system area, virtual devices and spare tracks (the real algorithm will be explained immediately following the simplified form):

```
disk sector # = block # modulo track size.
disk track # = block # div track size.
disk head # = disk track # modulo surfaces.
disk cylinder # = disk track # div surfaces.
```

Note that the disk track # is a temporary result and is not a directly addressable entity in the drive; a given block is addressed physically by sector #, head # and cylinder #.

The real algorithm for converting logical sector numbers to physical sector numbers is shown below:

```
disk sector # = block # modulo track size.
relative track # = block # div track size.
physical' track # = relative track # plus system area
                    offset plus virtual drive offset.
physical track # = physical' track # plus one for every
                    spare track preceding.'
disk head # = physical track # modulo surfaces.
disk cylinder # = physical track # div surfaces.
```

Where the following sizes apply:

SIZE	Model 6 Mb	Model 11 Mb	Model 20 Mb
Sectors/track	20	20	20
Surfaces (heads)	4	3	5
Cylinders	144	358	388
Total tracks per drive	576	1074	1940
Usable tracks per drive	561	1061	1923

2.3 Controller commands (numerical order)

2.3.1 Controller command notation

All of the controller commands are discribed in this section. The notation for each command is as follows: COMMAND NAME followed by (xxh : xxd), where xxh is the hex value of the command code, and where xxd is the equivalent decimal value of the same command code.

In some instances, a command code will consist of a primary code along with an additional command modifier. For these cases the notation is as follows: COMMAND NAME followed by (xxh,yyh : xxd,yyd), where xxh,yyh is the command code and the command modifier, respectively, and where and where xxd,yyd is the equivalent decimal value of the same command and command modifier.

2.3.2 Normal mode commands

2.3.2.1 Read sector (02h : 2d)

This command reads a 256 byte sector from the disk.

Send 4 bytes:

byte 1 = 02h (command).
byte 2 = logical drive #.
byte 3 = sector # (lsb).
byte 4 = sector # (msb).

Receive 257 bytes:

byte 1 = disk status.
byte 2-257 = sector data.

2.3.2.2 Write sector (03h : 3d)

This command writes a 256 byte sector to the disk.

Send 260 bytes:

byte 1 = 03h (command).
byte 2 = logical drive #.
byte 3 = sector # (lsb).
byte 4 = sector # (msb).
byte 5-260 = sector data.

Receive 1 byte:

byte 1 = disk status.

2.3.2.3 Get drive parameters (10h : 16d)

This command returns certain drive parameters.

Send 2 bytes:

byte 1 = 10h (command).
byte 2 = logical drive #.

Receive 129 bytes:

byte 1 =	status.
byte 2-32 =	ASCII text (31 bytes).
byte 33 =	firmware version.
byte 34 =	ROM version.

byte 35 = sectors/track.
byte 36 = tracks/cylinder.
byte 37 = cylinders/drive (lsb).
byte 38 = " (msb).
byte 39 = capacity of physical drive in 512 byte blocks (lsb).
byte 40 = capacity of physical drive in 512 byte blocks.
byte 41 = capacity of physical drive in 512 byte blocks (msb).
byte 42-57 = spare track list (see 2.5.4 for format).
byte 58 = interleave factor.
byte 59-70 = Constellation parameters.
byte 71-76 = pipe parameters (see 2.5.3 for format).
byte 77-90 = VDO table (see 2.5.5 for format).
byte 91-98 = LSI-11 VDO table (see 2.5.5 for format).
byte 99-106 = LSI-11 spare track list.
byte 107 = physical drive number.
byte 108 = capacity of logical drive in 512 byte blocks (lsb).
byte 109 = capacity of logical drive in 512 byte blocks.
byte 110 = capacity of logical drive in 512 byte blocks (msb).
byte 111-129 = filler.

2.3.2.4 Diagnostic mode select (11h : 17d)

This command takes the drive out of normal mode and sets it to diagnostic mode.

Send 514 bytes:

byte 1 = 11h (command).
byte 2 = logical drive #.
byte 3-514 = executable Z-80 code (execution starts at first byte). This code is the monitor for diagnostic mode which interprets the rest of the diagnostic mode commands. Normally, this block is the same as block 2 of the firmware.

Receive 1 byte:

byte 1 = disk status.

2.3.2.5 Read chunk (12h or 22h or 32h : 18d or 34d or 50d)

This command reads a 128, 256 or 512 byte "chunk" from the disk. The three read chunk command formats are shown below:

Send 4 bytes:

```
byte 1 = 12h (command).
byte 2 = logical drive #.
byte 3 = chunk # (lsb).
byte 4 = chunk # (msb).
```

Receive 129 bytes:

```
byte 1 = disk status.
byte 2-129 = data (128 bytes).
```

Send 4 bytes:

```
byte 1 = 22h (command).
byte 2 = logical drive #.
byte 3 = chunk # (lsb).
byte 4 = chunk # (msb).
```

Receive 257 bytes:

```
byte 1 = disk status.
byte 2-257 = data (256 bytes).
```

Send 4 bytes:

```
byte 1 = 32h (command).
byte 2 = logical drive #.
byte 3 = chunk # (lsb).
byte 4 = chunk # (msb).
```

Receive 513 bytes:

```
byte 1 = disk status.
byte 2-513 = data (512 bytes).
```

2.3.2.6 Write chunk (13h or 23h or 33h : 19d or 35d or 51d)

This command writes a 128, 256 or 512 byte "chunk" to the disk. The three write chunk command formats are shown below:

Send 132 bytes:

```
byte 1 = 13h (command).
byte 2 = logical drive #.
byte 3 = chunk # (lsb).
byte 4 = chunk # (msb).
byte 5-132 = data (128 bytes).
```

Receive 1 byte:

byte 1 = disk surface.

Send 260 bytes:

byte 1 = 23h (command).
byte 2 = logical drive #.
byte 3 = chunk # (lsb).
byte 4 = chunk # (msb).
byte 5-260 = data (256 bytes).

Receive 1 byte:

byte 1 = disk status.

Send 516 bytes:

byte 1 = 33h (command).
byte 2 = logical drive #.
byte 3 = chunk # (lsb).
byte 4 = chunk # (msb).
byte 5-516 = data (512 bytes).

Receive 1 byte:

byte 1 = disk status.

2.3.1.7 Boot (14h : 20d)

This command returns the contents of the specified sector of firmware on track #2.

Send 2 bytes:

byte 1 = 14h (command).
byte 2 = sector # (0-19).

Receive 513 bytes:

byte 1 = disk status.
byte 2-513 = boot data (512 bytes).

2.3.3 Diagnostic mode commands

2.3.2.1 Reset drive (00h : 0d)

This command takes the drive out of diagnostic mode and sets it in normal mode.

Send byte :

Byte 1 = 00h (command).

Receive 1 byte:

Byte 1 = 0

2.3.3.2 Format drive (01h : 1d)

This command formats a drive if the FORMAT switch is ON, else returns an error status.

Send 513 bytes:

byte 1 = 01h (command).
byte 2-513 = format pattern data (512 bytes).

Receive 1 byte:

byte 1 = disk status.

2.3.3.3 Verify (07h : 7d)

This command performs a CRC check of every sector on the disk.

Send 1 byte.

byte 1 = 07h (command).

Receive n*4+2 bytes (n = errors):

byte 1 = status.
byte 2 = number of bad sectors * 4.
byte 3 = head number of 1st bad sector.
byte 4 = cylinder of 1st bad sector (lsb).
byte 5 = cylinder of 1st bad sector (msb).
byte 6 = sector number of 1st bad sector.
.
.
byte n*4-1 = head number of nth bad sector.
byte n*4+0 = cylinder of nth bad sector (lsb).
byte n*4+1 = cylinder of nth bad sector (msb).
byte n*4+2 = sector number of nth bad sector.

2.3.3.4 Read Corvus firmware (32h : 50d)

This command reads a block of data from the system area.

Send 2 bytes:

```
        byte 1 = 32h (command)
        byte 2 + head (3 bits), sector (5 bits).
```

Receive 513 bytes:

```
        byte 1 = disk status.
        byte 2-513 = contents of block (512 bytes).
```

2.3.3.5 Write Corvus firmware (33h : 51d)

This command writes a block of data to the system area.

Send 514 bytes:

```
        byte 1 = 33h (command).
        byte 2 = head (3 bits), sector (5 bits).
        byte 3-514 = data (512 bytes).
```

Receive 1 byte:

```
        byte 1 = disk status.
```

2.3.4 Semaphore Commands

The principal reason for using semaphores is to avoid a situation where two or more users are simultaneously accessing the same volume.

There is no problem if two users are merely reading from the same volume. However, if one user is writing to a volume, another user simultaneously accessing that volume may cause inconsistant data to be read. A more serious problem occurs if multiple users are writing to a file or volume at the same time.

This problem arises because the operating system in each processor has a copy of the directory for each active disk volume. The directory is usually updated on the disk only when the local operating system thinks it is necessary. Since each user can be adding, deleting, or changing files, the directory may be different in two or more processor's memory. This leads to two users writing out their files or directories and only the last user to write actually updating the directory on the disk.

To avoid this problem, there are several alternatives useful in specific instances. Read-only access to system utilities or data bases avoids the problem on shared disks. Read-write access to shared volumes can be made safe if all writes are made to existing pre-allocated files and the file is locked while any program has write access to it.

Semaphores can be used to keep two or more programs from writing to the same file or section of a file at the same time. User application programs that need shared read-write access to a data base can be configured to test the status of a semaphore before allowing access to a file. The semaphore is used to indicate that a particular file is being written to.

Each processor may, at any time, request to lock a semaphore. The request is granted if no other processor has already locked that particular semaphore. The label for the semaphore can be any eight character name that is agreed upon by the programs that wish to share access.

The Lock and Unlock commands send an eight byte name, called the semaphore, that is either placed into or removed from the semaphore table managed by the Corvus disk controller. If the semaphore table is full or if a semaphore has already been entered, a locked semaphore status is returned. The application program using the semaphores can continue to poll the semaphore table until a space is available or the desired semaphore is no longer locked. The status of the semaphore prior to each operation is also returned to provide for a full test-set or test-clear operation.

The semaphore table can be initialized by any processor, but this should only be performed on system-wide initialization or for recovery from error conditions.

2.3.4.1 Semaphore Initialize (1Ah,10h : 26d,16d)

For command explanation see the table above.

Send 5 bytes:

byte 1 = 1Ah (command).
byte 2 = 10h (command modifier).
byte 3-5 = filler.

Receive 1 byte:

byte 1 = disk status.

2.3.4.2 Semaphore lock (0Bh,01h : 11d,1d)

For command explination see the table above.

Send 10 bytes:

byte 1 = 0Bh (command).
byte 2 = 01h (command modifier).
byte 3-10 = semiphore key (8 byte name).

Receive 2 bytes:

byte 1 = disk status.
byte 2 = semaphore status.

2.3.4.3 Semaphore unlock (0Bh,11h: 11d,17d)

Send 10 bytes:

byte 1 = 0Bh (command).
byte 2 = 11h (command modifier).
byte 3-10 = semaphore key (8 byte name).

Receive 2 bytes:

byte 1 = disk status.
byte 2 = semaphore status.

2.3.4.4 Semaphore status (1Ah,41h : 26d,65d)

Send 5 bytes

byte 1 = 1Ah (command).
byte 2 = 41h (command modifier).
byte 3 = 03h (command modifier).
byte 4-5 = filler (0's).

Receive 257 bytes:

byte 1-disk status.
byte 2-257 = semaphore table (256 bytes).

See section 2.5.2 for the format of the semaphore table.

2.3.5 Pipe commands

The Corvus disk controller provides a method, called Pipes, by which different computers or programs can send data to each other. A Pipe is a FIFO (first-in-first-out) buffer that is written by a sender and is read by a receiver. Pipe commands control writing data to and reading data from the FIFO buffer. Senders and receivers may be different programs on different computers (with the Constellation network) running at different times. The only restriction on the sender/receiver combination is that the sender must send all data before the data is available to the receiver.

Before a Pipe can be utilized, it must be opened for write. The program that is sending data issues an Open Write command which creates, names, and gives a number to a Pipe.

After the Pipe is successfully opened for writing, the Pipe is ready to receive data. Pipe Write commands are used to write data to the Pipe. The Pipe Write command contains the Pipe number returned by the Open Write command. A maximum of 512 bytes may be written with one Pipe Write command.

After all the desired data has been written to a Pipe, a Close Write command is issued. The Close Write command closes a Pipe for writing and makes the Pipe available for reading.

A Pipe cannot be read until it has been written in the sequence described above. To read a Pipe, an Open Read command is issued which opens the specified Pipe for reading.

After the Pipe is successfully opened for reading, the Pipe is ready to transmit data. Pipe Read commands are used to read data from the Pipe. The Pipe Read command contains the Pipe number returned by the Open Read command. A maximum of 512 bytes may be read with one Pipe Read command.

After all the data from the Pipe has been read, a Close Read command is issued. The Close Read command closes a Pipe for reading. If all the data from a Pipe has been read when it is closed for read, the resources allocated for that Pipe are released and may be used by other Pipes.

The Pipe Initialization command initializes a Pipes area on the disk. It contains the starting disk block number and the number of disk blocks to allocate for Pipe processing.

The Purge Pipe command is used to purge unwanted Pipes by Pipe number.

The Pipe Status command returns two data blocks (512 bytes each). The first data block contains a name table of active Pipes. The second block is the pointer table, which contains state information and pointers for both ends of each active Pipe.

In a Corvus network, Pipes provide a general communications mechanism that can be used to build more sophisticated network applications. Pipes can serve as a utility that enables different computers connected to the same Corvus disk system to communicate with each other or share common peripheral equipment.

2.3.5.1 Pipe read (1Ah,20h : 26d,32d)

Send 5 bytes

byte 1 = 1Ah (command).
byte 2 = 20h (command modifier).
byte 3 = pipe number from open command (1-62).
byte 4 = 0.
byte 5 = 2.

Receive 516 bytes

byte 1 = disk status
byte 2 = pipe status
byte 3 = length of data returned (lsb).
byte 4 = length of data returned (msb).
byte 5-516 = data (512 bytes)

2.3.5.2 Pipe write (1Ah,21h : 16d,33d)

Send 5 + data length bytes

byte 1 = 1Ah (command).
byte 2 = 21h (command modifier).
byte 3 = pipe number from the open command (1-62).
byte 4 = length of data actually written (lsb).
byte 5 = length of data actually written (msb).
byte 6-n = data.

Receive 12 bytes.

byte 1 = disk status.
byte 2 = pipe status.
byte 3 = length of data actually written (lsb).
byte 4 = length of data actually written (msb).
byte 5-12 = filler.

2.3.5.3 Pipe close (1Ah,40h : 26d,64d)

Send 5 bytes:

byte 1 = 1Ah (command).
byte 2 = 40h (command modifier).

byte 3 = pipe number from the open command (1-62).
byte 4 = action code.
byte 5 = filler.

Receive 12 bytes.

byte 1 = disk status.
byte 2 = pipe status.
byte 3-12 = filler.

2.3.4.4 Pipe status (1Ah,41h : 26d,65d)

Send 5 bytes:

byte 1 = 1Ah (command).
byte 2 = 41h (command modifier).
byte 3 = 1 for name table status (read 512 bytes).
         2 for pipe pointer table (read 512 bytes).
         0 for both of above (read 1024 bytes).
byte 4-5 = filler (0's).

Receive 513 or 1025 bytes:

byte 1 = disk status.
byte 2-513 = name table status or pipe pointer table.
byte 514-1025 = pipe pointer table, if specified.

See section 2.5.3 for the formats for the pipe tables.

2.3.5.5 Pipe open write (1Bh,80h : 27d,128d)

Send 10 bytes:

byte 1 = 1Bh (command).
byte 2 = 80h (command modifier).
byte 3-10 = pipe name (8 bytes).

Receive 12 bytes:

byte 1 = disk status.
byte 2 = pipe status.
byte 3 = pipe number assigned (1-62).
byte 4 = pipe state.
byte 5-12 = filler.

2.3.5.6 Pipe area initialize (1Bh,A0h : 27d,160d)

Send 10 bytes:

byte 1 = 1Bh (command).
byte 2 = A0h (command modifier).
byte 3 = pipe area disk block number (lsb).
byte 4 = pipe area disk block number (msb).
byte 5 = pipe area size -- number of blocks (lsb).
byte 6 = pipe area size -- number of blocks (msb).
byte 7-10 = filler.

Receive 12 bytes:

byte 1 = disk status.
byte 2 = pipe status.
byte 3-12 = filler.

2.3.5.7 Pipe open read (1Bh,C0h : 27d,192d)

Send 10 bytes:

byte 1 = 1Bh (command).
byte 2 = C0h (command).
byte 3-10 = pipe name (8 bytes).

Receive 12 bytes.

byte 1 = disk status.
byte 2 = pipe status.
byte 3 = pipe number assigned (1-62).
byte 4 = pipe state.
byte 5-12 = filler.

2.4 Controler status codes

2.4.1 Normal mode command status codes

Error codes returned by the Corvus disk controller contain the type of error and error severity. Error severity is coded as follows:

Bit 7 set = Fatal error
Bit 6 set = Verify error
Bit 5 set = Recoverable error

Disk Status Codes

Non-fatal				Fatal (>= 128)						
Recov-erable Error		Verify Error				Recov-erable Error		Verify Error		
dc	hx	dc	hx	dec	hx	dec	hx	dec	hx	
32	20	64	40	128	80	160	A0	192	C0	Header fault
33	21	65	41	129	81	161	A1	193	C1	Seek timeout
34	22	66	42	130	82	162	A2	194	C2	Seek fault
35	23	67	43	131	83	163	A3	195	C3	Seek error
36	24	66	44	132	84	164	A4	196	C4	Header CRC error
37	25	67	45	133	85	165	A5	197	C5	Rezero fault
38	26	68	46	134	86	166	A6	198	C6	Rezero timeout
39	27	69	47	135	87	167	A7	199	C7	Drive not online
40	28	70	48	136	88	168	A8	200	C8	Write fault
41	29	71	49	137	89	169	A9	201	C9	--
42	2A	72	4A	138	8A	170	AA	202	CA	Read data fault
43	2B	73	4B	139	8B	171	AB	203	CB	Data CRC error
44	2C	74	4C	140	8C	172	AC	204	CC	Sector locate error
45	2D	75	4D	141	8D	173	AD	205	CD	Write protected
46	2E	76	4E	142	8E	174	AE	206	CE	Illegal sector address
47	2F	77	4F	143	8F	175	AF	207	CF	Illegal command op code
48	30	78	50	144	90	176	B0	208	D0	Drive not acknowledged
49	31	79	51	145	91	177	B1	209	D1	Acknowledge stuck active
50	32	80	52	146	92	178	B2	210	D2	Timeout
51	33	81	53	147	93	179	B3	211	D3	Fault
52	34	82	54	148	94	180	B4	212	D4	CRC
53	35	83	55	149	95	181	B5	213	D5	Seek
54	36	84	56	150	96	182	B6	214	D6	Verification
55	37	85	57	151	97	183	B7	215	D7	Drive speed error
56	38	86	58	152	98	184	B8	216	D8	Drive illegal address error
57	39	87	59	153	99	185	B9	217	D9	Drive r/w fault error
58	3A	88	5A	154	9A	186	BA	218	DA	Drive servo error
59	3B	89	5B	155	9B	187	BB	219	DB	Drive guard band
60	3C	90	5C	156	9C	188	BC	220	DC	Drive PLO error
61	3D	91	5D	157	9D	189	BD	221	DD	Drive r/w unsafe

2.4.2 Diagnostic mode disk status codes

2.4.3 Semaphore command status codes

SEMAPHORE STATUS CODES

DECIMAL	HEX	MEANING
0	00	Prior semaphore state = not set.
128	80	Prior semaphore state = set.
253	FD	Semaphore table full.
254	FE	Disk error.

2.4.4 Pipe command status codes

PIPE STATUS CODES

DECIMAL	HEX	MEANING
0	00	Successful pipe request.
8	08	Tried to read an empty pipe.
9	09	Pipe was not open for read or write.
10	0A	Tried to write to a full pipe.
11	0B	Tried to open an open pipe.
12	0C	Pipe does not exist.
13	0D	No room for new pipe.
14	0E	Illegal command.
15	0F	Pipe area not initialized.

PIPE STATE CODES

DECIMAL	HEX	MEANING
1	01	Open for write, file empty.
2	02	Open for read, file empty.
128	80	Full, not open.
129	81	Full, open for write.
130	82	Full, open for read.

2.5 Controller theory of operation

2.5.1 Disk operations

2.5.1.1 CRC operations

On a data read, if the first try produces no CRC error the data is returned to the computer and no further action is taken. However, if the first try produces a CRC error then one of two things will happen: 1) if the data is read successfully within 10 tries then the data is rewritten to the disk and a soft error is reported or 2) if the data cannot be read successfully within 10 tries then the data read on the last try is rewritten to the disk (along with a new CRC) and a hard error is reported.

2.5.1.2 Format operation

2.5.2 Semaphores

Semaphores provide a method for communicating between independent programs and/or systems. The disk controller provides for up to 32 named semaphores, each key (name) being from 1 to 8 characters in length.

The semaphores are implemented using a lookup table containing an 8 byte entry for each of the 32 possible semaphore keys. The presence of a key indicates that the semaphore is locked, and the absence of a key indicates that the semaphore is unlocked. Unused table entries (and unlocked semaphores) are represented by 8 bytes of blank ASCII code (20h).

The format of the semaphore table on disk (block 7) is shown below:

```
+-----------+  byte 1
| key #1    |
+-----------+
| key #2    |
+-----------+
|           |
=           =
|           |
+-----------+
| key #31   |
+-----------+
| key #32   |
+-----------+  byte 256
```

Each of the key entries has the form shown below:

```
+--------------+
|   1st byte   |  relative byte 1
+-            -+
|   2nd byte   |
+-            -+
|   3rd byte   |
+-            -+
|   4th byte   |
+-            -+
|   5th byte   |
+-            -+
|   6th byte   |
+-            -+
|   7th byte   |
+-            -+
|   8th byte   |  relative byte 8
+--------------+
```

2.5.3 Pipes

There is a 6 byte region in the Constellation parameter block (see section 2.1) which provides pipe parameters, specifically a pipe area definition. The format for the pipe parameters is shown below:

```
+-------------------+
| block # of (lsb)  |
+-  pipe names     -+
|   table     (msb) |
+-------------------+
| block # of (lsb)  |
+-  pipe pointer   -+
|   table     (msb) |
+-------------------+
| number of (lsb)   |
+-  blocks in the  -+
|   pipes area (msb)|
+-------------------+
```

The three pipe parameters are intially set to 1111h, 2222h and 3333h,, which indicates an uninitialized pipe area. The pipe area may be defined by the user using the Pipe Initialize command (section 2.3.5.6).

The format of the pipe area is shown below:

```
        +-----------------+
        |   pipe names    |  1 block
        |   table         |
        +-----------------+
        |   pipe pointer  |  1 block
        |   table         |
        +-----------------+
        |   pipe data     |
        |   area          |
        =                 =  n blocks
        |                 |
        |                 |
        +-----------------+
```

The pipe names table contains 64 entries of 8 bytes each. The first and last names in the table are reserved for system use. The first name is "WOOFWOOF" and the last name is "FOOWFOOW".

The pipe pointer table also contains 64 entries of 8 bytes each, each entry being formatted as shown below:

```
        byte 1 = pipe number.
        byte 2 = starting byte address (lsb).
        byte 3 = starting byte address.
        byte 4 = starting byte address (msb).
        byte 5 = ending byte address (lsb).
        byte 6 = ending byte address.
        byte 7 = ending byte address (msb).
        byte 8 = pipe status (see 2.4.4).
```

Individual pipe disk space allocation

Definitions:

Active hole -- a contiguous aea of unused disk space bounded on the low address end by an open for writing pipe.

```
+-----------------+
|    open for     |
|    writing      |
|    pipe         |
+-----------------+
|    active       |   the open pipe in front of the hole
|    hole         |   can grow into this region.
+-----------------+
|    pipe         |
+-----------------+
```

Inactive hole -- a contiguous area of unused disk space bounded on the low address end by the pipe area limit, the end of a closed pipe or the end of an open for reading pipe.

```
+-----------------+
|    open for     |
|    reading or   |
|    closed pipe  |
+-----------------+
|    inactive     |   the pipe in front of the hole
|    hole         |   cannot grow into this region.
+-----------------+
|    pipe         |
+-----------------+
```

New pipe allocations are made by first examining all of the holes in the pipe area. The allocator looks for the larger of: 1) the largest inactive hole or 2) 1/2 the size of the largest active hole. A new pipe starts at the beginning of an inactive hole or at the midpoint of an active hole. All pipes grow in the same direction, by increasing address.

2.5.4 Spare tracks

There is a 16 byte region in the disk parameter block (see section 2.2.3.2) which provides for the sparing of up to 7 tracks. The format for the spare track list is shown below:

```
+--------------------+
|  track number (lsb)|
+-  of 1st          -+
|  spare track (msb) |
+--------------------+
|  track number (lsb)|
+-  of 2nd          -+
|  spare track (msb) |
+--------------------+
|                    |
+-                  -+
|                    |
+--------------------+
|  track number (lsb)|
+-  of 7th          -+
|  spare track (msb) |
+--------------------+
|  FFh          end  |
+-              of  -+
|  FFh         list  |
+--------------------+
```

The first entry with a track number equal to FFFFh will indicate the logical end of the list.

2.5.5 Virtual drives

There is a 14 byte region in the disk parameter block (see section 2.1) which provides for the definition of up to 7 virtual (logical) drives. The format for the virtual drive list is shown below:

```
        +--------------------+
        |  track offset (lsb)|
        +-  of 1st virtual  -+
        |  drive        (msb)|
        +--------------------+
        |  track offset (lsb)|
        +-  of 2nd virtual  -+
        |  drive        (msb)|
        +--------------------+
        |  track offset (lsb)|
        +-  of 2nd virtual  -+
        |  drive        (msb)|
        +--------------------+
        |  track offset (lsb)|
        +-  of 2nd virtual  -+
        |  drive        (msb)|
        +--------------------+
```

An entry with a track offest equal to FFFh will indicate the absence of the corresponding virtual drive.

3.1 General

The Corvus Systems MIRROR is an inexpensive interface that adds the capability to provide backup and archival storage for the Corvus disk system. This data formatting interface converts data from a digital signal on the disk to a video signal that can be recorded on a standard video cassette recorder (VCR) at the Standard Play (SP) speed. The MIRROR is compatible with all present hardware and software -- all programs and peripherals that work with the Corvus disk system will work with the MIRROR installed.

The MIRROR allows over 100 megabytes of storage on an inexpensive, removable, and transportable media, a video cassette tape.

Redundancy and CRC error detection assure the ability to recover data. Because of redundancy and built in error checking, it is possible to recover data reliably even when errors are encountered that could not be recovered on conventional tape storage media. The result is reliable backup of mass storage. This method generally produces a few soft errors during the backup process. An error may occur in one block of a set of multiple blocks, however, by having multiple copies of each block a single good block can normally be restored.

When data is being restored to the disk, the MIRROR uses the redundant blocks to reconstruct a good block of data.

With the MIRROR, the user can make a video tape copy of an entire Corvus disk, a virtual device, or a single file (contiguous area on the disk). In approximately fifteen minutes, the contents of an entire ten million byte disk can be transferred to a standard video cassette.

The normal format creates four images of each block being backed up. Since there are four images of each block, the possibility of unrecoverable errors is minimal.

3.2 Mirror functional description

backup
restore
redundant recording
error checking
high speed search

3.3 Mirror commands (numerical order)

3.3.1 Mirror command notation

All of the Mirror commands are discribed in this section. The notation for each command is as follows: COMMAND NAME followed by (xxh : xxd), where xxh is the hex value of the command code, and where xxd is the equivalent decimal value of the same command code.

In some instances, a command code will consist of a primary code along with an additional command modifier. For these cases the notation is as follows: COMMAND NAME followed by (xxh,yyh : xxd,yyd), where xxh,yyh is the command code and the command modifier, respectively, and where and where xxd,yyd is the equivalent decimal value of the same command and command modifier.

3.3.2 Backup (08h : 8d)

Send 520 bytes:

```
        byte 1 = 08h (command).
        byte 2 = logical drive number.
        byte 3 = image I.D.
        byte 4 = number of 512 byte blocks to backup (lsb).
        byte 5 = number of 512 byte blocks to backup (msb).
        byte 6 = number of first block to backup (lsb).
        byte 7 = number of first block to backup (msb).
        byte 8 = format type: 0 = fast, 1 = normal, 2 = compatible.
                                                  (for 6 MB drive).
        byte 9-520 = user defined header (512 bytes).
```

Receive 2 bytes:

```
        byte 1 = disk status.
        byte 2 = number of disk read errors, if byte 1 < 80h;
                 Mirror status, if byte 1 = FFh;
```

3.3.3 Restore (09h : 9d)

Send 8 bytes:

```
        byte 1 = 09h (command).
        byte 2 = logical drive number.
        byte 3 = image I.D.
        byte 4 = number of 512 byte blocks to restore (lsb).
        byte 5 = number of 512 byte blocks to restore (msb).
        byte 6 = number of first block to restore (lsb).
        byte 7 = number of first block to restore (msb).
        byte 8 = filler.
```

Receive 2 bytes:

```
      byte 1 = disk status.
      byte 2 = number of disk write errors, if byte 1 < 80h;
               Mirror status, if byte 1 = FFh;
```

3.3.4 Identify (0Ah,00h : 10d,0d)

Send 4 bytes:

```
      byte 1 = 0Ah (command).
      byte 2 = 00h (comand modifier).
      byte 3 = image I.D. to read: 0 = next header, else as
               specified.
      byte 4 = 0.
```

Receive 516 bytes

```
      byte 1 = disk status.
      byte 2 = image I.D., if byte 1 = 0;
               unused, if byte <> 0.
      byte 3 = number of blocks for image (lsb).
      byte 4 = number of blocks for image (msb).
      byte 5-516 = image header (512 bytes).
```

3.3.5 Verify (0Ah,01h : 10d,1d)

Send 4 bytes:

```
      byte 1 = 0Ah (command).
      byte 2 = 01h (comand modifier).
      byte 3 = image I.D. to verify.
      byte 4 = 0
```

Receive 2 bytes

```
      byte 1 = disk status.
      byte 2 = number of disk read errors, if byte 1 < 80h;
               Mirror status, if byte 1 = FFh;
```

3.3.6 Verify error report (0Ah,02h : 10d,2d)

Send 4 bytes:

```
         byte 1 = 0Ah (command).
         byte 2 = 02h (command modifier).
```

byte 3 = 0
byte 4 = 0

Receive 5 + 2 * hard errors bytes:

byte 1 = number of soft errors (lsb).
byte 2 = number of soft errors (msb).
byte 3 = number of CRC failures.
byte 4 = number of disk verify errors.
byte 5 = number of hard errors.
byte 6-n = hard error block numbers (lsb,msb).

3.3.7 Remote operation select (0Ah,04h : 10d,4d)

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 04h (command modifier).
byte 3 = operation code (see table below).
byte 4 = 0.

Receive 1 byte:

byte 1 = command status.

Operations codes:

0 = J3 pin 2 (PLAY) pulsed low.
1 = J3 pin 3 (FAST FORWARD) pulsed low.
2 = J3 pin 4 (REWIND) pulsed low.
3 = J3 pin 5 (STOP) pulsed low.
14 = J3 pin 1 (RECORD) is set high.
15 = J3 pin 1 (RECORD) is set low.

3.3.7.1 Remote status (0Ah,05h : 10d,5d)

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 05h (command modifier).
byte 3 = 0.
byte 4 = 0.

Receive one byte:

byte 1 = status (see table below).

Status bits (0 is lsb, 7 is msb):

bit 0 = CRC generator status; 0 = no error, 1 = error.
bit 1 = unused.
bit 2 = unused.
bit 3 = unused.
bit 4 = J3 pin 14 (REWIND status); 1 = tape rewinding.
bit 5 = unused.
bit 6 = J3 pin 13 (FRAME SYNC); 1 pulse per every 2 frames.
bit 7 = J3 pin 11 (START OF TAPE); 0 = start of tape.

3.3.7.2 Verify retry (0Ah,06h : 10d,6d)

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 06h (command modifier).
byte 3 = image I.D. to verify.
byte 4 = 0.

Receive 2 bytes:

byte 1 = disk status.
byte 2 = number of tape read errors, if byte 1 < 80h;
Mirror status, if byte 1 = FFh;

3.3.7.3 Jump forward (0Ah,07h : 10d,7d)

Requires a model NV8200 Panasonic VCR and remote option.

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 07h (command modifier).
byte 3 = number of blocks to jump / 256 (lsb).
byte 4 = number of blocks to jump / 256 (msb).

Receive 1 byte:

byte 1 = 0.

3.3.7.4 Jump reverse (0Ah,08h : 10d,8d)

Requires a model NV8200 Panasonic VCR and remote option.

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 08h (command modifier).

byte 3 = number of blocks to jump / 256 (lsb).
byte 4 = number of blocks to jump / 256 (msb).

Receive 1 byte:

byte 1 = 0.

3.3.7.5 Find present location (0Ah,09h : 10d,9d)

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 09h (command modifier).
byte 3 = 0.
byte 4 = operation code (see table with Remote operation command, section 3.36).

Receive 8 bytes:

byte 1 = disk status.
byte 2 = image I.D.
byte 3 = image format (0-2).
byte 4 = block type found: F8h = image header, F1 = image trailer, F6h,06h = data bloc
byte 5 = block number (lsb).
byte 6 = block number (msb).
byte 7 = image size in blocks (lsb).
byte 8 = image size in blocks (msb).

3.3.7.6 Find image trailer (0Ah,0Ah : 10d,10d)

Send 4 bytes:

byte 1 = 0Ah (command).
byte 2 = 0Ah (command modifier).
byte 3 = 0.
byte 4 = 0.

Receive 2 bytes:

byte 1 = disk status.
byte 2 = image I.D.

3.3.8 Restore retry (0Ch,00h : 12d,0d)

Send 4 bytes:

byte 1 = 0Ch (command).
byte 2 = logical drive number.
byte 3 = 00h (command modifier).
byte 4 = filler.

Receive 2 bytes:

byte 1 = disk status.
byte 2 = number of disk read errors, if byte 1 = 00h;
Mirror status, if byte 1 = FFh.

3.3.9 Error report for backup, restore, verify, retry (0Ch,01h : 12d,1d)

Send 4 bytes:

byte 1 = 0Ch (command).
byte 2 = logical drive number.
byte 3 = 01h (command modifier).
byte 4 = filler.

Receive 5 + 2 * hard errors bytes:

byte 1 = number of soft errors (lsb).
(recovered errors / rebuild attempts)
byte 2 = number of soft errors (msb).
(recovered errors / search misses)
byte 3 = number of CRC failures.
(tape read errors / rebuild failures)
byte 4 = number of disk verify errors.
(disk write errors)
byte 5 = number of hard errors.
(disk read errors / bad blocks)
byte 6-n = hard error block numbers (lsb,msb).

3.3.10 Partial restore (0Dh : 13d)

Send 10 bytes:

byte 1 = 0Dh (command).
byte 2 = logical drive number.
byte 3 = image I.D.
byte 4 = number of 512 byte blocks to restore (lsb).
byte 5 = number of 512 byte blocks to restore (msb).
byte 6 = destination of first block to restore (lsb).
byte 7 = destination of first block to restore (msb).
byte 8 = offset within image (lsb).
byte 9 = offset within image (msb).
byte 10 = filler.

Receive 2 bytes:

byte 1 = disk status.
byte 2 = number of disk read errors, if byte 1 = 80h;
         Mirror status, if byte 1 = FFh.

3.4 Mirror status code

MIRROR STATUS CODES

DECIMAL	HEX	MEANING
0	00	Successful Mirror request.
1	01	Image I.D. mismatch.
2	02	Illegal restore command.
3	03	Illegal retry command (retry not enabled).
4	04	Image size mismatch.
5	05	Illegal opcode.
7	07	Start of image not found (30 second timeout).
8	08	Position error.
134	86	Tape dropout duirng playback operation (5 second timeout).

3.5 Mirror theory of operation

```
+----------+     +----------+     +----------+
| Computer |     |          |     |   Disk   |
|    or    +-----+  Mirror  +-----+ Controler|
|   Mux    |     |          |     |          |
+----------+     +----+-----+     +----------+
                      |
                 +----+-----+
                 |          |
                 |   Vcr    |
                 |          |
                 +----------+
```

VCR cable (j3) description

The control cable that connects the Mirror to the VCR has the command and status lines below:

pin 1 = RECORD low (pulse).
pin 2 = PLAY low (pulse).
pin 3 = FAST FORWARD low (pulse).

```
pin 4 = REWIND low (pulse).
pin 5 = STOP low (pulse).
pin 11 = START OF TAPE status.
pin 13 = FRAME SYNC status.
pin 14 = REWIND status.
```

Format of a tape frame

Images on tape consist of a number of frames, each frame corresponding to one commmplete TV picture scan. Frames are recorded on tape at a rate of 60 per second, and have the general format shown below.

```
+---------------+
|     sync      |
+---------------+
|  image I.D.   |
+---------------+
|               |  F8h = image header,, F1h = image
|  image code   |  trailer, F6h or 06h = image
|               |  data.
+---------------+
|  image size   |  number of disk blocks.
+---------------+
|  rel block #  |  within image.
+---------------+
|     data      |
=               =  512 bytes.
|  portion      |
+---------------+
|     CRC       |
+---------------+
|  CRC reset    |
+---------------+
```

Format of a tape image

Each image on tape is comprised of groups of the three frame types (header, data and trailer) as shown below:

```
+----------------------------------+
|  image     |  image     |  image     |
|  header    |  data      |  trailer   |
|  frames    |  frames    |  frames    |
+----------------------------------+
```

Image header

The image header consists of approximately 4 seconds of header frames (240 frames), with the data portion of each frame containing the 512 byte user I.D.

Image data

The image data is recorded in one of two formats: slow (4 copies of each disk block) or fast (2 copies of each disk block). For both formats, a given data frame contains two copies each of three disk blocks, as shown below:

```
+---------------+
|   block m     |
|   block m     |
|   block m+1   |
|   block m+1   |
|   block m+2   |
|   block m+2   |
+---------------+
```

Slow format data frames are grouped as shown below:

```
    +---------------+
    | blocks m      |  2 copies of each block.
    | thru m+2      |
    +---------------+
    | blocks m      |  2 copies of each block (4 total).
    | thru m+2      |
    +---------------+
    | blank         |  3 to n of these frames, as necessitated
    | blocks        |  by disk timing.
    +---------------+
    | blocks m+3    |
    | thru m+5      |
    +---------------+
    | blocks m+3    |
    | thru m+5      |
    +---------------+
    | blank         |
    | blocks        |
    +---------------+
    =               =
    +---------------+
    | blocks n-2    |
    | thru n        |
    +---------------+
    | blocks n-2    |
    | thru n        |
    +---------------+
    | blank         |
    | blocks        |
    +---------------+
```

Fast format data frames are grouped as shown below:

```
+---------------+
| blocks m      |   2 copies of each block.
| thru m+2      |
+---------------+
| blocks m+3    |   2 copies of each block
| thru m+5      |
+---------------+
| blank         |   3 to n of these frames, as necessitate
| blocks        |   by disk timing.
+---------------+
| blocks m+6    |
| thru m+8      |
+---------------+
| blocks m+9    |
| thru m+10     |
+---------------+
| blank         |
| blocks        |
+---------------+
=               =
+---------------+
| blocks n-5    |
| thru n-3      |
+---------------+
| blocks n-2    |
| thru n        |
+---------------+
| blank         |
| blocks        |
+---------------+
```

Image trailer

The image trailer consists of approximately 2 seconds of trailer frames (120 frames).

Format of images on tape (archival Mirror)

Images (each of which consists of many frames) are stored on tape sequentially, as shown below.

```
+-------------------------------------||----------------------+
|  directory   |  image   |  image   |        |  image  | image |
|   (image     |   #1     |   #2     |        |  #n-1   |  #n   |
|    #0)       |          |          |        |         |       |
+-------------------------------------||----------------------+
```

The directory is maintained by external software and the directory data is read and written using the same commands as any other image. The directory contains 16 bytes of information for each of up to 32 images, as shown below:

```
+-----------+
|   date    |  2 bytes.
+-----------+
|   size    |  2 bytes.
+-----------+
|   name    |  12 bytes.
+-----------+
```

{This page left intentionally blank}

Appendix A

DISK COMMAND SUMMARY

Command	Code:Modifier	Number of Data Bytes Sent	 Received
Normal Mode Commands:			
Read Sector	02	4	257
Write Sector	03	260	1
Get Drive Parameters	10	2	129
Diagnostic Mode Select	11	514	1
Read Chunk (128 Bytes)	12	4	129
Read Chunk (256 Bytes)	22	4	257
Read Chunk (512 Bytes)	32	4	513
Write Chunk (128 Bytes)	13	132	1
Write Chunk (256 Bytes)	23	260	1
Write Chunk (512 Bytes)	33	516	1
Boot	14	2	513
Diagnostic Mode Commands:			
Reset Drive	00	1	1
Format Drive	01	513	1
Verify	07	1	4n+2
Read Corvus Firmware	32	2	513
Write Corvus Firmware	33	514	1
Semaphore Commands:			
Semaphore Initialize	1A:10	5	1
Semaphore Lock	0B:01	10	2
Semaphore Unlock	0B:11	10	2
Semaphore Status	1A:41	5	257
Semaphore Initialize (Rev A)	10:0A	5	12
Pipe Commands:			
Pipe Read	1A:20	5	516
Pipe Write	1A:21	x+5	12
Pipe Close	1A:40	5	12
Pipe Status 1	1A:41	5	513
Pipe Status 2	1A:41	5	513
Pipe Status 0	1A:41	5	1025
Pipe Open Write	1B:80	10	12
Pipe Area Initialize	1B:A0	10	12
Pipe Open Read	1B:C0	10	12

Mirror Commands:

Backup	08	520	2
Restore	09	8	2
Read Image Header	0A:00	4	516
Verify Image	0A:01	4	2
Report Errors After Verify	0A:02	4	n
Remote Operation	0A:04	4	1
Remote Status	0A:05	4	1
Verify Retry	0A:06	4	2
Jump Forward	0A:07	4	1
Jump Reverse	0A:08	4	1
Find Present Location	0A:09	4	8
Find Image Trailer	0A:0A	4	2
Restore Retry	0C:00	4	2
Restore Error Report	0C:01	4	1
Partial Restore	0D	10	2

n = number of errors x = number of data length bytes

Appendix B

STATUS CODE SUMMARY

Error codes returned by the Corvus disk controller contain the type of error and error severity. Error severity is coded as follows:

Bit 7 set = Fatal error
Bit 6 set = Verify error
Bit 5 set = Recoverable error

Normal mode command status codes

Disk Status Codes

Non-fatal				Fatal (>= 128)						
Recoverable Error		Verify Error				Recoverable Error		Verify Error		
dc	hx	dc	hx	dec	hx	dec	hx	dec	hx	
32	20	64	40	128	80	160	A0	192	C0	Header fault
33	21	65	41	129	81	161	A1	193	C1	Seek timeout
34	22	66	42	130	82	162	A2	194	C2	Seek fault
35	23	67	43	131	83	163	A3	195	C3	Seek error
36	24	66	44	132	84	164	A4	196	C4	Header CRC error
37	25	67	45	133	85	165	A5	197	C5	Rezero fault
38	26	68	46	134	86	166	A6	198	C6	Rezero timeout
39	27	69	47	135	87	167	A7	199	C7	Drive not online
40	28	70	48	136	88	168	A8	200	C8	Write fault
41	29	71	49	137	89	169	A9	201	C9	--
42	2A	72	4A	138	8A	170	AA	202	CA	Read data fault
43	2B	73	4B	139	8B	171	AB	203	CB	Data CRC error
44	2C	74	4C	140	8C	172	AC	204	CC	Sector locate error
45	2D	75	4D	141	8D	173	AD	205	CD	Write protected
46	2E	76	4E	142	8E	174	AE	206	CE	Illegal sector address
47	2F	77	4F	143	8F	175	AF	207	CF	Illegal command op code
48	30	78	50	144	90	176	B0	208	D0	Drive not acknowledged
49	31	79	51	145	91	177	B1	209	D1	Acknowledge stuck active
50	32	80	52	146	92	178	B2	210	D2	Timeout
51	33	81	53	147	93	179	B3	211	D3	Fault
52	34	82	54	148	94	180	B4	212	D4	CRC
53	35	83	55	149	95	181	B5	213	D5	Seek
54	36	84	56	150	96	182	B6	214	D6	Verification
55	37	85	57	151	97	183	B7	215	D7	Drive speed error
56	38	86	58	152	98	184	B8	216	D8	Drive illegal address error
57	39	87	59	153	99	185	B9	217	D9	Drive r/w fault error
58	3A	88	5A	154	9A	186	BA	218	DA	Drive servo error
59	3B	89	5B	155	9B	187	BB	219	DB	Drive guard band
60	3C	90	5C	156	9C	188	BC	220	DC	Drive PLO error
61	3D	91	5D	157	9D	189	BD	221	DD	Drive r/w unsafe

SEMAPHORE STATUS CODES

DECIMAL	HEX	MEANING
0	00	Prior semaphore state = not set.
128	80	Prior semaphore state = set.
253	FD	Semaphore table full.
254	FE	Disk error.

PIPE STATUS CODES

DECIMAL	HEX	MEANING
0	00	Successful pipe request.
8	08	Tried to read an empty pipe.
9	09	Pipe was not open for read or write.
10	0A	Tried to write to a full pipe.
11	0B	Tried to open an open pipe.
12	0C	Pipe does not exist.
13	0D	No room for new pipe.
14	0E	Illegal command.
15	0F	Pipe area not initialized.

PIPE STATE CODES

DECIMAL	HEX	MEANING
1	01	Open for write, file empty.
2	02	Open for read, file empty.
128	80	Full, not open.
129	81	Full, open for write.
130	82	Full, open for read.

MIRROR STATUS CODES

DECIMAL	HEX	MEANING
1	01	File ID mismatch.
2	02	Illegal restore command (usually checksum error).
3	03	Illegal retry command (retry not enabled, or checksum error).
4	04	File size mismatch.
5	05	Illegal opcode.
7	07	Start of image not found (30 sec. timeout).
8	08	Position error.
134	86	Tape dropout during playback operation (5 sec. timeout).