BRK
01/09/16

Corvus Flat Cable Interface Information

Introduction

The first products introduced by Corvus Systems in 1980 involved various Parallel Port interface cards for use with the Corvus 11 & 20 MB disk drives (**Rev A & B drives**) (which internally used an 8 inch Winchester drive made by IMI). Corvus later introduced 6 MB drive in this family followed later by a new series of 6, 11, & 20 MB drives in smaller boxes using 5 inch Winchester drives, called **Rev. H drives**. All later drives used the **Omninet** interface (1 MBit/Sec twisted pair network interface). There was also an Omninet "disk server" product that allowed interfacing the flat cable drives to Omninet.

The Omnidrives all used 5 inch hard disks of different capacities. The largest one ever shipped had a 128 MByte capacity.

Most of the technical information you need to write drivers for the flat cable interface card can be found in the Corvus General Technical Information for Mass Storage (Manual #7100-05945-01) (MSGTI) which can be found in many internet archives (as a pdf file), for instance:
http://www.textfiles.com/bitsavers/pdf/corvus/710005945_General_Technical_Information_Mass_Storage_Oct84.pdf

Electrical Interface Signals

The flat cable interface used a 34 pin flat cable. These cable signals are:

Data Name	Originator	Flat Cable Pin(s)
D0 (data, bit 0)	bi-directional	25
D1	bi-directional	26
D2	bi-directional	23
D3	bi-directional	24
D4	bi-directional	21
D5	bi-directional	22
D6	bi-directional	19
D7	bi-directional	20
DIRC (Bus Direction)	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
Alternate Select	drive	11
Reserved	computer	5
Unused	-------	1, 2, 7, 12-16, 18, 33

This signal definition was sort an accident of the way signals were available on the back plane of the IMI drive. In particular, note that the data lines are NOT interleaved with ground lines so there is more cross coupling of the data lines than one might want. However, the STROBE and status lines do have neighboring ground lines. This issue is of some importance since Corvus had an 8 port "Constellation" Multiplexer product that allowed sharing a

single drive with up to 8 separate computers (using 34 pin cables of up to 50 feet in length) or up to 64 computers by plugging 8 slave multiplexers into the master multiplexer slots.

Normally the drive is waiting for a command, so the DIRC line is high (Corvus drive NOT driving the data bus) and so is the READY line.

Drive Commands and Protocols

A typical drive command sequence involves sending a command byte followed by one or more additional bytes (as appropriate for the command) and then waiting for the Corvus drive to take control of the bus (signaling this by dropping the DIRC line) and read one or more bytes back from the drive to the computer interface. The various commands supported by the drive can be found in pages 2-71 (and 224) of the MSGTI, with a summary table on pages 3-4 and 224.

A byte is read or written to the bus only if the READY line is high, so this acts as a sort of flow control signal. The drive will set this line low while it is processing the byte. The Constellation Multiplexer uses the READY line (holding it low) as the means to make the unselected computers to wait and (with the READY line allowed to go high) of allowing the selected computer to control the drive.

So the process of writing data to the drive, typically has the steps:

1. Wait for READY line to go high
2. Put the data byte (to send) out on the bi-directional data lines
3. Toggle the STROBE line low (typically for about 500 nsec or longer (for the 1-4 MHz clocks used in the Computers of that era) then high, on the rising edge of this, the drive will latch this data and read it. This data must be held on the bus for at least 50 nsec after the rise of the STROBE line (data HOLD requirement). The READY line will drop while it is processing the byte
4. Do steps 1-3 for subsequent bytes to send.

These steps, as signals on the bus, are shown on page 203 of the MSGTI.

After writing the require data (for the command used), you need to read at least one byte back from the drive. Typically, the first byte read back is an error code. If bit 7 of this is set, it was a fatal error. In that case, you abort the operation of reading any remaining (expected) bytes. The steps for reading bytes back (having writing the command bytes first) are:

1. Monitor the DIRC line and wait until it drops.
2. Delay for at least 20 usec
3. Wait for the READY line to go high
4. Read the byte from the data bus
5. Toggle the STROBE line low then high, on the rising edge of this, the drive will assume this byte was accepted and latched by the computer and will begin the process of fetching the next byte (if any) to present on the bus, dropping the READY line until this next byte is ready to be read.
6. If this was the first byte to be read back, it is probably the return error status byte. If bit 7 of this is set, there was a fatal error and the remaining read back operations will be aborted. Otherwise, if more bytes are expected, loop back to step 3 to read the next expected byte.

These steps, as signals on the bus, are shown on page 204 of the MSGTI. The error codes are also documented in the MSGTI on pages 225-226.

On most of the Corvus interface cards, the STROBE signal is generated automatically by the address decode circuits that decode the I/O port address used (along with the appropriate processor status lines that decode read/write operations…) although a few of these cards use memory mapped I/O instead of I/O ports. However,

some do NOT do this automatically and allow generating the STROBE signal by a separate I/O operation (a little slower, but required in these cases by peculiarities of the particular computer). For instance, in the case of the Osborne 1 computer, the only I/O we could access was a PIO chip built into the computer to drive the GPIB port. So we used the GPIB port and one of the GPIB bus control signals was used instead for the STROBE signal (which was on a different I/O port [actually the Osborne used memory mapped ports]).

Another issue with the automatic generation of the STROBE signal is that the signal generating the strobe would normally also turn off the interface card tri-state drivers to the bus (on data writes to the Corvus drive), potentially causing a data hold problem. Thus most of these interface cards have a little RC + diode delay circuit the keeps the bus driver on for a short time after the STROBE goes high (typically done with a diode bridging a 470 ohm resistor followed by a 200 pf capacitor).

Some diagnostic & test programs, perhaps assuming the drive may be in a weird state, will try to get the drive in a "Ready for new command state", by:

1. Sending the drive an illegal command (a single 0xFF) – without monitoring the READY line!!
2. Do a long delay (~ 500 usec)
3. Check if the DIRC line is low, if not go to step 1
4. Wait for READY to go high
5. Read back a single byte from the bus
6. Toggle the STROBE line
7. Check if the returned data byte was 0x8F
8. If it was not 0x8F, go back to step 1

DRIVE READ/WRITE Sector commands

The actual sector size on the Corvus flat cable drives is 512 bytes. However, many computer operating systems use other sector sizes like 128 bytes (CP/M for example) or 256 bytes (like Apple DOS). To avoid complicated buffering schemes in the drivers (perhaps needing more RAM that is available), the drive can do the required buffering for 128 and 256 byte sectors. In these cases, the sector addressing is also in these sector size units (thus avoiding the required arithmetic operations to convert the sector addresses to the native 512 byte units).

The way the particular sector size is selected is by the command byte used for sector R/W commands:

Command	**Sector Size (in bytes)**		
	128	256	512
Read Sector	0x12	0x22	0x32
Write Sector	0x13	0x23	0x33

For historical reasons (first Apple II DOS product), commands 0x02/0x03 also work for Read/Write 256 byte sectors.

The Read and Write sector command work by first sending the command, followed by three bytes that specify the "drive number" and sector address:

Byte1: R/W command byte
Byte2: drive # in bits 0-3 and upper nibble of 20 bit drive address in bits 4-7
Byte3: lower byte (bits 0-7) of sector address
Byte4: middle byte (bits 8-16) of sector address

So for example, to read from drive 1, (256 byte) sector address 0x23456, a 256 byte sector, we would send the bytes:

0x22, 0x21, 0x56, 0x34

The "drive number" referred to is just 0x01 for a single drive and is only used (other than 0x01) when drives are daisy chained or the "virtual drive offset table" has been setup on a drive (see MSGTI pages 194 & 68-69).

For doing the **write sector** command, the 4 bytes would be followed by the bytes for the sector (128 bytes, 256 bytes, or 512 bytes). You would then wait for the bus to turn around (as described above) and read back the error status byte.

For doing a **read sector** command, the 4 bytes are written and then you wait for the bus to turn around and you read back the error status byte. If bit seven of this is not set (no fatal error) you then proceed to read back the sector bytes (128 bytes, 256 bytes, or 512 bytes).

Examples of some assembly language routines that do these operations can be found in the MSGTI and also in "listing" manual for early (1980) CP/M utilities and drivers. This manual had a yellow cover and was called:

Corvus Utilities for CP/M

This is a little harder to find on the internet, here is one place:

http://maben.homeip.net/static/s100/corvus/OS/CPM/Corvus%2011-20mb%20Utilities%20for%20CPM.pdf

Number of Accessible (user area) Blocks on Flat Cable drives

If you are writing your own tools to R/W sectors on a Flat Cable drive, you will ultimately need to know how many sectors are available to you. Knowing the nominal size (6 MB, 11 MB, 20MB) only gives you an "estimate" as the sizes are rounded to the nearest MB. The specific numbers can be found from the Service Manuals for these drives (most are on the WEB), by using specific commands that read drive parameters, or from those shown in the MSGTI on pages 198 & 200.

For example, on page 200 it says that the REV H. 6 MB drive has 11540 blocks (512 byte sectors). Thus, if you are accessing this as 512 byte sectors, it will respond to sector addresses 0-11539. However, if you are using 256 byte sectors, the range is 0-23079 (and 0-46159 for 128 byte sectors).

By the way, the smaller sector sizes are laid out in the 512 byte sector in the "natural" way:

512 byte Sector 10 : 256 byte sector 20 followed by 256 byte sector 21.
: 128 byte sectors 40, 41, 42, 43

Operating System Drivers

In a number of the operating systems that Corvus supported (interfaced to), there wasn't really any provision for "large" logical drives (much bigger than the sizes of the floppy drives they were written for). In these systems, the Corvus drive (or a portion of it) was laid out as a collection of floppy sized logical hard drive volumes with some provision for selection of which floppy sized image to access. In particular, this was true for Apple Dos, where the original advertisements for the 11 MB drive system referred to supporting as many as 80 floppy drive images.

Other more “adaptable” systems like CP/M and MSDOS supported large volumes (of selectable size).

In the first versions of support for these systems, it was assumed that the volumes sizes and locations were mostly “fixed”, either by “wiring” these assumptions into the Corvus drivers (and supplied utilities) or by writing some hidden tables on the drives (accessible by the Corvus OS drivers and utilities with these assumptions built in).

With the introduction of the Corvus Constellation multiplexer, there was pressure to change this in a “uniform” way that supported user selection of the volumes locations, sizes, and access privileges. This was first done on the Apple II, allowing DOS, UCSD Pascal, and CP/M volumes with individual setup of volume “mounting” configuration based on user name (with user passwords supported). This system was called “Constellation Software” with the later versions called Constellation II and Constellation III.

Most of the earlier OSes/systems Corvus initially supported (as single drive/OS) based systems, like the TRS80 model 1, Atari, and CP/M based systems, were NOT supported with this new software system. In the case of CP/M the driver sources were supplied, so some users were able to place their CP/M volumes (by changing tables in the drivers) to agree with locations setup with the Corvus Constellation software). The fact that the constellation utilities even supported CP/M was in part due to support for the Microsoft Z80 “soft card”(for Apple II) with CP/M and support for CP/M volumes on the **Corvus Concept** (but there was a hope we might eventually extend Constellation software support to the Corvus CP/M drivers – requiring in particular some extension of the tables in the drivers and a “logon” program that would configure the tables dynamically, based on the “mount” configuration selected for the user (triggered by entering the username and password into prompts from the ”logon” program). However, by this time most of the early CP/M system manufactures were going out of business and MSDOS based systems had replaced them (and MSDOS was supported by Constellation software tools).