R Kirk

WANG LABORATORIES, INC.

MEMORANDUM

COMPANY CONFIDENTIAL

TO: R. Kolk, D. Angel, B. Patterson, P. Seymour, J. Sevigny
FROM: Max Blomme
DATE: November 10, 1980. Revised May. 14,1981
SUBJECT: 2200 Disk Command Sequences for 2200 LVP

NOTE

Release 1.9 or later of the MVP and Release 1.8 or later of the VP are required for the new $GIO disk sequences; an execution error will occur if the user attempts to execute the new sequences on an older version of the VP or MVP. GIO to the disk is not in general supported. GIO notation is used here for documentation and diagnostic purposes only.

The existing 2200 disk sequences are not adequate for handling more than 4 disk platters/drive or more than 8 Mb/platter, assuming that no changes are to be made to the existing disk controllers. In addition, commands other than read, write and compare may be required for disk formatting, error recovery, and more efficient operations. The following sequences have been modeled after the existing sequences in order to reduce the change required in the 2200VP/MVP BASIC-2 Interpreter and Bootstrap.

It is assumed that device ready will only be ready when the disk microprocessor is willing to receive a strobe (CBS or OBS) or waiting to send a strobe (IBS), and that the data will be taken within the 5 usec. strobe. The 2200 will wait for device ready before issuing any output strobes. Busy must be set during the physical disk operation so that the 2200MVP can overlap processing with the disk operation. Ready/busy may be used to control the data rate to the disk processor during write and compare operations; however, the disk processor must be able to keep up with the existing 2200 sequences that do not check RB (this is approximately 15 usec./character). Particular attention should be paid to optimizing the data transfer loops in the disk microprocessor.

(1)

2200 Disk Command Sequences for 2200 LVP
COMPANY CONFIDENTIAL

2200 MVP Timing Requirements:

1. In order to provide disk overlap, the 2200 MVP breakpoints during disk operations if the disk does not become ready within a prescribed period of time. When a breakpoint occurs, the disk is disabled (however, hog mode has been set). Thereafter, the MVP periodically (between execution time slices) checks for disk ready. When ready, the disk is enabled and the disk sequence is continued. The disk microprogram should be structured to minimize the number of breakpoints that will occur since unnecessary breakpoints cause poor disk performance and poor CPU utilization.

2. Before each output strobe (OBS), the 2200 MVP waits up to 1 ms for disk ready before breakpointing.

3. For input strobe (IBS) device ready indicates that the disk has a byte ready to strobe into the 2200.

   a. ERR I90 results if the start of disk operation (initialization) acknowledge is not received within 8 ms. after the initialization strobe is sent.

   b. The 2200 MVP waits up to .25 ms (1 ms for $GIO sequences) for device ready after strobing the sector address to the disk. If the disk is ready, CPB is set to ready and the 2200 waits for the acknowledgement. If the disk does not become ready, a breakpoint occurs. Thus, the sector address checking by the disk should occur within the .25 ms.

   c. The MVP waits up to .25 ms (1 ms for $GIO sequences) for disk ready indicating that the operation is complete (disk is to send acknowledgement). If ready is not set, the MVP breakpoints.

   d. For all other IBS's, ERR I92 results if the IBS is not received within 8 ms.

A. General Disk Command Format

IOB	A0	IOB's = start of disk command.
WR/OBS		Signal Disk Microprocessor to check IOB's
IBS	cd	acknowledgement D0 if OK 00 if error

Note:
1. An error results if acknowledgement not received within 50 ms.
2. The ACK value D0 indicates that the "Phoenix" disk sequence is to be used to communicate with the disk microprocessor.

IOB	40	IOB's = disk operation in progress.

Note:
1. IOB's are only checked by the disk microprocessor during OBS strobes. At other times the IOB's are undefined.

2200 Disk Command Sequences for 2200 LVP
COMPANY CONFIDENTIAL

WR/OBS/echo cccppppp ccc = command

ppppp = platter: 10 = removable
00 = 1st fixed
01 = 2nd fixed
02 = 3rd fixed
03 = 4th fixed
04 = 5th fixed

ccc = command: 000 = read sector
010 = write sector
100 = compare sector
001 = extended command (next byte specifies command

WR/OBS/echo	ef	extended command byte (not send if read, write, or compare sector). Echo is inverted if the DPU cannot handle the command.

If a sector address is required,

WR/OBS/echo	gh	
WR/OBS/echo	ij	ghijkl = sector address. If 'gh' or 'ij' is
WR/OBS/echo	kl	echoed incorrectly, the sequence is restarted.
WR/IBS	op	acknowledgement of valid sector address 00 if OK 01 if sector not on disk (ERR I98) 02 if disk hardware error (ERR I91) 04 if format key engaged (ERR I94)(not used) If echo of 'kl' was incorrect, sequence is restarted.

The remainder of the sequence is customized for each command, but usually consists of:

WR/OBS	00	start physical disk operation
.		
.		
.		
physical disk operation		
.		
.		
.		
WR/IBS	yz	operating completion code

where:

OBS = OBS strobe from 2200
echo = IBS strobe to 2200 of data just sent to the disk
IBS = IBS strobe to 2200
WR = 2200 wait for disk ready
IOB = set the IOB address latches (no strobe)

B. Command Sequences

1a. Read Sector

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 000ppppp		command/platter
WR/OBS/echo	gh	
WR/OBS/echo	ij	sector address
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
WR/OBS	qr	signal disk to check IOB's to insure not restarting disk sequence.
WR/IBS	st	acknowledgement of read 00 if OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if CRC error (ERR I96)
IBS		
. (256 bytes)		data
.		
.		
IBS		
		Note:
		Error I92 results if IBS is not received within 1 ms. after CPB = ready.
IBS	uv	LRC (binary add without carry of all data bytes)
		ERR I97 if LRC is incorrect.

1b. Read Bad Sector

If a sector is bad (CRC error), the data can be requested by continuing the read sequence after "IBS st" in read sequence above. The IBS following the read acknowledgement signals that the bad data is to be transferred even though a CRC error was detected.

5

2. Write Sector

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 010ppppp		command/platter
WR/OBS/echo	gh	
WR/OBS/echo	ij	sector address
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
WR/OBS		
. (256 bytes)		data
.		
.		
WR/OBS		

Note:

Disk processor should check IOB's to be sure not restarting disk sequence.

WR/OBS	uv	LRC

Note: disk should check LRC (but doesn't).

WR/IBS	st	acknowledgement of write 00 if OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if LCR error (LRC didn't check what the disk calculated, see note)(ERR I96)

3. Compare Sector (used by read-after-write operations)

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 100pppp		command/platter
WR/OBS/echo	gh	
WR/OBS/echo	ij	sector address
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
WR/OBS	qr	signal disk to check IOB's to insure not restarting disk sequence.
WR/IBS	st	acknowledgement of read 00 if OK non zero (08), read after write error (ERR I99)
WR/OBS		
. (256 bytes)		data
.		
.		
WR/OBS		
WR/OBS	uv	LRC
WR/IBS	wx	acknowledgement 00 if OK non zero (08), read after write error (ERR I99)

4. Format Platter (zero data, verify)

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 001pppp		command/platter
WR/OBS/echo	02	command
WR/OBS	00	start formatting
WR/IBS	yz	acknowledgement of format 00 if OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if CRC error (ERR I96)

5. Copy Sectors (data not transferred to 2200)

Enable source disk		
IOB	A0	start of operation
WR/OBS		
IBS	cd	cd = acknowledgement
IOB	40	
WR/OBS/echo 001ppppp		command/platter
WR/OBS/echo	01	command
WR/OBS/echo	gh	
WR/OBS/echo	ij	starting address on source platter
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
WR/OBS/echo	gh	
WR/OBS/echo	ij	ending address on source platter
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
Enable destination disk		
WR/OBS/echo 000ppppp		destination platter
WR/OBS/echo	gh	
WR/OBS/echo	ij	starting address on destination platter
WR/OBS/echo	kl	
		disk should check implied ending address
WR/IBS	op	acknowledgement of valid sector address
WR/OBS	00	start copy
WR/IBS	yz	acknowledgement of format 00 if OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if CRC error (ERR I96)

6. Multi-Sector Write

If several sequential sectors are to be written (e.g., program SAVE), the disk processor multi-sector buffering capability can be taken advantage of by preceding the sector writes with a "start of multi-sector write" command and termnating the multi-sector write with an "end of multi-sector write" command. The standard write sequence is used for each sector to be written. However, the sectors are not written until the disk processor's buffers are full or the "end of multi-sector write" command is received, at which point the sectors are written in a single rotation of the disk. Write errors are reported as soon as the processor has noticed an error and is in a position to give an error or when the "end of multi-sector write" command is received; otherwise the disk acknowledges a "good" write even though the write has not yet occurred. The following sequence is for multi-sector write:

a. Start of Multi-Sector Write

IOB WR/OBS	A0	start of operation
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 001ppppp		command/platter
WR/OBS/echo	10	command
WR/OBS	00	start of multi-sector write

b. Write

The standard write sequence is used for each sector. However, write errors may not be reported until the "end of multi-sector write" command is received.

c. End of Multi-Sector Write

IOB WR/OBS	A0	start of operation
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 001ppppp		command/platter
WR/OBS/echo	11	command
WR/OBS	00	request of write acknowledgement
WR/IBS	st	acknowledgement of writes 00 if all writes OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if CRC error (ERR I96)

7. Verify Sectors (data not transferred to 2200)

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo	001ppppp	command/platter
WR/OBS/echo	12	command
WR/OBS/echo	gh	
WR/OBS/echo	ij	starting sector address
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
WR/OBS/echo	gh	
WR/OBS/echo	ij	ending sector address
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement
WR/OBS	00	start verifying
WR/IBS	qr	
IBS	st	sector address of last sector
IBS	uv	verified
WR/IBS	wx	acknowledgement of verify 00 if OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if CRC error (ERR I96) 09 if beyond limits error (ERR I98)

8. Format track

IOB WR/OBS	A0	start of operation
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo	001ppppp	command/platter
WR/OBS/echo	18	command
WR/OBS/echo	gh	
WR/OBS/echo	ij	sector address (track which contains this sector is formated)
WR/OBS/echo	kl	
WR/IBS	op	acknowledgement of valid sector address
WR/OBS	00	start format
WR/IBS	yz	acknowledgement of format 00 if OK 01 if seek error (ERR I95) 02 if format error (ERR I93) 04 if CRC error (ERR I96)

9. Read Status

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 001ppppp		command/platter
WR/OBS/echo	16	command
WR/OBS	00	start reading status
WR/IBS	uv	count of bytes in message (not including this one)
WR/IBS	ef	
WR/IBS	gh	'efgh' = DPU type (2 ASCII digits)
WR/IBS	ij	protocol level (1 ASCII digit)
WR/IBS	kl	'klmn' = microprogram release (2 ASCII
WR/IBS	mn	digits)
WR/IBS	op	
WR/IBS	qr	'opqrst' binary number of sectors
WR/IBS	st	for the platter addressed
WR/IBS	ab	future expansion
WR/IBS	ab	future expansion
WR/IBS	ab	future expansion
WR/IBS	ab	future expansion
WR/IBS	ab	future expansion
WR/IBS	ab	future expansion
WR/IBS	ab	future expansion

10. Turn off retry and ignore address check

IOB	A0	start of operation
WR/OBS		
IBS	cd	acknowledgement
IOB	40	
WR/OBS/echo 001ppppp		command/platter
WR/OBS/echo	17	command
WR/OBS	00	do the operation

WARNING: Since no address checking is done, it is possible to cause the drives to seek beyond normal ranges, thereby possibly damaging them. Reset restores the DPU back to normal retries and address checking.

C. SUGGESTED $GIO SEQUENCES FO.. DISK COMMANDS

The following list of $GIO sequences for Phoenix type disk commands uses the following parameters.

```
                HEX(10) if removable platter
platter =       HEX(00) if fixed platter

error return = HEX(abcdef)

where:

abcdef = 000000 if no errors
abcdef = 000004 if echo error, should retry command.

     ab = 01 if sector not on disk (ERR I98)
          02 if disk hardware error (ERR I91)
          04 if format key engaged (ERR I94)

     cd = 01 if seek error (ERR I95)
          02 if format error (ERR I93)
          04 if CRC error (ERR I96)
          08 if read after write error (ERR I99)
          09 if beyond end of disk (ERR I98)

/xyz = disk address

variables used:  DIM G$15, G$(4) 64, E$(5) 2
```

1. Format Platter

```
Entry:  STR(G$,1,1) = platter logical-or hex(20)

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6802 4000 8B67, G$)

Return:  STR(G$,6,3) = error return
```

2. Read Sector

```
Entry:  STR(G$,1,1) = platter
        STR(G$,2,2) = sector address

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6800 6A20 6230 8705 1704
1156 1576 4000 8367 C640 860B, G$) G$()

Return:   STR(G$,6,3) = error return
          STR(G$,11,1) = LRC (binary add without carry of all data
          bytes)
          G$() = data
```

Read bad sector

Entry: STR(G$,2,2) = sector address
DATALOAD BA T/xyz, (STR(G$,2,2)) G$():ERROR X=ERR:IF X = 96 THEN $GIO/xyz(C640 860B,G$)G$()

Return: STR(G$,6,3) = error return
STR(G$,11,1) = LRC (binary add without carry of all data bytes)
G$() = data

3. Write Sector

Entry: STR(G$,1,1) = platter logical-or hex(40)
STR(G$,2,2) = sector address
STR(G$,4,1) = LRC (binary add without carry of all data bytes)
G$() = data

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6800 6A20 6230 8705 1704 1156 1576 1300 A000 4240 8B67, G$) G$()

Return: STR(G$,6,3) = error return

4. Compare Sector

Entry: STR(G$,1,1) = platter logical-or hex(80)
STR(G$,2,2) = sector address
STR(G$,4,1) = LRC (binary add without carry of all data bytes)
G$() = data

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6800 6A20 6230 8705 1704 1156 1576 4000 8B67 A000 4240 8B67, G$) G$()

Return: STR(G$,6,3) = error return

5. Copy Sectors

Entry: STR(G$,1,1) = source platter logical-or hex(20)
STR(G$,2,2) = source starting sector address
STR(G$,4,2) = source ending sector address
STR(G$,11,1) = source disk address
STR(G$,12,1) = destination platter
STR(G$,13,2) = destination starting address
STR(G$,15,1) = destination disk address, allowed to differ (xor) from source disk address by hex(40) only

$GIO/xyz (0600 0700 73B0 70A0 4002 88D0 7040 6A10 6801 6800 6A20 6230 8701 1704 1116 1576 6800 6A40 6250 8701 1704 1116 1576 73F0 7040 6AC0 6800 6AD0 62E0 8701 1704 1116 1576 4000 8B67, G$)

Return: STR(G$,6,3) = error return

6. Verify Sectors

Entry: STR(G$,1,1) = platter logical-or hex(20)
STR(G$,2,2) = starting sector address
STR(G$,4,2) = ending sector address

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6812 6800 6A20 6230 8701 1704 1116 1576 6800 6A40 6250 8705 1704 1156 1576 4000 870B 860C 860D 8A67, G$)

Return: STR(G$,6,3) = error return
STR(G$,11,3) = sector address of last sector verified

7. Start of Multi-Sector Write

Entry: STR(G$,1,1) = platter logical-or hex(20)

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6810 4000, G$)

Return: STR(G$,6,3) = error return

8. End of Multi-Sector Write

Entry: STR(G$,1,1) = platter logical-or hex(20)

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6811 4000 8B67, G$)

Return: STR(G$,6,3) = error return

9. Check Disk Type

$GIO/xyz (0600 0700 70A0 4002 870B, G$)

Return: STR(G$,6,3) = error return
STR(G$,11,1) = disk initialization response
D0 if Phoenix type (2280, LVP/SVP DPU)
C0-CF if earlier type (2230, 2240, 2260, or 2270 series)

10. Format track

Entry: STR(G$,1,1) = platter logical-or hex(20)
STR(G$,2,2) = sector address (track that contains this sector is formated)

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6818 6800 6A20 6230 8705 1704 1156 1576 4000 8707, G$)

Return: STR(G$,6,3) = error return

11. Read Status and Error Counts
Entry: STR(G$,1,1) = platter logical-or hex(20)

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6816 4000 8705 1A00 C340, G$) G$; STR(E$(),1,VAL(STR(G$,5,1)))

Return: STR(G$,6,3) = error return
STR(E$(),1,2) = DPU type
STR(E$(),3,1) = protocol level
STR(E$(),4,2) = release of microprogram
STR(E$(),6,3) = sectors on this platter

12. Ignore address check and do zero retries

Entry: STR(G$,1,1) = platter logical-or hex(20)

$GIO/xyz (0600 0700 70A0 4002 88D0 7040 6A10 6817 4000, G$)

Return: STR(G$,6,3) = error return

Note: This condition holds untill a reset is sent to the disk : $GIO/xyz (4501)