XEBEC PRODUCT SPECIFICATION
HOST ADAPTER PERSONALITY CARD
FOR THE IEEE 696.1 (S-100) BUS

XEBEC PART NO. 104612-00

8/8/82

TABLE OF CONTENTS

1.0 INTRODUCTION

The Xebec Host Adapter Personality Card is a single interface card for the IEEE 696.1 (S-100) Bus. This Host Adapter may be utilized with any Xebec disk controller (or Disk subsystems which uses a SASI host interface such as the S1410). This specification provides the programming mechanism utilized by the Host Adapter. The detailed specifications for the disk controllers can be found in the respective controller documentation.

The Host Adapter fits into a single S-100 Bus slot.

Commands are issued to the controller through the Host Adapter in the host computer. The Host Adapter transfers data between the host memory and controller.

2.0 THEORY OF OPERATION

Disk commands are issued to the controller via commands stored in the host memory (the command structure is described in each of the controller specifications). Depending on the type of command, the controller will request up to 10 command bytes. Upon receiving the last command byte, command execution is begun.

For data transfer commands, a check is performed on the disk address, and status is flagged if it exceeds the drive limits. Data is stored in a sector buffer on the controller before it is transferred to the host or disk drive, eliminating any possibility of data overruns or underruns between the host and the disk.

Upon command completion, the controller outputs the completion status to the data register in the Host Adapter. (Further completion status detail can be requested by issuing the appropriate sense commands.)

2.1 Electrical Interface

The electrical interface to the disk controller conforms to the requirements described in the disk controller interface specifications.

The electrical interface to the S-100 Bus conforms to the requirements of the S-100 backplane.

2.2 Specifications

HOST ADAPTER PHYSICAL PARAMETERS

(The Host Adapter fits into a single S-100 slot).

Width	10.0 inches
Length	5.125 inches
Height	.075 inch
Weight	0.7 lbs.

ENVIRONMENTAL PARAMETERS

	Operating:	Storage:
Temperature (degrees F/C)	32/0 to 131/55	-40/-10 to 167/75
Relative Humidity (@ 40 degrees F, wet bulb temp, no condensation)	10% to 95%	10% to 95%
Altitude	sea level to 10K feet	sea level to 35K feet

POWER REQUIREMENTS

Voltage @ current (host adapter) +8 VDC @ 1.5A (max)

3.0 HOST ADAPTER DEFINITION

The Host Adapter is designed to operate in IEEE 696.1 (S-100) systems.

3.1 Interface Register Definition

The interface registers for the Host Adapter are listed below. Where b represents the 6 most significant bits of the I/O address.

Host Address	Register
b00	Data in/out Register
b01	Control Register (write only)
b02	Status Register (read only)
b02	Host Adaptor and Disk Controller Clear (write only)

The bit definition for each register is described in the following sections.

3.1.1 Data In/Out Registers (b00)

This register interfaces to the Disk Controller SASI DATA bus.

3.1.2 Control Register (b01)

<u>S100 BUS</u>

Bit 7	Not used
Bit 6	Assert select and Data bit 0 of the controller SASI BUS (SEL-; DATA0-)
Bit 5	Not used
Bit 4	Not used
Bit 3	Not used
Bit 2	Not used
Bit 1	Enable data, after the selection process
Bit 0	Not used

3.1.3 BUS STATUS - (B02) processor can read status of Disk Controller host bus

BUS STATUS (BSTAT)

Bit 7	REQ - indicates the controller either requests data or has data for the host adapter.
Bit 6	IN/OUT* (reference to controller) - low indicates data to host adapter, high indicates data to controller.
Bit 5	MSG - indicates last byte in data or command string.
Bit 4	COM/DTA* - a command to the controller will be high, data will be low.
Bit 3	BUSY - indicates the status of the busy signal. High means controller is busy.
Bit 2	Not used (low)
Bit 1	Not used (high)
Bit 0	Not used (high)

3.1.4 RESET - (B02) a write to this port resets the Disk Controller and control and Data Register on the host adaptor.

3.2 Software Theory of Operation (Reference Sample Program in Appendix A)

The method by which a command is executed is as follows:

1. Device driver builds a Device Control Block (DCB) in system memory (see the appropriate Disk controller specification).

2. The driver then writes the address of the first byte of the DCB into the Command I/O Pointer Block (CIOPB) of the command driver routine.

3. The DATA ADDRESS (DAD) is also set up if a data transfer is required.

4. The driver now calls the GETCON routine to determine if the controller is busy. When it is not busy, the GETCON routine will assert the SELECT line until the controller responds with a BUSY.

5. When the controller responds to the host adapter by asserting BUSY, the driver calls the OUTCON routine. In response to the REQuest bit in the BSTAT, the driver passes the command a byte at a time to the controller.

6. The controller verifies that the command is correct and begins the command execution phase.

7. After the command has completed, the controller enters the command completion phase. The controller sends a one-byte status to the host adapter indicating whether or not an error occurred during command execution. This is handled by the CMPSTAT routine. Finally, the controller sends the message byte (of zeroes), and the operation is complete.

8. The controller then enters the idle (non BUSY) phase awaiting another command. If an error was encountered by the controller, the CMSTAT routine will return with it in the C register. It is the responsibility of the device driver to issue a REQUEST SENSE command to request any detailed information about the error.

3.3 Hardware Theory of Operation

The Host Adapter serves as a data channel for the controller. Commands and data are fetched/stored to the system memory as a function of REQ. The Host Adapter consists of Command and Status Registers, an optional DMA channel, and an interrupt latch. The registers are addressed as I/O ports. Commands and data are passed through these registers as a function of the I/O driver routine and the controller status lines. The host adapter will return an ACK after each DATA or COMMAND cycle has been completed.

Each memory cycle is initiated when the controller asserts REQ. The driver will respond by reading/writing the data register.

When data is transferred to the host adapter, the data on the host bus is held until the memory write is completed. When data is transferred to the controller, the data is latched into a holding register, then sent to the controller.

3.3.1 I/O Logic Operation (Bus Slave)

The Host Adapter responds to commands from the CPU processor to either read a particular register or write to a register. The address selection switches are set with the dipswitch at location 5D. The dipswitch selects a block of four I/O addresses. A read is selected when lines DBIN, pR/W*, and sINP are asserted with the appropriate I/O address. A write is performed when sOUT is high and pR/W* is low along with the I/O address. Because low power Schottky logic is used, the I/O logic will perform at the highest speed clocks now currently in use.

4.0 INSTALLATION

4.1 Inspection

Inspect all shipping containers for damage. If a container is damaged, the contents should be checked and the Host Adapter inspected for physical damage. If package damage is found, the carrier should be notified immediately.

4.2 Preparation for Use

Before the Host Adapter can be used, initial setup may be required. Be sure the power requirements for the Host Adapter are met (section 2.2).

A 50-pin, mass-terminated cable connectes the Host Adapter to the controller board (pin 1 is marked on the Host Adapter and on the controller silkscreen). Refer to the interconnection diagram in the appropriate controller specification for connection of the controller to the disk drives. Note that all cables, including drive cables, are of the mass-terminated type, so no inadvertant signal swapping can occur.

Be sure the controller has adequate DC power (refer to the controller specification). To set up the controller, refer to the switch setting instructions found in the controller specification.

The following sections describe proper settings and checkout of the Host Adapter.

4.2.1 Address Switches

The address switch is located in position 5D.

Note: If the switch is on, the logic compares for zero (0V to 0.8V) on the S-100 bus.
Bit assignment is as follows:

POSITION	ADDRESS
1	A7
2	A6
3	A5
4	A4
5	A3
6	A2
7	NC
8	NC

4.3 Initial Checkout

The initial verification of the disk subsystem can be done via an appropriate monitor PROM or through a debugging utility such as DDT under CP/M*.

First, verify that all the interface registers are accessible through the correct addresses and that the registers can be read/written with the expected results. Manually input driver routines appendix A or from the XEBEC Driver Diskette. Next, attempt to issue a few commands to the disk subsystem.

*CP/M is a registered trademark of Digital Research

A recommended approach is to first issue a RECALIBRATE command. Then, issue a FORMAT DRIVE command; the recommended interleave for the S-100 system running at 2MHz is 4 (see Disk Controller Manual). After verifying that it executed correctly, issue a SEEK command to verify that the Logical Address calculation has been performed correctly. Finally, data transfer commands should be issued to verify the data path.

5.0 INSTALLATION INSTRUCTIONS FOR *CP/M-80 HARD DISK DRIVER

The Xebec hard disk driver software package consists of four files supplied on the distribution diskette. These are "HDINSTL.DOC", "HDFMT.ASM", "HDLINK.ASM" and "HDDVR.ASM".

HDINSTL.DOC (this file) is all the documentation necessary for the user to install the driver on his S100 Bus system.

HDFMT.ASM is a utility program which enables the user to format the hard disk and initialize the directory area.

The files HDLINK.ASM and HDDVR.ASM are the files which enables the user to generate the hard disk driver program.

The driver operates by linking itself to the existing floppy disk bios. This is accomplished by finding the floppy bios jump table and patching the entry points for all disk operations. The user should only run "HDBOOT" once per cold boot of the system, due to the nature of the patching process.

The advantage of this driver over the conventional type of driver is that the user is not required to create a custom bios to get his hard disk subsystem running. By simply editing a few variables into the source files and assembling them, the driver is ready to operate on the user's specific system.

Following are the steps necessary to create a driver tailored to the user's system.

1. A 1K byte block of memory must be created above CP/M for the driver program. This is normally done by a CP/M SYSGEN.

2. Edit the desired starting address of the driver into the file "HDLINK.ASM". This will be the beginning address of the 1K block created in step 1. The variable name is "START".

3. Edit the following variables into the file "HDDVR.ASM":
   A. Desired starting address of the driver (this must be the same as the address in "HDLINK.ASM"). The variable name is "START".

   B. The host adaptor base address. The variable name in the file is "BASE". The factory standard is 80H...if a different base address is desired the address switches on the host adaptor and this equate will need to be changed.

   C. If your system has a Z80 CPU, setting the equate Z80 to true and the equate C8080 to false will allow the driver to use the Z80 block move instructions.

4. Edit the host adaptor base address into the file "HDFMT.ASM". Again, the factory standard is 80H. The variable name in the file is "BASE".

5. Assemble HDFMT, HDLINK and HDDVR using the CP/M "ASM" program.

6. Create HDFMT.COM using the CP/M "LOAD" program.

7. To create the "HDBOOT.COM", do the following:

   A. Insure that DDT.COM is on the same diskette as the files "HDLINK.HEX" and "HDDVR.HEX".

   B. TYPE:

      ```
      DDT HDLINK.HEX
      ```

   C. When DDT returns with the - prompt, type:

      ```
      IHDDVR.HEX
      RXXXX
      ```

      Where XXXX is determined by the formula:
      XXXX=FC00H - desired starting address (H) + 800H
      Example:

      Starting Address F800
      XXXX=FC00H - F800H + 800H
      XXXX=0C00H

   D. Type CTRL-C to return to the CP/M console command processor.

   E. Type:

      ```
      Save 8 HDBOOT.COM
      ```

      You now have the driver program under the name "HDBOOT.COM"

8. Format the hard disk using the utility program "HDFMT".

9. When the hard disk has been successfully formatted, you are ready to install the driver.

10. Type:

    ```
    HDBOOT
    ```

    This will install the driver for this session. Each time you COLD BOOT the system, you will need to run the "HDBOOT" program. Remember, only do this once per COLD BOOT.

*CP/M is a registered trademark of Digital Research

6.0 REFERENCE DOCUMENTATION

6.1 IEEE S-100

a. IEEE 696.1 Standard Specifications for S-100 Bus Interface Devices.

b. S-100 CPU/System Manual - use version appropriate for your system.

6.2 Disk Drive Documentation

Use the appropriate drive manufacturer's manual for your disk drive.

6.3 XEBEC Disk Controller or Disk Subsystem User Manual

APPENDIX A
SAMPLE ROUTINES FOR XEBEC S-100 HOST ADAPTER PERSONALITY CARD

This Host Adapter programming example uses programmed I/O, taking advantage of the fact that the Xebec Disk controllers have a built-in sector buffer. The control lines of the host bus are available to the CPU through the Bus Status Register. Data and commands are transmitted through the host bus by a simple handshake procedure as outlined in the controller specifications.

PROGRAMMING: (8080 CODE)

```
BASE equals Base I/O Address
DATAN equals BASE
DATAOUT equals BASE
BCON equals BASE+1 ;Buss Control
BSTAT equals BASE+2 ;Bus Status
DMAOUT equals Base+3 ;DMA control bytes
DMAIN equals BASE+3 ;DMA status information
CIOPB ;Command Address
BUFFER ;Data Address
```

Sample Routine to GET CONTROLLER;

```
GETCON:  IN BSTAT            ;input from status port
         ANI 08H             ;select bit 3 (busy)
         JNZ GETCON          ;if busy wait in getcon loop
         MVI A,40H           ;get ready to assert SEL and DATA0
         OUT BCON            ;to get attention of controller
CBUSY:   IN BSTAT            ;input from bus status
         ANI 08H             ;again look at BUSY
         JZ CBUSY            ;we have controller attention else loop
         MVI A,02H           ;get ready to allow data enable
         OUT BCON            ;done
         RET                 ;return from get controller routine
```

Sample Routine to OUTPUT COMMANDS:

```
OUTCOM: LHLD CIOPB          ;load pointer to command queue
COMREQ: IN BSTAT            ;input from bus status
        MOV C,A             ;store in C
        ORA A               ;set flags
        JP COMREQ           ;wait for REQ
        ANI 10H             ;check for command/data
        RZ                  ;return when data is requested
        MOV A,C             ;also see if controller switched direction
        ANI 40H
        RZ                  ;if it wants to send data, return
        MOV A,M             ;move commands from queue to accumulator
        OUT DATAOUT         ;write commands to controller
        INX H               ;increment pointer
        JMP COMREQ          ;loop as long as commands are requested
                            from controller
```

Sample Routine to WRITE DATA TO CONTROLLER:

```
WRITE:   LHLD BUFFER        ;load pointer to data
DAREQ:   IN BSTAT           ;input from bus status
         MOV C,A            ;store
         ANI 80H            ;set flags
         JZ WTREQ           ;wait for REQ
         ANI 10H            ;check for COM
         JNZ CMPSTAT        ;on receipt of command completion status is
                            present
         MOV A,M            ;move data into accumulator
         OUT DATAOUT        ;output to controller
         INX H              ;increment pointer
         JMP WTREQ          ;go back for another byte
CMPSTAT: IN DATAIN          ;input completion status
         MOV C,A            ;place in C for further use
```

```
LREQ:   IN BSTAT           ;looking for last REQ
        MOV B,A            ;save for checking
        ANI 80H            ;check for REQ

        JZ LREQ            ;loop until found
        IN DATAIN          ;input last byte
        ORA A              ;see if last byte is non-zero
        JNZ BADBYTE        ;if last byte is non zero
        MOV A,C            ;now check completion status
        ORA A              ;to see if it is zero
        JNZ BADSTAT        ;if not zero
        XRA A              ;zero accumulator
        RET                Everything is OK
```

For information on how to decode errors generated, refer to the appropriate XEBEC controller specification.

Sample program to READ DATA FROM CONTROLLER:

```
READ:   LHLD BUFFER        ;load data pointer
RDREQ:  IN BSTAT           ;input bus status
        MOV C,A            ;store for further checking
        ANI 80H            ;look for REQ
        JZ RDREQ           ;else loop
        MOV A,C
        ANI 10H            ;check for COM
        JNZ CMPSTAT        ;if COM present must be completion status
        IN DATAIN          ;input data from controller
        MOV M,A            ;move data to pointer
        INX H              ;increment pointer
        JMP RDREQ
```