SCSI

***S**mall **C**omputer **S**ystems **I**nterface*

Host Adapter Card

WE HARNESS TECHNOLOGY

WESTERN HORIZON TECHNOLOGIES

SOFTWARE INTERFACE SPECIFICATION

This page left intentionally blank

Table Of Contents

This page left intentionally blank

Introduction

The WHT SCSI host adapter peripheral is a powerful interface allowing the connection and communication with nearly any type of SCSI peripheral including hard drives, floppy controllers, CD ROM drives and tape drives. The purpose of this manual is to make you familiar with how the TI 99/4a Home Computer and the Myarc Geneve 9640 utilize the SCSI host adapter and attached mass storage devices to save and load files, file management, and programming. This manual does not instruct on how to configure or set up the hardware, please read the ***Hardware Configuration*** manual for that information.

Supported SCSI peripherals

The WHT SCSI host adapter peripheral supports many different SCSI peripherals. The following peripherals are supported automatically without any additional software required:

- Hard drives up to 1 gigabyte capacity supporting 512 bytes per sector, SCSI 1 or SCSI 2 compliant. This includes any MAC, PC, or AMIGA compatible hard drives.
- The TEAC FC-1 Floppy controller daughter card with **TEAC** compatible floppy drives attached.
- NEC CD-ROM drives (Audio functions only).
- ANY CD-ROM drive supporting SCSI 2 protocol (Audio functions only).

Peripherals that are compatible but require device specific drivers written to operate:

- Hard drives larger than 1 gigabyte
- Scanners
- Tape and DAT drives
- Printers
- Ethernet Adapters

This manual contains information sufficient to communicate to these kinds of devices via the SCSI bus but does NOT contain information specific to these devices necessary for full operation. Refer to the manufacturers software interfacing specifications on accessing these devices.

SCSI Basics

What is SCSI? Simply put, SCSI is a specification that describes two things, a physical hardware interface and a software communication protocol. It is very similar to RS232, but is a parallel bus rather than serial, and has a complete set of instructions to perform certain functions like reading and writing sectors to a hard drive. SCSI is NOT just another way to hook up a hard drive.

SCSI can be asynchronous or synchronous, eight, sixteen or thirty-two bits wide, and run at various speeds. Our SCSI bus is an eight bit bus, running synchronously or asynchronously (most operations are async) at a maximum of 3 megahertz. In actuality, we can expect peak transfer rates of about 150K Bytes per second on the 99/4a, and 360K Bytes per seconds on the Myarc Geneve. However, because most of the files we transfer are very small, the transfer speed will be much lower than that. Special programs can be written to move large amounts of data over the SCSI bus and achieve those transfer rates.

Devices that attach to SCSI must be “intelligent”. They must contain their own microprocessor for communicating over the SCSI bus and for sending requested data over the bus. Each device is “spoken to” independently and must know how to respond and transfer data across the SCSI bus.

Every SCSI device has an address where it can be communicated to over the bus. There are 8 addresses, and the host adapter card resides at one of these addresses. In our systems, the first address (address 0) must always be a hard drive in order to store the ALIAS file described later in this document. Every other device in the chain may be at any other address providing it is not at an address taken by another device in the chain. You may have more than one host adapter on the bus, and therefore share hard drives between systems. Care must be taken to make sure both systems do not write to the same disk at the same time or you may lose data. This is not a defect in SCSI by any means, but a lack of networking and file sharing protocol in the TI operating system.

Configuring for compatibility

ALIASing drive names

One of the CALL's provided for use with SCSI allows you the user to configure your SCSI devices to be most compatible with existing peripherals in your system. But before you can proceed you need to know a few things about your system without the SCSI card. Take a few moments and scan through your documentation for your other peripherals and make a list of device names already in use. Here are some examples of peripherals and their standard device names:

Peripheral	Device names						
TI Floppy controller	DSK1.	DSK2.	DSK3.				
Myarc & Cor Comp Floppy Controllers	DSK1.	DSK2.	DSK3.	DSK4.			
Myarc HFDC	DSK1.	DSK2.	DSK3.	DSK4.	WDS1.	WDS2.	WDS3.
Myarc HFDC with another FDC	DSK5.	DSK6.	DSK7.	DSK8.	WDS1.	WDS2.	WDS3.
Horizon RAM disk	DSK1. through DSKZ. (This depends on your configuration)						
Other RAM disks	DSK5. through DSK9.						

Write down ALL possible device names in use (even those that don't have drives attached!) in order to avoid all possible conflicts. If you ALIAS a SCSI device to an existing one, you may encounter lock ups, data loss, or other problems that we will not be responsible for.

Once you have identified what drive names exist, decide what you want the SCSI devices to respond as. As a default, all SCSI devices can be accessed as devices SCSX.YY where X is the device address in the SCSI chain plus 1, and YY is a two digit number indicating the logical unit number. Hard drives usually are at a logical unit number of 0, and thus the logical unit number can be omitted. For example, the first hard drive can always be accessed as SCS1. The TEAC FC-1 on the other hand provides access to the floppies attached as logical units 1, 2 and 3 and thus the first floppy drive on the FC-1 located at SCSI bus address 3 can be accessed via SCS4.01. this concept is essential for accessing and ALIASing drive names. Each SCSI device must be ALIAS'ed to another name besides the default name for complete access of this device through assembly, GPL and other high level access.

Here is an example setup... You own 1 hard drive, an FC-1 and a CD-ROM drive... As a default, you MUST set the hard drive as device 0 on the SCSI chain. This is necessary for the ALIAS file to be stored. Without an ALIAS file, the SCSI devices can only be accessed via their default device names. Set the address of the FC-1 to 3 just to move it out of the way of any other hard drives you may choose to install at a later date. Next set your CD-ROM drive to address 4. Here is a table of how you can access the drives in this example case:

Device	SCSI Address	Logical Unit #	Default device name
Hard drive	0	0	SCS1.
FC-1 floppy #1	3	1	SCS4.01
FC-1 floppy #2	3	2	SCS4.02
FC-1 Floppy #3	3	3	SCS4.03
CD-ROM	4	0	SCS5.

Now that we know the device names that may already be used, and we know what the default device names are for our SCSI devices we can now ALIAS the device names to a more common names and avoid conflicts. Lets say you will be using the SCSI card and optional FC-1 floppy daughter card as your only disk controller card. In this case we can ALIAS our devices for WDS1. and DSK1., etc... Here is what we would do. From BASIC, we type in the CALL ALIAS command with parameters to assign a new name to our SCSI device. Here is what we would type for our test case:

```
CALL ALIAS("SCS1.","WDS1.")
CALL ALIAS("SCS4.01.","DSK1.")
CALL ALIAS("SCS4.02.","DSK2.")
CALL ALIAS("SCS4.03.","DSK3.")
```

CALL ALIAS(“SCS5.”,”CDR1.”)

Now our SCSI devices will respond as WDS1, DSK1, DSK2, DSK3, and CDR1. We could ALIAS our SCSI devices to any three letter combination if we choose, you are not limited to WDS, DSK or CDR. The only restrictions are that the device name must be three letters and one number/letter. Valid number/letters are 0 through 9 and A through F. So we support DSKA, DSKB, etc... as well as DSK1...

Once you have entered each ALIAS command, the SCSI card will update the ALIAS file on the first hard drive (SCSI address 0) and the device will respond as it’s ALIAS or it’s default name as long as this file exists. If you want to change all the ALIASes, it can be as simple as re-aliasing an ALIAS to another name, or deleting the file ALIAS/SYS off the root volume of the first hard drive and starting over.

Aliasing does not stop here. We can alias one drive to respond as multiple drives by repeating the process and changing the device name. We can also alias a sub-directory, or entire path as a device name by the same process. For example we can ALIAS WDS1.PROGRAMS.TELCO.DOWNLOAD. to a device name of WDSA. and be able to access sub directory with just the device name of WDSA. and not the entire path, making it quicker to access for you. There are restrictions however, the path including the ALIAS’s device name as it would be typed in it’s unaliased form (i.e. WDS1.PROGRAMS.TELCO.DOWNLOAD.) and file name should not exceed 40 characters for maintainability.

The ALIAS function is truly a powerful function and is a necessary part of making the SCSI peripheral as compatible and flexible as it is.

Managing files through BASIC

Interfacing with BASIC

Accessing files with the WHT SCSI host adapter is fully compatible with the TI floppy disk controller. Complete instructions for loading and saving programs and data files is included with the TI Users Reference Guide and you should refer to that manual for information. However, there are some commands not included with the TI specification that allow us to completely manage our files with DOS like calls.

BASIC calls for disk management

Included with the SCSI card are built in calls for SCSI disk management. The commands syntax is listed here with definitions of terms following. Words in capitals are keywords and must be typed in as you see them. Words in italics must be replaced with the correct syntax for the particular parameter. Words in brackets [ ] are optional and may or may not be included in the command. With the exception of brackets, all punctuation marks must be included in the command.

General SCSI Commands

CALL RESET
Resets the SCSI BUS in the event of a BUS lock up.

Media formatting & directory management commands

CALL FORMAT(“*device name.*”,”[/*sides*] [/*sectors per track*] [/*tracks per side*] [/V *Volume name*]”)
This formats floppies and hard drives attached to the SCSI bus.
Syntax for the device name is SCSX.YY where X is the SCSI BUS address plus one and the YY is a two digit number defining the logical unit number or a valid ALIAS’d name. Hard drives are usually logical unit 0, and thus the logical unit can be omitted. The TEAC FC-1 accesses each floppy as logical unit 01, 02, and 03 respectively.
Syntax for the format parameters is the same as the MDOS specification of its FORMAT command. When formatting hard drives, only the volume name option will be utilized. When formatting floppies with the TEAC FC-1, the following parameters are valid:

Sides	1 or 2
Sectors per track	9, 16, 32, or 61
Tracks per side	40 or 80

These result in 90k, 180k,320k,640k,1.2MB, and 2.44MB floppies.
Volume names may be any 10 or fewer characters except the space or period characters.

CALL LABEL(“*device name.*”,”*new volume name*”)
This labels a floppy or hard drive after it is formatted. Syntax is the same as the /V option for formatting.

CALL MKDIR(“*fspec.directory name*”)
This creates a sub-directory on a floppy or hard drive. Syntax for the file specification (fspec) is any valid device name (either aliased or default) and path such as: WDS1. or WDS1.PROGRAMS.XB. Syntax for the directory name is the same as for volume names.

CALL RMDIR(“*fspec.directory name*”)
This removes an empty sub-directory from a hard drive or floppy drive.

CALL CUTDIR(“*fspec.directory name*”)
This removes a sub-directory from a hard drive or floppy drive whether or not it is empty.

CALL RNDIR(“fspec.directory name”,”new directory name”)

This renames a sub-directory on a hard drive or floppy drive

CALL DIR("fspec.")
This prints a directory of the file spec to the screen.

CALL PATH("fspec.directory name")
This sets the current path for the device name DEF. Once set, you may refer to this path with the device name DEF instead of an alias or default device name.

File management commands

CALL FILES(*number of files*)
Provided for compatibility, has no effect on operations.

CALL RNFILE("*fspec.file name*","*new file name*")
This renames a file on a floppy or hard drive.

CALL COPY("*source fspec.file name*", "*destination fspec.file name*")
This copies a file from one device or subdirectory to another.

CALL MOVE("*source fspec.file name*", "*destination fspec*.")
This moves a file from one device or subdirectory to another.

CALL ATTRIB("*fspec.file name*", "*attribute*")
This protects or unprotects a file. A +p as the attribute sets protections, a -p removes protection.

Audio CD commands

CALL CDPLAY("*valid track number*")
This plays an audio track on the first NEC or SCSI-2 compatible CD-ROM drive in the SCSI chain.

CALL CDSTOP
This stops the currently playing CD on the first NEC or SCSI-2 compatible CD-ROM drive in the SCSI chain.

CALL CDPAUS
This pauses the currently playing track on the first NEC or SCSI-2 compatible CD-ROM drive in the SCSI chain.

CALL CDRESU
This resumes the currently paused track on the first NEC or SCSI-2 compatible CD-ROM drive in the SCSI chain.

CALL CDSTAT(*return string variable$*)
This returns the status of the AUDIO CD in the string provided. This string varies from manufacturer to manufacturer but includes the current track number, playing status, and time.

Other SCSI commands

CALL EJECT(“*device name*.”)

This ejects the media in the drive specified by the device name. This can be an AUDIO CD, or removable media such as a optical disk or floppy with automatic eject mechanism.

CALL ALIAS(“*fspec*.”,” *alias name*”)

This aliases a device name, device name and path, or another alias to a shorter alias name. Valid file specifications (fspec) are the same as the MKDIR command. Valid alias names are any three character combination and a device number. Device numbers can be 0-9 or A-F. If the device number is omitted, then the device is searched for like DSK. For example:

CALL ALIAS(“SCS1.”,”WDS1.”) Aliases the first SCSI device as WDS1

CALL ALIAS(“SCS.”,”WDS.”) Sets the alias WDS to search ALL SCSI devices and logical unit numbers

CALL ALIAS(“WDS1.”,”DSK”) Sets the alias DSK to search all directories on the root volume of WDS1. This is particularly useful for multiplan where if a directory called TIMP exists on this drive, then it will run from hard disk.

When aliasing devices to “searchable” values (DSK, WDS, etc...), only level 3 file access or higher can be made. Before low level device access can be made, each device must be ALIASed to a device name and number. This name and number will define the name and number that low level DSR links will use to access the device. Only level 3 or higher access can be made to a device without a device name/number alias.

Level 3 access is defined as NON SECTOR relative access such as BASIC and EXTENDED BASIC use. Programs like TELCO, ARCHIVER and may more require Level 2 and 1 access and thus need a device name/number ALIAS to be made before they will operate correctly.

The ALIAS command provides us with the most flexible solution to compatibility with both hardware and software.

Accessing SCSI devices through Assembly

DSR link sub programs

The following is an abbreviated syntax of low level DSR link subprograms for access to SCSI devices through GPL and assembly. Information on how to utilize these subprograms is provided in the TI Editor/Assemble manual.

In general the >1X routines are for floppy access, >2X routines are for hard drive access and >3X routines are for access to AUDIO based commands on CD-ROM's. Error codes are generally returned in CPU address >8350.

Sector READ/WRITE - Subprograms >10 and >20

>834A Returned sector number
>834C Unit # Where bit 0-3 is the drive number
bit 4 and 5 is the absolute sector size
0,0 = 256 bytes per sector
0,1 = 512 bytes per sector
1,0 = 1024 bytes per sector
1,1 = 2048 bytes per sector
bit 6 activates absolute sector size access
0 = use logical sector size of 256 bytes
with absolute size of 512 bytes

1 = use absolute sector size in bits 4 &5
bit 7 buffer location
0 = VDP RAM
1 = CPU RAM
>834D Read/Write Where 0 = write
1 = read
>834E Buffer start address in CPU or VDP RAM. 16 bit integer location. Buffer will match the selected sector size of 256, 512, 1024 or 2048 bytes.
>8350 32 bit address of sector number to read/write

Format media - Subprogram >11 floppies

>834A Returned number of sectors per disk
>834C Unit # Where bit 0-3 is the drive number
bit 4 and 5 is the absolute sector size
0,0 = 256 bytes per sector
0,1 = 512 bytes per sector
1,0 = 1024 bytes per sector
1,1 = 2048 bytes per sector
bit 6 activates absolute sector size access
0 = use logical sector size of 256 bytes
with absolute size of 512 bytes
1 = use absolute sector size in bits 4 &5
bit 7 buffer location
0 = VDP RAM
1 = CPU RAM

>834D	# of tracks	Must be 40 or 80
>834E	Buffer start address in CPU or VDP RAM. 16 bit integer location. Buffer will always be 256 bytes.	
>8350	Density	Density of media where 0,1 = Single density FM, 125 K bits/second 2 = Double density MFM, 250 K bits/second 3 = High density MFM, 500 K bits/second 4 = Extra high density MFM, 1M bits/second
>8351	Number of sides	Must be 1 or 2

Format media - Subprogram >21 hard drives, optical and tapes

>834A	Returned SCSI status	
>834B	Returned SCSI message byte	
>834C	Unit #	Where bit 0-2 is the SCSI address bit 4-6 is the logical unit number
>834D	Defect processing	Where bit 0 activates defect processing bit 1 allocate extra sectors/track for error mapping bit 2 use existing sectors for error mapping bit 7 use CPU defect table for error mapping
>834E	Buffer start address in CPU RAM for error mapping. 16 bit integer location.	
>8350	Returned 32 bit number of 256 byte logical sectors per disk	
>8354	32 bit absolute sector size in multiples of 128 bytes.	

Modify file protection - Subprograms >12 and >22

>834C	Unit #	Where bit 0-3 is the drive number bit 7 buffer location 0 = VDP RAM 1 = CPU RAM
>834D	Protect code	Where if 0, file unprotected. <>0 file protected
>834E	Pointer to file name	16 bit pointer to 10 character buffer in CPU or VDP RAM

File/subdir rename - Subprograms >13 and >23

>834C	Unit #	Where bit 0-3 is the drive number bit 7 buffer location 0 = VDP RAM 1 = CPU RAM
>834E	Pointer to new name	16 bit pointer to 10 character buffer in CPU or VDP RAM
>8350	Pointer to old name	16 bit pointer to 10 character buffer in CPU or VDP RAM

Access direct input file - Subprograms >14 and >24

>834C	Unit #	Where bit 0-3 is the drive number bit 7 buffer location 0 = VDP RAM 1 = CPU RAM
>834D	Access code	Where if 0, transfer file parameters, if not 0, indicates the number of sectors to be read in a file. Starting sector number in additional info.
>834E	Pointer to file name	16 bit pointer to 10 character buffer in CPU or VDP RAM
>8350	Additional info	Offset to information at >83XX

>83XX	Buffer start address
>83XX+2	Number of first sector
>83XX+4	Status flags
>83XX+5	Number of records per sector
>83XX+6	EOF Offset
>83XX+7	Logical record size
>83XX+8	Number of level 3 records allocated
>83XX+10	MSB of first sector
>83XX+11	MSB of level 3 record
>83XX+12	Extended record length

Access direct output file - Subprograms >15 and >25

>834C	Unit #	Where bit 0-3 is the drive number bit 7 buffer location 0 = VDP RAM 1 = CPU RAM
>834D	Access code	Where if 0, create file and copy level 3 parameters , if not 0, indicates the number of sectors to be written in a file. Starting sector number in additional info.
>834E	Pointer to file name	16 bit pointer to 10 character buffer in CPU or VDP RAM
>8350	Additional info	Offset to information at >83XX

>83XX	Buffer start address
>83XX+2	Number of first sector
>83XX+4	Status flags
>83XX+5	Number of rescords per sector
>83XX+6	EOF Offset
>83XX+7	Logical record size
>83XX+8	Number of level 3 records allocated
>83XX+10	MSB of first sector
>83XX+11	MSB of level 3 record
>83XX+12	Extended record length

Buffer allocation - Subprogram >16

>834C	Number of file buffers allocated

Set current pathname - Subprogram >27

>834C	Unit #	Where bit 0-3 is the drive number bit 7 buffer location 0 = VDP RAM 1 = CPU RAM
>834E	Pointer to path name	16 bit pointer to 39 character buffer in CPU or VDP RAM

Create subdirectory - Subprogram >18 and >28

>834C	Unit #	Where bit 0-3 is the drive number bit 7 buffer location 0 = VDP RAM 1 = CPU RAM
>834E	Pointer to directory name	16 bit pointer to 10 character buffer in CPU or VDP RAM

Delete subdirectory - Subprogram >19 and >29

>834C	Unit #	Where	bit 0-3 is the drive number
			bit 7 buffer location
			0 = VDP RAM
			1 = CPU RAM
>834E	Pointer to directory name	16 bit pointer to 10 character buffer in CPU or VDP RAM	

Will not remove a subdirectory that is not empty

Cut subdirectory - Subprogram >1A and >2A

>834C	Unit #	Where	bit 0-3 is the drive number
			bit 7 buffer location
			0 = VDP RAM
			1 = CPU RAM
>834E	Pointer to directory name	16 bit pointer to 10 character buffer in CPU or VDP RAM	

Removes an empty or full subdirectory or tree

SCSI Direct - Subprogram >1B

>834A	Returned SCSI status byte		
>834B	Returned SCSI message byte		
>834C	Unit #	Where	bit 0-2 is the SCSI ID
			bit 3 polled I/O select
			1 = use P-DMA access
			0 = Force polled i/o
			bit 6 data buffer location
			0 = VDP RAM
			1 = CPU RAM
			bit 7 CDB buffer location
			0 = VDP RAM
			1 = CPU RAM
>834D	Read/Write	Where	0 = write
			1 = read
>834E	CDB buffer 16 bit start address		
>8350	CDB buffer 16 bit length		
>8352	Data buffer 16 bit start address		
>8354	Data buffer 16 bit length		

Examine SCSI unit - Subprogram >1C

>834A	Returned SCSI status byte		
>834B	Returned SCSI message byte		
>834C	Unit #	Where	bit 0-2 is the SCSI ID
			bit 7 buffer location
			0 = VDP RAM
			1 = CPU RAM
>834E	Buffer 16 bit start address for 44 byte packet		

Packet structure:

Offset	Description
0	8 bytes of information identical to the response to the SCSI inquiry command
8	8 bytes of vendor ID string
16	16 bytes of product ID string
32	4 bytes of product revision string
36	32 bit total number of sectors on device
40	32 bit device sector size

Play AUDIO CD - Subprogram >60

>834C	SCSI Unit where CD player is attached - same bits as defined as format command
>834D	Track number from 01 to 99, use min/sec/frame if 0
>834A	Returned current playing status, 0 if playing, 1 if error
>834B	Returned track number playing
>834E	Minutes to start play at
>834F	Seconds to start play at
>8350	Frame to start play at

Stop CD play - Subprogram >61

>834C	SCSI Unit where CD player is attached
>834A	Returned current playing status, 0 if playing, 1 if stopped

Pause CD play - Subprogram >62

>834C	SCSI Unit where CD player is attached
>834A	Returned current playing status, 0 if playing, 1 if paused
>834B	Track number paused on
>834E	Minutes paused on
>834F	Seconds paused on
>8350	Frame paused on

Resume CD play - Subprogram >63

>834C	SCSI Unit where CD player is attached
>834A	Returned current playing status, 0 if playing, 1 if error
>834B	Track number resumed on
>834E	Minutes resumed on
>834F	Seconds resumed on
>8350	Frame resumed on

Get CD play status - Subprogram >64

>834C	SCSI Unit where CD player is attached
>834A	Returned current playing status, 0 if playing, 1 if paused, 2 if stopped
>834B	Track number on
>834E	Minutes on
>834F	Seconds on
>8350	Frame on

Eject media - Subprogram >65

>834C	SCSI Unit
>834A	Returned SCSI status & message
	error code >06 if media not ejectable
	error code >04 media not able to be ejected at this time

Understanding disk structures

Directory tree structures

In order to make file storage and retrieval more efficient, drives are aid out in a tree like structure containing files and directories. Directories can be thought of the "limbs" of our tree, and the files the "leaves". The very top most directory is referred to as the root directory or root volume. The root volume can contain up to 127 files and 114 subdirectories. Within each subdirectory you can have up to 127 files and 114 subdirectories. The total number of nested subdirectories is limited to the length of the path which is 40 characters. It is possible to have 16 directories nested beneath the root volume assuming each subdirectory name is 1 character long. This is not feasible, and you should restrict your subdirectory names to something recognizable.

Floppy disk formats.

The standard floppy disk format for SCSI Floppies is as follows. Each floppy can contain 3 subdirectories, plus the root directory, and each directory can contain 127 files for a total of 508 files per floppy disk. At the beginning of each volume, there is a Volume Information Block stored in sector 0. This VIB contains the following information about the floppy:

0	Disk volume name	
2		
4		
6		
8		
10	Total number of sectors	
12	Sectors per track	"D"
14	"S"	"K"
16	Protection	Tracks per side
18	Number of sides	Density
20	Directory 1 name	
22		
24		
26		
28		
30	Pointer to dir 1 file descriptor record	
32	Directory 2 name	
34		
36		
38		
40		
42	Pointer to dir 2 file descriptor record	
44	Directory 3 name	
46		
48		
50		
52		
54	Pointer to dir 3 file descriptor record	
56	Allocation bit map	
58		
|		
252		
254		

Disk volume name - Contains the name of the volume in any combination up to 10 ASCII characters except for the period and null character. Un-used characters are padded with spaces.
Total number of sectors - 16 bit integer indicating the total number of sectors on the disk.
Number of sectors per track - 8 bit integer indicating the number of sectors per track.
DSK - Identification characters to indicate formatted floppy media.
Protection - Used for some disk protection schemes.
Number of tracks per side - 8 bit integer indicating the number of tracks per side.
Number of sides - 8 bit integer indicating the number of sides, 1 or 2 sides valid
Density - Indicates the density of the floppy media:

Density	Value	Transfer Rate
Single	0, 1	125 Kbit/sec FM
Double	2	250 Kbit/sec MFM
High	3	500 Kbit/sec MFM
Extra high	4	1 Mbit/sec MFM

Directory names - Indicates the names of the potential directories on this floppy. Format is identical to the volume name
Pointers to directory file descriptor record pointers - Pointers to the location of the FDR for the associated directory. If zero, directory does not exist
Allocation bit map - Used to allocate and reallocate sectors on the floppy. Each bit represents one sector on floppies less than 640K, two sectors on floppies 640K, four sectors on floppies 1.2MB, and eight sectors on floppies 2.44MB.

File descriptor index record

The file descriptor index record contains up to 127 sixteen bit word pointers. Each non-zero pointer indicates the location of a file descriptor record. These pointers are alphabetically sorted according to the filename in the associated file descriptor record. The pointer list starts at the beginning of this block and ends with a zero entry.

The file descriptor index for the main directory is located at sector one of the floppy disk, while the index for the subdirectories are pointed to be their respective pointers.

File descriptor records

The file descriptor record (FDR) contains information about its associated file. All the information the system needs to locate and update the file is contained within the FDR.

0	File name	
2		
4		
6		
8		
10	Extended record length	
12	File status flags	Number of recs/sec
14	Number of sectors currently allocated	
16	End of file offset	Logical record length
18	Number of level 3 records allocated	
20	Date and time of creation	
22		
24	Date and time of last update	
26		
28	Data chain pointer blocks	
30		
|		
252		
254		

File name - Contains the file name of the file. The file name can be any combination of up to 10 ASCII characters excluding the space, period and null character. The name is padded with blanks in the case of less than 10 characters.
Extended record length - Contains the record length if it is a data file and has a record length greater than 255 bytes.
File status flags - These flags indicate the type of file and are as follows:

Bit	Description	Values
0	Program/Data indicator	0= Data file, 1= Program file
1	Binary/ASCII indicator	0= ASCII, 1= Binary
2	Reserved	
3	Protect flag	0= Not protected, 1= Protected
4	Backup flag	0= Not modified since last backup, 1= Modified
5,6	Reserved	
7	Fixed/Variable	0= Fixed length records 1= variable record length

Number of records per sector - Indicates the number of records that will fit into one sector.
Number of sectors currently allocated - Contains the number of sectors allocated for this file.
End of file offset - Contains the offset into the highest sector used in the case of variable and program files
Logical record length - Contains the logical record length of the data file. If this value is 0, length is then given in the extended record length bytes 10 and 11.
Number of level 3 records allocated - In the case of fixed length records, this contains the highest record actually written to. In the case of variable length records, it contains the highest sector actually written to. The bytes in this field are in reverse order.
Date and time of creation/update - Gives the date and time the file was created/updated on this disk.
Data chain pointer blocks - Each data chain pointer block is comprised of three bytes which comprise a cluster. The three bytes are broken down into 2 three nibble blocks. The first indicates the actual sector in the cluster. The second indicates the highest logical record offset in the cluster of contiguous sectors. The following diagram shows how each three byte cluster relates to the address of the first sector and the highest logical sector offset in the cluster.

Byte 1	N2	N1	Byte 2	M1	N3	Byte 3	M3	M2
First sector		00 N3 N2 N1			Highest sector		00 M3 M2 M1	

Hard disk structures

The hard disk format for SCSI hard drives is as follows. At the beginning of each volume, there is a Volume Information Block stored in sector 0. This VIB contains the following information about the disk:

0	Disk volume name	
2		
4		
6		
8		
10	Total number of sectors	
12	Reserved	AU's (W)
14	Reserved (I)	Reserved (N)
16	Reserved/Sectors per AU	
18	Date and time of creation	
20		
22	# of files	# of subdirectories
24	Pointer to file descriptor record pointer	
26	reserved	
28	Pointers to subdirectories	
30		
|		
34		
36		

Disk volume name - Contains the name of the volume in any combination up to 10 ASCII characters except for the period and null character. Un-used characters are padded with spaces.
Total number of allocation units - 16 bit integer indicating the total number of allocation units on the disk.
AU's (W) - The number of AU's allocated for file/directory headers at the beginning of the disk.
Date and time of creation - Date and time the disk was initialized.
Number of files - Indicates the number of files in this directory.
Number of subdirectories - Indicates the number of subdirectories in this directory.
Pointer to file descriptor record pointer - Indicates the AU where the file descriptor index record for this directory is located on the disk.
Pointers to subdirectories - Up to 114 AU pointers to Directory Descriptor Records can be supported by the VIB. The total number is given by the value at byte 23. Pointers are arranged in alphabetical order.

File descriptor index records

The file descriptor index record contains up to 127 one word pointers. Each non-zero pointer points to a file descriptor record. These pointers are alphabetically sorted. The pointer list starts at the beginning of this block, and the 128th entry always points back to its associated Directory Descriptor Record.

The AU number of a File Descriptor Index Record is given in bytes 24 and 25 of its associated Directory Descriptor Record.

Directory Descriptor Records

The format of a Directory Descriptor Record (DDR) is the exact same as that of the VIB with the following two exceptions:

The fields W, I, and N are replaces with ASCII characters “D”, “I” and “R”
Byte 26 contains the pointer to the parent DDR AU.

File descriptor records

The file descriptor record (FDR) contains information about its associated file. All the information the system needs to locate and update the file is contained within the FDR.

<table>
<tr><td>0
2
4
6
8</td><td colspan="2">File name</td></tr>
<tr><td>10</td><td colspan="2">Extended record length</td></tr>
<tr><td>12</td><td>File status flags</td><td>Number of recs/sec</td></tr>
<tr><td>14</td><td colspan="2">Number of sectors currently allocated</td></tr>
<tr><td>16</td><td>End of file offset</td><td>Logical record length</td></tr>
<tr><td>18</td><td colspan="2">Number of level 3 records allocated</td></tr>
<tr><td>20
22</td><td colspan="2">Date and time of creation</td></tr>
<tr><td>24
26</td><td colspan="2">Date and time of last update</td></tr>
<tr><td>28</td><td>“F”</td><td>“I”</td></tr>
<tr><td>30</td><td colspan="2">Pointer to parent FDR (AU)</td></tr>
<tr><td>32</td><td colspan="2">Pointer to offspring FDR (AU)</td></tr>
<tr><td>34</td><td colspan="2">Number of AU’s allocated for this FDR</td></tr>
<tr><td>36</td><td colspan="2">Pointer to file descriptor index record</td></tr>
<tr><td>38</td><td colspan="2">Extended information</td></tr>
<tr><td>40
42
|
252
254</td><td colspan="2">Data chain pointer blocks</td></tr>
</table>

File name - Contains the file name of the file. The file name can be any combination of up to 10 ASCII characters excluding the space, period and null character. The name is padded with blanks in the case of less than 10 characters.
Extended record length - Contains the record length if it is a data file and has a record length greater than 255 bytes.
File status flags - These flags indicate the type of file and are as follows:

Bit	Description	Values
0	Program/Data indicator	0= Data file, 1= Program file
1	Binary/ASCII indicator	0= ASCII, 1= Binary
2	Reserved	
3	Protect flag	0= Not protected, 1= Protected
4	Backup flag	0= Not modified since last backup, 1= Modified
5,6	Reserved	
7	Fixed/Variable	0= Fixed length records
		1= variable record length

Number of records per sector - Indicates the number of records that will fit into one sector.
Number of sectors currently allocated - Contains the number of sectors allocated for this file. The most significant nibble is located in extended information.
End of file offset - Contains the offset into the highest sector used in the case of variable and program files
Logical record length - Contains the logical record length of the data file. If this value is 0, length is then given in the extended record length bytes 10 and 11.
Number of level 3 records allocated - In the case of fixed length records, this contains the highest record actually written to. In the case of variable length records, it contains the highest sector actually written to. The bytes in this field are in reverse order.
Date and time of creation/update - Gives the date and time the file was created/updated on this disk.
Pointer to parent FDR (AU) - Points to the parent FDR if this is not the top most FDR. If it is the top most FDR this value is 0.
Pointer to offspring FDR (AU) - Points to the AU of the offspring FDR. If this entry is 0, then no offspring exist.
Number of AU's allocated for this FDR - Indicates the number of AU's allocated for this FDR.
Pointer to file descriptor index record - Points to the file descriptor index record which points to this file.
Extended information - This word is divided into 4 nibbles. The left most nibble is the most significant. nibble of the number of sectors allocated to this file. The next nibble is the most significant nibble of the number of sectors actually used in a variable record file. The next nibble is the sector number within an AU pointing to the parent FDR. The last nibble is the sector within an AU pointing to the offspring FDR.
Data chain pointer blocks - This section of the FDR contains the data block pointer clusters. Each cluster is a two-entity. The first word points to the first AU in the data block and the second word points to the last AU in the data block. The number of contiguous AU's allocated by the cluster is the difference plus 1.

SCSI Development Information

The SCSI project was brought to you by the combined efforts of many people:

Don O’Neil	Primary hardware design, board layout, hardware and software documentation
Jeff White	Hardware design, testing and debug
Mike Maksimik	DSR design and programming, MDOS software support
Brad Snyder	Disk manager developer, DSR development assistance
Tim Tesch	MDOS software support
Bud Mills	Assembly, sales and funding

WHT would like to thank Mike Maksimik for his endless hours of coding, debugging and endurance through the development of the DSR.

THREE-MONTH LIMITED WARRANTY

Western Horizon Technologies Inc. extends this consumer warranty only to the original consumer purchaser.

WARRANTY COVERAGE

This warranty covers the electronic and case components of the diskettes and board. These components include all semiconductor chips and devices, diskettes, plastics, boards, wiring, and all other hardware contained in the board("the Hardware"). This limited warranty does not extend to the programs contained on the diskette or cartridge or the accompanying book materials ("the Programs").

The Hardware is warranted against malfunction due to defective materials or construction.

THE WARRANTY IS VOID IF THE HARDWARE HAS BEEN DAMAGED BY ACCIDENT, UNREASONABLE USE, NEGLECT, IMPROPER SERVICE, OR OTHER CAUSE NOT ARISING OUT OF DEFECTS IN MATERIALS OR WORKMANSHIP.

WARRANTY DURATION

The Hardware is warranted for a period of three months from the date of original purchase by the consumer.

WARRANTY DISCLAIMERS

ANY IMPLIED WARRANTIES ARISING OUT OF THIS SALE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE THREE MONTH PERIOD. WESTERN HORIZON TECHNOLOGIES SHALL NOT BE LIABLE FOR LOSS OR USE OF THE HARDWARE OR OTHER INCIDENTAL OR CONSEQUENTIAL COSTS, EXPENSES, OR DAMAGES INCURRED BY THE CONSUMER OR ANY OTHER USER.

Some states do not allow the exclusion or limitation of implied warranties or consequential damages, so the above limitations or exclusions may not apply to you.

LEGAL REMEDIES

The warranty gives you specific legal rights, and you may also have other rights that vary from state to state.

Who to contact if things don't work:

<u>Before calling</u>, please be at your computer and have all revision numbers of <u>all</u> hardware <u>and</u> software on your system available.

Bud Mills Service - Horizon
166 Dartmouth Drive
Toledo, OH 43614
419-385-5946 8am - 8pm EST ask for Bud

Western Horizon Technologies
10225 Jean Ellen Drive
Gilroy, CA 95020
408-848-5947 Anytime, ask for Don