CD Emulator

Publication date: August 1998

Sony Computer Entertainment America
919 E. Hillsdale Blvd., 2nd floor
Foster City, CA 94404

Sony Computer Entertainment Europe
Waverley House
7-12 Noel Street
London W1V 4HH, England

Table of Contents

List of Figures iv
List of Tables iv
About This Manual v
Changes Since Last Release v
Developer Reference Series v
Typographic Conventions vi
Developer Support vi

Chapter 1: System Overview

CD Emulator Introduction 1-3

Chapter 2: Installing the System

Installing the Hardware 2-3
Diskette Contents 2-5
Binaries 2-5
CD boot files 2-6
Documentation 2-6
Samples 2-6
Installing the Software 2-7

Chapter 3: Preparing the Emulation

CDDISK Introduction 3-3
Installing the Emulator 'Boot' Data 3-3
Hard Drive Partitions 3-4
Creating Partitions 3-4
Setting the Active Partition 3-5
Modifying Partition Sizes 3-5
Deleting a Partition 3-5
The RCUBE Demo 3-5

Chapter 4: Generating Emulation Images

BuildCD Introduction 4-3
The Structure of a CD 4-3
BuildCD's Output 4-3
BuildCD Parameters 4-4
Writing BuildCD Control Files 4-5
Global Commands 4-6
Outermost Level Commands 4-10
Disc Commands 4-12
Track Commands 4-17
Volume Commands 4-24
PRIMARY Volume Commands 4-27
Directory Hierarchy Commands 4-41
Directory Commands 4-52
File Commands 4-63
XA Interleaved File Commands 4-72
XA Interleaved Channel Commands 4-78

Chapter 5: Tools and Techniques for Development with the CD Emulator

.CTI File Creation 5-3
CDBoot6x.bin Usage 5-5
2MByte Emulation 5-6
CDEmuPrm Instructions 5-7
cti2cd Instructions 5-9
CutCD Version 1.0 5-10
GenCTI Instructions 5-12
MultiCD Instructions 5-14

Chapter 6: Modifying the Emulation Image

Modifying the Emulation Image-Using UPDATECD 6-3

Chapter 7: Connecting Multiple Emulation Hard Drives

Hard Drive Connections 7-3
Removing the Termination Resistors from the CD Emulator Board 7-5

Appendix A: Sample Control Files

Appendix B: ISO Character Sets

Appendix C: SCSI Errors

List of Figures

Figure 1–1: Emulator SCSI Bus Configuration 1-3
Figure 2–1: CD Emulator Board (Not all chips shown) 2-3
Figure 2–2: CD Emulator Connections 2-5
Figure 4–1: BuildCD program I/O 4-4
Figure 7–1: Emulator with Internal Hard Drive 7-3
Figure 7–2: Emulator with External Hard Drive 7-3
Figure 7–3: Emulator with Internal and External Hard Drives 7-4
Figure 7–4: Emulator with Two Internal Hard Drives 7-4
Figure 7–5: Emulator with Two External Hard Drives 7-4
Figure 7–6: Emulator Termination Resistors 7-5
Figure B–1: ISO 'd' characters (shown unshaded) B-3
Figure B–2: ISO 'a' characters (shown unshaded) B-4

List of Tables

Table 2–1: I/O Base Addresses 2-4
Table 2–2: Executables and Utilities 2-5
Table 2–3: CD Boot Files 2-6
Table 2–4: Documentation File 2-6
Table C–1: SCSI Error Definition C-3

About This Manual

This manual is the latest release of instructional material relating to the CD Emulator as of Run-Time Library Release 4.3. The purpose of this manual is to provide directions on installing and using the CD Emulator.

This CD Emulator has been designed for use exclusively with the PlayStation® Development tool DTL-H2000/H2500. If you incorrectly install this equipment or attempt to connect it to devices other than described herein, you may destroy this emulator and the equipment to which it is connected.

We recommend that you install the DTL-H2000/H2500, libraries and SDevTC development software first, then familiarize yourself with how they function prior to installing the CD Emulator.

Please take a few minutes to read the "System Overview" chapter describing the emulator's design and function. An understanding of the system components and how they function will lead to a quicker and easier installation.

Changes Since Last Release

This document combines the existing CD Emulator manual and the latest CD Emulator ReadMe support document, version 1.13.2, dated 10-28-97. Most updates have gone into Chapter 5 (new): "Tools and Techniques for Development with the CD Emulator."

Developer Reference Series

This manual is part of the *Developer Reference Series*, a series of technical reference volumes covering all aspects of PlayStation development. The complete series is listed below:

Manual	Description
PlayStation Hardware	Describes the PlayStation hardware architecture and overviews its subsystems.
PlayStation Operating System	Describes the PlayStation operating system and related programming fundamentals.
Run-Time Library Overview	Describes the structure and purpose of the run-time libraries provided for PlayStation software development.
Run-Time Library Reference	Defines all available PlayStation run-time library functions, macros and structures.
Inline Programming Reference	Describes in-line programming using DMPSX, GTE inline macro and GTE register information.
SDevTC Development Environment	Describes the SDevTC (formerly "Psy-Q") Development Environment for PlayStation software development.
3D Graphics Tools	Describes how to use the PlayStation 3D Graphics Tools, including the animation and material editors.
Sprite Editor	Describes the Sprite Editor tool for creating sprite data and background picture components.
Sound Artist Tool	Provides installation and operation instructions for the DTL-H800 Sound Artist Board and explains how to use the Sound Artist Tool software.
File Formats	Describes all native PlayStation data formats.

Data Conversion Utilities	Describes all available PlayStation data conversion utilities, including both stand-alone and plug-in programs.
CD Emulator	Provides installation and operation instructions for the CD Emulator subsystem and related software.
CD-ROM Generator	Describes how to use the CD-ROM Generator software to write CD-R discs.
Performance Analyzer User Guide	Provides general instructions for using the Performance Analyzer software.
Performance Analyzer Technical Reference	Describes how to measure software performance and interpret the results using the Performance Analyzer.
DTL-H2000 Installation and Operation	Provides installation and operation instructions for the DTL-H2000 Development System.
DTL-H2500/2700 Installation and Operation	Provides installation and operation instructions for the DTL-H2500/H2700 Development Systems.

Typographic Conventions

Certain Typographic Conventions are used through out this manual to clarify the meaning of the text. The following conventions apply to all narrative text except for structure and function descriptions:

Convention	*Meaning*
`courier`	Indicates literal program code.
Bold	Indicates a document, chapter or section title.

The following conventions apply within structure and function descriptions only:

Convention	*Meaning*
Medium Bold	Denotes structure or function types and names.
Italic	Denotes function arguments and structure members.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In North America	***In North America***
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd., 2nd floor Foster City, CA 94404 Tel: (650) 655-8000	E-mail: DevTech_Support@playstation.sony.com Web: http://www.scea.sony.com/dev Developer Support Hotline: (650) 655-8181 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In Europe	*In Europe*
Attn: Production Coordinator Sony Computer Entertainment Europe Waverley House 7-12 Noel Street London W1V 4HH Tel: +44 (0) 171 447 1600	E-mail: dev_support@playstation.co.uk Web: https://www-s.playstation.co.uk Developer Support Hotline: +44 (0) 171 447 1680 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT or BST/BDT)

Note: SN Systems designed and built the CD Emulator software and hardware. If you have any questions, feel free to send them e-mail:

E-mail: support@snsys.com .

Web Site: http://www.snsys.com.

Chapter 1:
System Overview

CD Emulator

CD Emulator Introduction

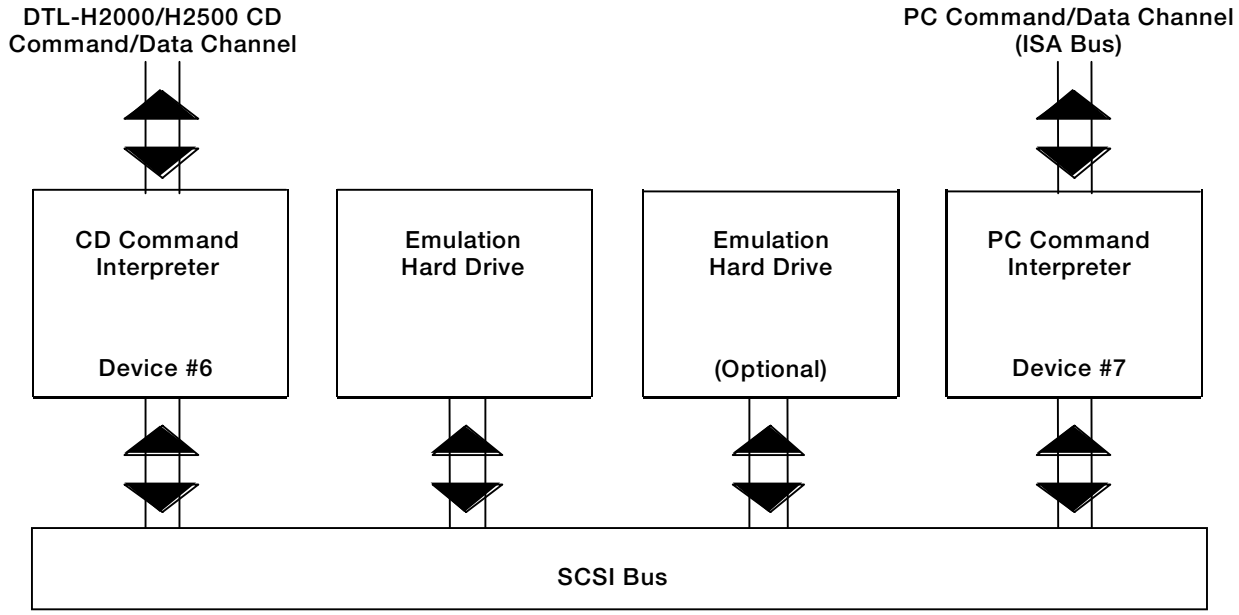
Once installed, the emulator will provide you, the user, with a device that mimics the functions of a real CD-ROM drive. Normally the command signals generated by the DTL-H2000/H2500 would go directly to a CD-ROM drive, which would respond and retrieve data from a CD. However, the DTL-H2000/H2500 can be configured (under software control) to switch these command signals so that they are sent to the CD emulator. The emulator then responds as if it were a CD-ROM drive, but in this case retrieving data from a hard drive (HD) connected to the emulator.

This emulator consists of three (logically) different subsections:

- The emulation HD(s)
- An interface to the PC
- An interface to the 'CD command channel'

Within the electronics of the emulator we have implemented a SCSI bus that connects each of the above sections in the following way:

Figure 1–1: Emulator SCSI Bus Configuration



Up to eight 'devices' can be connected to a SCSI bus. In this design we have used two of the eight 'devices' to implement command channels to and from the PC and the DTL-H2000/H2500, leaving six free for connection to emulation HDs. We do not anticipate many users will need more that one HD. However, if you do wish to connect more than one HD then please refer to Chapter 7.

The emulator has been designed so that either external or internal HDs can be connected: internally using the 50 -50 ribbon connector and/or externally using the 1m cable. External connection is very simple using the cable provided.Internal connection requires a little more work in mounting the HD in a free drive bay and routing of the cable.

To be able to emulate a high speed CD drive a certain data transfer rate needs to be maintained between the emulation HD and the DTL-H2000/H2500 CD command channel. We have found that some HDs cannot maintain this data transfer rate. The manufacturers' specifications for HDs declare an average rate which may seem enough to satisfy the emulator's requirements. However, most drives need to recalibrate themselves periodically to account for thermal changes. During this period the data rate on some drives falls

off and leads to stalls in the emulation data flow. This is observed as a glitch in film playback or an interruption in sound replay.

The following hard drives are suitable for use as CD-Emulator HDs. This list is current as of August 1997:

DCAS-34330 ULTRA SCSI 4.3GB/IBM
Fireball ST-3200S ULTRA SCSI 3.2GB/Quantum
Fireball TM-3200S ULTRA SCSI 3.2GB/Quantum
Atlas XP-4550S ULTRA SCSI 4.5GB/Quantum
Medalist ST52160N ULTRA SCSI 2GB/Seagate
Barracuda ST32171N ULTRA SCSI 2GB/Seagate

These were tested on the following PC models:

GL5200ST/DEC
GL5133ST/DEC
GXPRO6200/DELL
MILLENNIA-C/MICRON
FMV-6200T5/FUJITSU
G6-200 MALTI MEDIA/GATEWAY
P5-200/GATEWAY

Warning: The list above is not comprehensive, and does not include data about whether a DTL-H2500 or DTL-H2000 was used in the test. Therefore, if you have a DTL-H2500 board, please don't run out and buy a Gateway or a Micron, because these have had problems with DTL-H2500 software in the past.

Now that you have a basic understanding of the system the next thing to do is get it installed and working. The following chapter describes the following installation steps:

- Configuring your emulator board.
- Plugging the card into the PC.
- Connecting the emulation HD.
- Connecting the emulator to the DTL-H2000/H2500.
- Installing the software.
- Setting up the emulation HD to be ready for image building and emulation.
- Building and testing the example program.

Once these steps are complete, the CD emulator is ready for use.

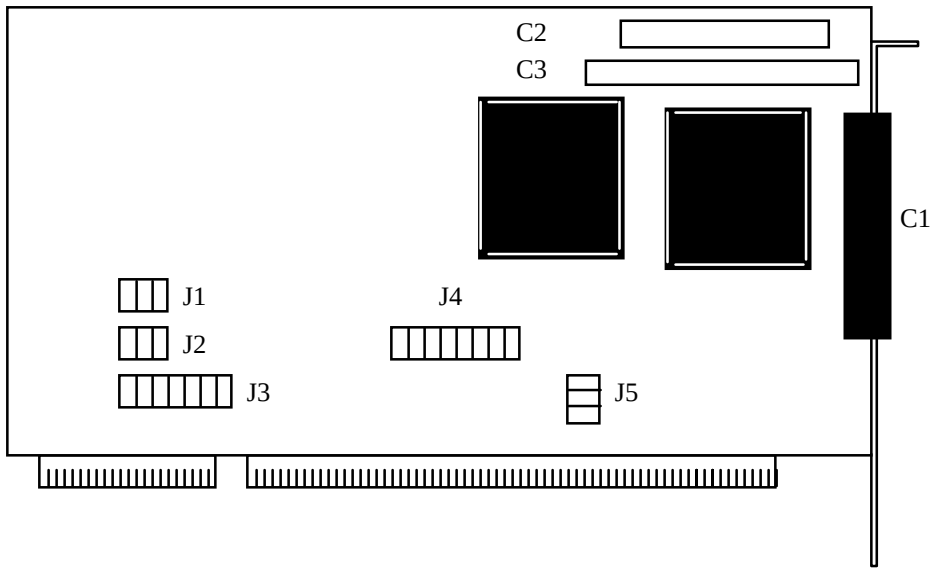
Chapter 2:
Installing the System

CD Emulator

Installing the Hardware

Follow the steps below to install the CD Emulator Board. Turn off your PC and disconnect it from the power supply. Remove the cover and check all the cards in your system, noting their base addresses, irq settings and dma channel usage. Determine which channels/addresses are free to set the jumpers on the emulator board appropriately.

Figure 2–1: CD Emulator Board (Not all chips shown)



CAUTION: This board is sensitive to static electricity; hold by the metal support bracket when handling it.

The Connectors

C1	External SCSI connector
C2	CD command channel (connects to DTL-H2000/H2500)
C3	Internal SCSI connector

The Jumpers

J1 & J2	DMA channel select (both must be set the same). Possible settings are (from left to right) 7, 6 or 5.
J3	Interrupt request setting. Possible settings are (from left to right) 15, 12, 11, 10, 7 or 5.
J4	I/O base address select. Possible settings are (from left to right) 300, 308, 310, 318, 380, 390, or 398.
J5	SCSI ID device number (also software selectable). Possible settings are from 7 - 0. This is a 3 bit binary selection. With the top two jumpers closed, the ID will be 6.

1. Set and confirm the emulator board I/O, DMA, and IRQ jumpers.

The default settings have been chosen so that the possibility of contention with other internal boards is minimized. Nevertheless, care should be taken that settings on the CD emulator board do not conflict with any other cards in your system.

The factory default setting for the I/O base address is 308 (decimal notation), which also happens to be the default setting of the DTL-H2000. 308 is 134 in hexadecimal notation, which actually corresponds to the hexadecimal address of 1340. This odd notation conversion scheme is explained by the following: although

the emulator board's actual address is in 4 byte *hexadecimals*, the DIP switch host's A15 - A4 (3 bytes) are in *decimal* format. The actual addresses and a table of their equivalents are entered below:

Table 2–1: I/O Base Addresses

Decimal Notation	Hex Notation	Actual Address (in hex)	Remarks
300	0x12C	0x12C0	
308	**0x134**	**0x1340**	**Default**
310	0x136	0x1360	
318	0x13E	0x13E0	
380	0x17C	0x17C0	
388	0x184	0x1840	
390	0x186	0x1860	
398	0x18E	0x18E0	

In this case, take A15-A4 from 0x1340 and match it with 0x134 to get "308".

You must also set the DMA channel jumpers. The numbers 5 and 7 on the board can sometimes be printed in reverse, so be careful --- the jumper to the farthest left is 7.

Finally, set the interrupt level jumper J3; the factory default is 15.

J5, the SCSI ID device number setting, comes set to 7 (all three jumpers connected). This is the PC command channel device number. See Fig 1-1. Note: this setting is also configurable from the command line.

2. Install the emulation board in the PC/AT Interchanger ISA Bus expansion slot. Insert the emulation board into an available ISA slot next to the DTL-H2000 PIO board or the DTL-H2500 board. As shown in Figure 2-2, connect C2 to the similar socket on the top edge of the PIO board, using the flat connection cable provided. Since the emulation board and the development CD-ROM (either the DTL-H2010 or the DTL-H2510) are able to coexist, there is no need to detach the CD-ROM drive.

3. Connect the emulator-dedicated HD to the emulator board. Both internal and external connection methods are possible. As of October, 1997, we recommend the Micropolis 4110AV, the IBM Spitfire (0662), or the IBM Pegasus. The IBM drives require that the auto start jumper be set prior to use.

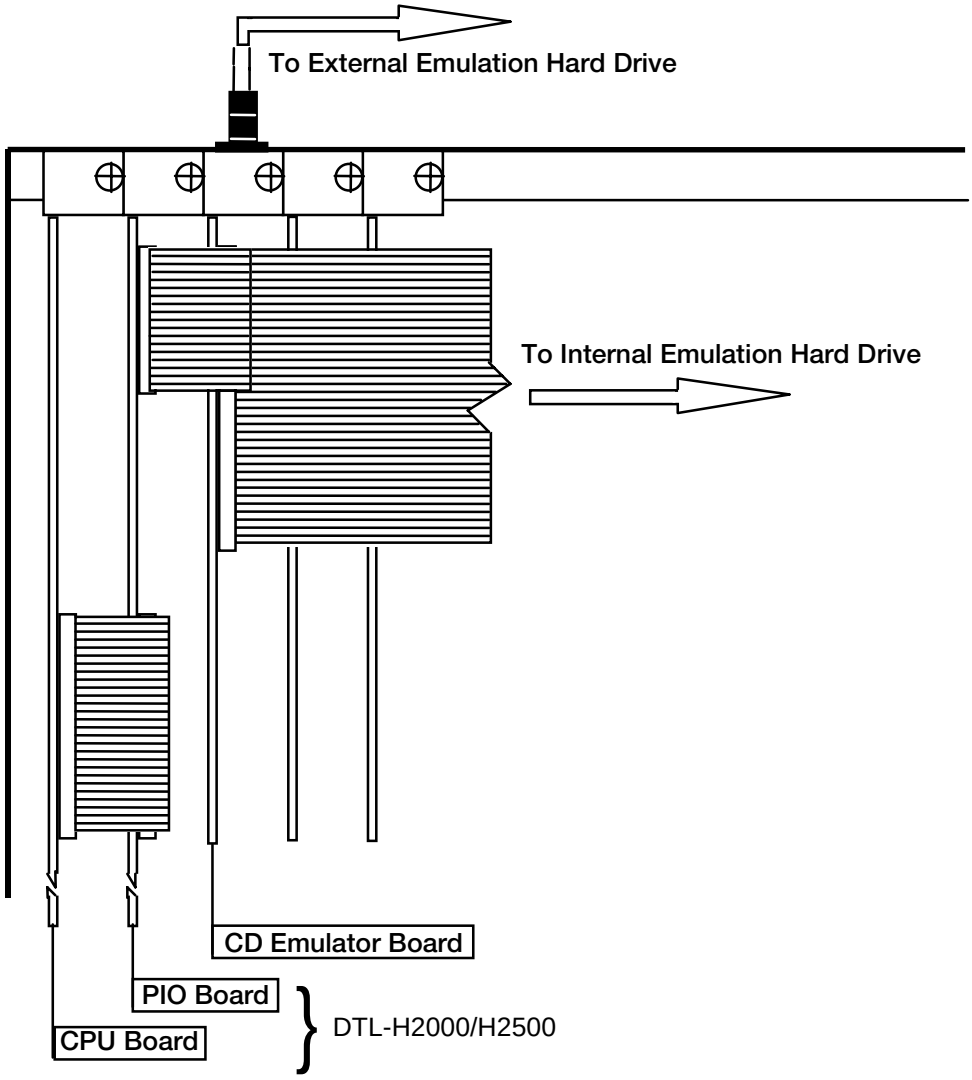
If you intend to use an internal HD with the emulator, mount it in a free bay. Connect it to the emulator board using the large ribbon cable provided (connector C3). Normally there is a free power connector for extra drives within the PC. Connect it to the power socket of the HD. Note the SCSI ID setting of the drive. You will need this later. Refer to manufacturer's instructions on how to set this.

Replace the cover on the PC.

If you are using an external drive, connect it now.

NOTE: Never connect or disconnect any hard drives without first removing **all** sources of power.

Figure 2–2: CD Emulator Connections



Diskette Contents

A brief description of the contents of your diskette appears in the sections below. Files have been zipped with sub-directories. Use pkunzip -d <file> version 2.04g or the latest version of WinZip to unzip the files.

Binaries

The following CD-emulation executables and utilities provided by SN Systems appear in the directory **cdemu\bin** on your diskette.

Table 2–2: Executables and Utilities

File	Version	Description
bcdflat.exe	2.41	BETA version of flat model for buildcd.
buildcd.exe	2.41	Build image program
ccs2cti.exe	1.10	CCS (cdgen output) to CTI file converter.
Cdbios.com	1.04	Emulator board communication driver.

Cddisk.exe	1.17	Emulator HD disk utility
cdemuprm.exe	1.00	Allows you to select normal or zero seek timings; and also control errors within CD streams. Both settings are unaffected by RESETPS (which causes the CD emulator's micro
controller		to reload CDBOOT from the SCSI HD). Requires at least CDBOOT35/CDBOOT65.
cdmon.com	1.03	TSR program for debugging.
cti2cd.exe	1.03 beta	Bug fix for Yamaha audio track cutting.
Cutcd.exe	1.02 beta	YAM CDR100/SONY CDW900E cutter utility from CD emulator image.
Gencti.exe	1.0 beta	GenCTI generates a ".cti" file based on the directory structure on your PC.
dos4gw.exe	1.97	Dos extender (for bcdflat)
multicd.exe	1.01	Multi CD emulator Utility. Allows simulation of CD emulator image changing and door opening.
updatecd.exe	1.18	Update the build image utility.

CD boot files

Boot files are binary files that contain the initialization routines for the CD-emulator to use on start up. You install them whenever you need to format the CD-emulator dedicated hard-drive. For instructions on updating your files, read the section "CDBIN3x updates" in this document.

Note: If you install version 3.x of the boot code then no information will be passed back to the PC but you do not need to worry about the interrupt at all.

Table 2–3: CD Boot Files

File	Version	Description
CDBOOT36.bin	3.6	Emulator Boot code.
CDNOSK3x.bin	DELETED	For old timers: The reason there is no CDNOSK3x.BIN any more is that, with the advent of version CDBOOT3.6, the seek timings are now programmable using the utility called CDEMUPRM.
cdboot66.bin	6.6	Emulator Boot code, identical to version 36.bin, except that version 6.6 reports information back to the PC so that the PC can show the current CD activity on screen. This is achieved with the use of the CDMON.COM TSR.

Documentation

The following file appears on your diskette in the directory **CDEMU\DOC.**

Table 2–4: Documentation File

File	Version	Description
cdemul.pdf	-	CD Emulator Manual

Samples

CDEMU\Sample\Basic. This tutorial is a very basic overview of how to use the CD-Emulator. Compiles with Library 4.0 and later; it may also compile with earlier versions of the library.

CDEMU\Sample\Rcube. This is an example that you can use on your emulator. This is the standard sample provided by SN Systems. You cannot compile this sample, as source code was not provided.

CDEMU\Sample\GenCti. This sample directory contains a sample "Test.cti" output file from the "GenCTI" program. Refer to the "GenCTI" instructions in the "readme.doc" of this distribution for more details on the syntax of GenCTI.

Please note that "GenCTI" was never updated beyond 1.0beta. For more details on the CTI syntax, refer to the CD Emulator manual.

Installing the Software

Install the software. Copy the CD emulator software on the diskette to your computer. We recommend copying the files to `c:\ps\psx\cdemu`.

`DEXBIOS.COM` is a driver which you will be familiar with. It allows the code development system to communicate with the DTL-H2000. In the same way, `H25BIOS.COM` allows communication with the DTL-H2500. Similarly, we have provided `CDBIOS.COM` to provide communication services to the CD emulator (in addition to `DEXBIOS.COM` or `H25BIOS.COM`, not as a replacement). **Modify the autoexec.bat** file to load `CDBIOS.COM`. Your autoexec.bat file should contain the line listed below.

```
[full path to "cdbios.com"]\cdbios /aXXX /iXX /dXX
```

where

aXXXX refers to I/O base address select on the CD-Emulator board. Possible settings are (from left to right) 300, 308, 310, 318, 380, 388, 390, or 398. Refer to the table in Step 1 of Installing the Hardware for the actual addresses to which these correspond.

iXX refers to Interrupt request settings on the CD emulator board. Possible settings are (from left to right) 15, 12, 11, 10, 7, and 5.

dXX refers to DMA channel selects on the board. Possible settings are (from left to right) 7, 6 or 5.

For example,

```
c:\ps\psx\cdemu\bin\cdbios /a398 /i11 /d5
```

IMPORTANT. Care should be taken that the settings on the CD emulator board do not conflict with any other cards in the system. For example, `CDBIOS.COM` must be installed at a different address from a previously installed `DEXBIOS.COM` or `H25BIOS.COM`. The interrupt you choose should be used exclusively for the emulator.

Modify your path variable. Edit your autoexec.bat file to contain the path to the CD-emulator binaries. For example, you could add the line

```
set path=c:\ps\psx\cdemu\bin;%path%
```

if the CD-emulator binaries are in c:\ps\psx\cdemu\bin.

Reboot your machine. Make sure that the CD emulator board is connected to either an external or internal hard drive. Power on the external hard drive, if any, **BEFORE** powering on the PC.

Open up a DOS-console box and run the appropriate driver for your development boards. Run `DEXBIOS.COM` (DTL-H2000 ISA boards) or `H25BIOS.COM` (DTL-H2500 PCI boards) as usual, by typing in "dexbios /aXXXX /iXXX" or "h25bios /aXXXX /iXXX". Refer to the DTL-H2000 or DTL-H2500 manuals for more details.

CD Emulator

Chapter 3:
Preparing the Emulation

CDDISK Introduction

CDDisk is the general emulation disk manager program. It performs the following functions:

- Initialization of the HD
- Installation of the emulator board 'emulation program'
- Partitioning of the HD(s) into pseudo 'Cds'
- Setting the 'active' partition.

It operates in one of two modes, direct command line actions or menu mode and takes the general command line form

```
CDDISK [option] [disk SCSI ID]
```

Valid options are

`-a<part_id>`	Set the active partition (no menus).
`-b<filename>`	(Re)Install the emulation program (no menus).
`-n`	Disk is new. Initialize the partition sector.

where

'`disk SCSI ID`' is the ID number set on the emulation HD.

Generally HDs are shipped with an ID of zero and can be plugged without change into the emulator board. However, when using more than one HD, special installation instructions need to be followed (see Chapter 6) and the ID of each one must be different from each of the other 'devices' on the local SCSI bus.

(Free IDs are 1, 2, 3, 4, and 5)

For the following examples it will be assumed that the HD has an ID of 0.

When the system is first installed, or a new HD is added, it must be initialized with the following:

```
CDDISK -n 0
```

This places the disk into a known state ready to accept emulation images. The user will then be asked to confirm this action because ALL information previously written on it will be destroyed.

Installing the Emulator 'Boot' Data

The 'cd command/data device' will, upon reset, make contact with each device on the bus in turn from device 0 to device 7. It is looking to identify an emulation HD.Once found, it then looks for a 'boot' partition on that device. Data in this (small) partition is then loaded into the emulation card. This is the emulation program that interprets the signals coming from the DTL-H2000/H2500. Booting in this way allows for future upgrades to the emulation system should they be necessary.

Only one of these 'boot' partitions needs to exist in multiple HD configurations, and it is recommended that it be placed onto the HD with the lowest SCSI ID.

NOTE: Without this 'boot' partition in place, the emulator CANNOT function.

Assuming the emulator software was installed as described above, then to install the 'boot' program directly:

```
CDDISK -bC:\PS\PSX\CDEMU\BIN\CDBOOTxx.BIN 0
```

If this command was completed successfully, it will return without errors.

To install the boot program using the menu system:

```
CDDISK 0
```

Please select the ‘Load Boot Program’ option from the menu.

Type in the full pathname to the file `CDBOOTxx.BIN`

```
e.g.   C:\ PS\PSX\CDEMU\BIN\CDBOOTxx.BIN
```

If all is well, no errors will be reported.

You only need to do this once, unless you receive updates of cdboot*xx*.bin.

If you need to update a previously existing boot file, either use the 'Load boot program' option from CDDisk, or perform the following steps:

1: Launch cdbios: cdbios /a<address> /I<interrupt> /d<dma channel>. For example,

```
cdbios /a398 /i5 /d7
```

2: If `DEXBIOS` or `H25BIOS` are running, then stop the emulation, which is necessary since the emulator hogs the SCSI drive. Do this by typing the following commands:

```
RESETPS 1     (allows DEX target to be selected)
RUN SelCD     (suspends emulation)
```

3: Install new CDBOOT using CDDISK.

```
CDDISK -bCDBOOTxx.BIN <SCSI id>
```

where "xx" is the version number, and where `SCSI id` is the SCSI id number of the emulator HD.

4: If `DEXBIOS` or `H25BIOS` are running, then restart the emulation:

```
RESETPS 1     (forces emulator to load new CDBOOT code from SCSI drive)
RUN SelEmu    (restarts emulation)
```

Hard Drive Partitions

Creating Partitions

A partition on the emulation HD is effectively a pseudo CD. It is not limited to being a standard CD size of 72 minutes for example but can be any size. This provides flexibility for the developer who may only require an effective 10 minute CD but wishes to have 5 versions on one CD of up to 99 minutes. (There is little point in defining a partition larger than this because the CD directory structure does not cater to it.)

Partitions can be defined in units of either MBytes, Frames (CD frames) or Sectors.

One Frame is 5 sectors (allowing for extra information needed for emulation). There are 75 frames per second. From this information you can calculate the size of partition you require.

For example, a 60 minute CD will require a partition size of 270,000 frames or 1,350,000 sectors or 660 MBytes.

To create the first partition on disk 0, use

```
CDDISK 0
```

Please select the ‘Create Partition’ option from the menu.

Select the size type you wish to define this partition in. 1 for sectors, 2 for MBytes and 3 for frames. We will define this one in frames, so enter 3 and then type in the size. For this example we will define a 60 minute CD needing 270,000 frames.

To allow for easy identification you will now need to provide a partition name.

e.g. 60 Min CD.

You will now see this added to the list. Partition 0 is the boot partition and partition 1 is 60 Min CD.

Setting the Active Partition

The act of creating the above partition has made this the currently active partition. This means that when the DTL-H2000/H2500 issues CD commands it is from currently active partition that the data is read. Only one partition on each HD can be marked active at any one time. This can be viewed as being equivalent to inserting a real CD into a drive.

If you have more than one HD, the active partition on the HD with the lowest number ID will be considered as the current partition. Activating partition 0 (the boot one) effectively deselects all partitions on an HD.

To set an active partition - e.g. Disk 0 partition 1,

```
CDDISK -a1 0
```

or

```
CDDISK 0
```

and select the 'Select active partition' option from the menu and type 1.

You will notice that the currently active partition is indicated in red.

Modifying Partition Sizes

Partition sizes can be modified only through the option menus. Select the partition you wish to resize and enter the new size (only in sectors!).

NOTE: Resizing a non-active partition will NOT set it as the active partition.

Deleting a Partition

Select the partition number of the partition you wish to delete and confirm the action.

The RCUBE Demo

Build the image of the RCUBE demo executable onto the hard drive. The following steps will install the **RCUBE demo** included on your diskette. We assume the CD emulator board has these settings:

I/O addr = 0x390
IRQ = 5
DMA = 7
SCSI ID of connected internal or external HD = 4.

At an MS-DOS command window, type

```
cd c:\ps\psx\cdemu\sample\rcube
buildcd -mcd.map cd.cti -s4:1
```

where

`cd.cti` = the RCUBE demo cti file included on your diskette

`-m` = option that enables map file output. In this example, the output is "cd.map"

`-s4:1` = option that forces the image to be output to SCSI device at SCSI "4" in partition "1".

`buildcd` will run in the MS-DOS window. Ignore the warning message #66. When **`buildcd`** finishes successfully, exit the program by hitting ESC on the keyboard. The result of **`buildcd`** is an image of the

CD-ROM on the emulator hard disk. The generated image is an ISO-9660 formatted MODE 1 CDROM, and the files `PSX.EXE`, `RCUBE.EXE`, `RCUBE.TIM`, and `RCUBE.TMD` are on Track 1.

Run the RCUBE demo. After building the image onto the CD drive, you may now execute the program. These steps can be used to launch your examples. You should already have `H25BIOS.COM` or `DEXBIOS.COM` running as a TSR in the background.

```
resetps 1
pause
run <dir>:\psyq\bin\snpatch   Warning: this is only for DTL-H2000 boards!
pause
run <dir>:\psyq\bin\selemu
```

This command instructs the DTL-H2000 to boot up from the CD-Emulation HD. (To instruct the DTL-H2000 or the DTL-H2500 to boot up from the CD-ROM unit, execute `run c:\psx\bin\selcd`. The `selemu` utility needs to be executed prior to first emulation session. It doesn't need to be executed before every emulation session, as long as `selemu` has been executed once before.)

Now you are ready to launch the program:

```
pause
run <dir>:\pssn\bin\cdexec
```

You should notice the green LED flashing on your attached emulation HD. After a long interval the colorbars on your NTSC monitor disappear and the RCUBES demo appears on screen.

To quit this program, you must press the rectangular button.

If a program does not start, ensure that you have quit a previously running program. You may also type `RESETPS 1` at any time to reset the ISA boards and the CD emulator board.

Controller button naming conventions:

The PlayStation controller has 14 buttons arranged, as follows:

Lpad:	Left pad (left set of four buttons, located on top of controller, similar to D-PAD)
Rpad:	Right pad (right set of four buttons, located on top of controller, four circular buttons)
Lflip:	Left side (set of two flipper buttons on the left side)
Rflip:	Right side (set of two flipper buttons on the right side)
Select:	Rectangular button (located on top between Lpad and Rpad)
Start:	Triangular button (located on top between Lpad and Rpad)

RCUBE:	3D Texture map, transparency, lighting example
Rpad Left:	Toggle between black background and fog effect
Rpad Up:	Toggle between texture maps, lighting effects, etc.
Lpad:	Rotation
LFlip:	Explode cubes
RFlip:	Reset cubes

Chapter 4:
Generating Emulation Images

BuildCD Introduction

BuildCD is a program to create CD emulator images and ISO9660 files. It takes source files at various levels, automatically generates directory and marker information and generates the relevant subcode information.

The program requires users to have a reasonably good idea of the structure of a CD, as the control language is quite low-level. The command set is structured to force users to work in a manner consistent with CD structure.

Central to the running of the BuildCD program is the concept of the control file. This defines the order and structure of the intended Compact Disc and directs the program as to the outputs it should produce.

The Structure of a CD

A CD is arranged as a continuous anti-clockwise spiral of data leading out from the centre of the Compact Disc. The smallest logical division of this spiral is a frame, this being the amount of data required for 1/75th of a second's playback. A frame contains 2352 bytes of data and 98 bytes of subcode.

The CD spiral is divided logically into tracks. The track nearest to the centre is called the leadin track. It is followed by a number of data tracks (up to 99) and the they are then followed by a leadout track. The leadin contains the CD's table of contents stored in the subcode (which is what a CD player spins up and reads when a new CD is put in). The table of contents details the positions of all the tracks on the disc in terms of time offset from the start of the disc. BuildCD automatically generates a table of contents. The last track, the leadout track, is just a marker to flag the end of the disc. Neither the leadin nor leadout track should have any data in it.

All tracks other than the leadin and leadout tracks can be used freely for holding data. They can be standard audio tracks or can hold data in a variety of Modes.

Mode0 frames can only hold empty records.

Mode1 frames can hold up to 2048 bytes of data and automatically include cyclic redundancy checks and error correction. For critical data where data integrity has to be maintained (such as executable programs), Mode1 should always be used.

Mode2 frames can hold up to 2336 bytes of data but afford no cyclic redundancy checks or error correction. For non-critical data where data integrity needed not be worried about too much (such as graphics), Mode2 frames can be used.

XA frames are a modification of the Mode2 frame standard and allow two types of data records that can be freely interleaved.

Form1 mirrors Mode1 in that it can hold up to 2048 bytes of data and automatically include cyclic redundancy checks and error correction.

Form2 mirrors Mode2 in that it can hold up to 2324 bytes of data without error correction.

BuildCD's Output

BuildCD takes a control file as primary input, pulling all specified files as and when required. The outputs from this program are all for specific purposes, as detailed below. None of the outputs will be produced without the correct command line switch being present when the executable is invoked. The inputs and outputs for the BuildCD program can be viewed as:

Figure 4–1: BuildCD program I/O

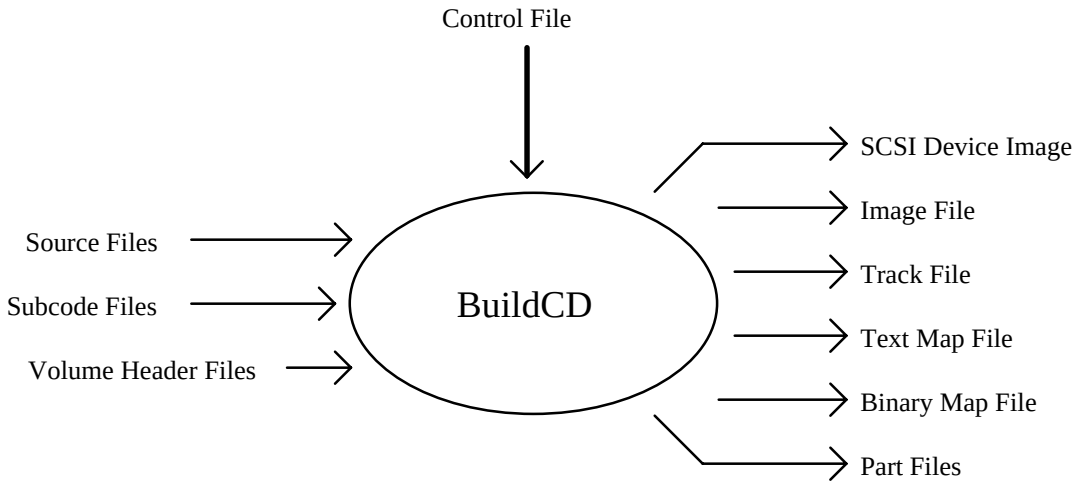


Image File	An image file is a complete representation of all CD data. This is the final output of the BuildCD program and can be used for future cutting onto disc via a CD writer.
SCSI Device Image	An image file can be written automatically to the SCSI drive resident in the CD emulator. Written as a SCSI device image, it can be used by the CD emulator directly.
Track File	A track file is a representation of an ISO9660 volume. This type of file output can be written directly into a data track. However if any of its contents were to change, it would have to be recreated.
Text Map File	A text map file is a textual representation of what has gone into the image file. This can be used by the UpdateCD program to check the source file datetimes and sizes to ensure that the image is up-to-date, and on finding any out-of-date sections, automatically updating the relevant files. A text map file will only be written if an entire image is being created.
Binary Map File	The Binary Map File is a binary representation of the image, specifying where the source files and part files of the image can be found. This file is for use by the Software and Hardware emulators to build images on the fly.
Part Files	The part files are automatically generated sections of track files that have to be created (rather than the source files that are already resident on disk). In fact, these are binary representations of the CD Directories, PathTables and Volume headers. These part files are for use in conjunction with the Binary Map File. Flagging the output of the Binary Map File will automatically cause the creation of the part files.
SonyCD Generator Save Files	The CD generator save files can be used by Sony's CD Generator program to import the CD specification creation of CDs prior to direct writing from within the BuildCD program.
Interleaved Output Files	Interleaved output files are in the same format as those files imported using an XASource command within the control file command language. Interleaved files are produced only when a `-g` command line argument is used and the output filename is specified in the XAInterleavedFile control file command.

BuildCD Parameters

BuildCD has the following parameter structure:

```
BuildCD [options] [[[d:]path]control_filename.ext]
```

As is normal for DOS, options are distinguished from normal parameters by being preceded by either a "-" or "/" character. The options available to the BuildCD program are detailed on the help screen which is displayed whenever the program is invoked without any parameters. The valid parameters are:

`-1`	Ensures that all file and directory names on the CD are checked for being ISO level 1.
`-2`	Ensures that all file and directory names on the CD are checked for being ISO level 2.
`-b[<FileName>]`	Enables the output of any Binary Map File specified within the Control File.
`-d<var>=<var>`	Predefines a string substitution to be used upon the Control File.
`-e<num>`	Restricts the number of error messages output.
`-g[<FileName>]`	Enables the output of Sony CDGenerator save files and enables output of InterleavedFile file output.
`-i[<FileName>]`	Enables the output of the Image File.
`-m[<FileName>]`	Enables the output of any Text Map File specified within the Control File.
`-p[<dir>]`	Causes output of Part Files. If a directory is specified then they will be output there. The Part Files will be automatically output if Binary Map File output is enabled.
`-s<s_id>:<p_id>`	Causes output of the image to a SCSI device. s_id is the SCSI ID and the p_id is the partition ID.
`-t[<FileName>]`	Enables the output of Track Files.
`-w`	Suppresses all warning messages.

Writing BuildCD Control Files

It should be noted that the structure of the control file is dependent on the level at which the command is being specified. There is a hierarchy of levels into which the program descends and from which it then rises. The easiest way to see how a control file has to be specified is to look at the examples in Appendix A. These examples show all the allowable commands at least once and also show the two entry points to the command sets.

It should be noted that all commands are case insensitive, i.e. the case of the command is ignored although error messages do faithfully echo the case of characters within the control file.

BuildCD has been written to run in conjunction with UpdateCD, a program that works from the Text Map File to ensure that the contents of the image and all other output files are still up-to-date. UpdateCD cannot move internal image file structures around. If it had to do this, the increase in size of a source file at the start of the image would cause a ripple effect that would take longer than writing the image from scratch. UpdateCD will simply inform the user that the source file has outgrown its allotted area, and stop before writing anything. An image can however be written with extra space to allow for this eventuality, using the MinLength and AddLength commands at the File Command level. If a file is likely to grow, then it is suggested that these commands be used.

Another 'trick' that has been used by developers takes place when they know in advance that more files than exist at the moment are going to be required. The incorrect solution adopted has been to build an image with the 'vapour' files being sourced from existing files, in the hope that the source for these files can be later swapped. Unfortunately this would cause an inconsistency with the existing control file and therefore is not allowed. The correct way to prepare for the future source files is to create dummy files, potentially even empty files, and again use the MinLength and AddLength functions to ensure that there is enough space for the eventual files to reside in. When the files have been created they need only be copied over the dummy files for the UpdateCD program to correctly load them.

Global Commands

Global commands are valid at any point in the control file. They fall into the following categories:

Defining of default values and substitution strings.	`Define`, `GreenwichOffset`
Status message output during program execution.	`Echo`, `ShowDefines`
File inclusion.	`Include`

Define

Function

Defines substitution variables for replacement within the control file(s)

Syntax

Define <var> <value> [<var> <value> [...]]

Default

BuildCD has the following predefined variables:

program	Executing program name
version	Current program version number
edition	Current program edition number
second	Present DOS second of minute
minute	Present DOS minute of hour
hour	Present DOS hour of day (24hr clock)
day	Present DOS day of month
month	Present DOS month of year
year	Present DOS year
weekday	Present DOS day of the week (Monday = 1, ... , Sunday = 7)
yearday	Present DOS day of the year

Remarks

All strings enclosed within square braces in the control file will be substituted for the values previously defined. If a substitution is attempted for a variable that is undefined then an error will be output.

Substitutions are not recursive, nor can they be nested. When the program encounters a replacement string it looks for the first closing square brace and then attempts to match all characters between. Matches are case insensitive. Once matched, the string and the square braces are replaced by the defined replacement string.

There is no limit (barring memory limitations) to the number of strings that can be defined or the number of substitutions allowable in a single line.

Substitution strings can also be defined in the command line using the `-d` option.

Substitution strings can be redefined at any point, to effectively undefine a string, it has to be redefined to a null string. For example:

```
Define PsyQ  ""
```

Example

```
Define PsyQ c:\psyq\bin
```

See also: Command line arguments

Echo

Function

Outputs all arguments to the Echo window of the output screen.

Syntax

Echo <argument1> [...]

Remarks

All arguments are echoed to the appropriate output window. Output is not buffered.

All substitutions will have been performed before output.

All comments are ignored.

Example

```
Define PsyQ c:\psyq\bin
Echo PsyQ var = [PsyQ]
```

would output the string:

```
PsyQ var = c:\psyq\bin
```

GreenwichOffset

Function

Sets the default Greenwich offset which is to be used whenever a date is specified without an explicit value.

Syntax

GreenwichOffset <OffsetValue>

Default

The Greenwich offset defaults to a value of 0.

Remarks

The Greenwich offset is a value defined within the ISO9660 document. Basically, it is a time offset from Greenwich Mean Time based on the number of 15 minute intervals that the author's time zone is from London. Greenwich offsets range from -48 (west) to 52 (east).

Typical examples are:

New York	-20
Tokyo	32

Example

```
GreenwichOffset -15
```

Include

Function

Includes the contents of the specified file into the control file at the specified point.

Syntax

Include <FileName>

Remarks

Include files can be nested. There is no imposed limit to the number of nested include files, stack space and memory allowing.

Example

```
Include psyqgame.cti
```

ShowDefines

Function

Outputs details of all defined substitution strings to the Echo window of the output screen.

Syntax

ShowDefines

Remarks

All defined substitution strings are output to the window whether they are defined in the control file, defined in the invocation arguments or predefined within the program. Output is not buffered.

Example

```
Define PsyQ c:\psyq\bin
ShowDefines
```

would result in a typical output of:

```
version      1.00
edition      1.00
program      C:\PSYQ\BIN\BUILDCD.EXE
second       23
minute       02
hour         15
day          27
month        07
year         1994
weekday      03
yearday      208
PsyQ         c:\psyq\bin
```

See also: Define

Outermost Level Commands

Outermost level commands are the highest level of command for a control file. Only global commands can be recognized before an outermost level command.

There are two outermost level commands Disc and Volume. The Volume command at this level is identical to the Volume command at the track level of a disc definition.

Disc

Function

Marks the start of the definition of a disc and defines the primary output file for the program.

Syntax

Disc <DiscType> [<ImageFileName>]

Remarks

Valid disc types are:

CDDA, CDAUDIO or AUDIO	Audio discs
CDROM or ROM	Data and audio discs
CDROMXA or XA	CDROM XA discs
CDROMXA_PSX or XA_PSX	Standard XA discs with PSX restrictions and extensions

The Disc statement specifies the default image output file name. No image file will be written unless the `-i` command line parameter is used when the program is invoked. Any image file name specified in the command line will override this image file name if specified.

For a PSX_XA disc there are certain restrictions on the general XA format. Most notably:

- The hierarchy path tables must all appear and be in a fixed order
- Volume descriptors can only be written once
- Only primary and termination volume descriptors can be defined
- System identifier must be PLAYSTATION
- SubSource definitions cannot be made
- No Mode1 track can appear within the image

The Disc statement enables the Disc commands until an EndDisc statement is encountered.

Example

```
Disc CDROM state.qmu
```

Volume

Function

Marks the start of the definition of a track volume and defines the primary output file for it.

Syntax

Volume <VolumeType> [<TrackFileName>]

Remarks

The Volume statement specifies the default track output file name. No track file will be written unless the `-t` command line parameter is used when the program is invoked. Any track file name specified in the command line will override this track file name if specified.

The only valid volume type is ISO9660.

The Volume statement enables the Volume commands until an EndVolume statement is encountered.

Example

```
Volume ISO9660 psyqtrk.trk
```

Disc Commands

Disc commands define the highest level of disc structure. They fall into the following categories:

Defining the position of tracks within the disc.	`LeadIn`, `Track`, `LeadOut`
Disc specific definitions.	`CatalogNumber`, `MapFile`, `CDGeneratorFile`
Disc specific checks.	`CDSize`
Disc definition termination.	`EndDisc`

CatalogNumber

Function

Allows for the definition of the CD catalog number that describes the disc and appears in the Q subcode channel.

Syntax

CatalogNumber <Number>

Remarks

If no catalog number is defined then it will not be included within the disc's Q subcode channel. The catalog number can be up to 13 digits long, if it is less than this the leading digits will be padded with zeros.

Example

```
CatalogNumber 622345
```

CDGeneratorFile

Function

Specifies the name of the default Sony CDGenerator output save file.

Syntax

CDGeneratorFile <FileName>

Remarks

No file will be written unless the `-g` command line parameter is used when the program is invoked. Any CDGenerator file name specified in the command line will override this file name.

Example

```
CDGeneratorFile c:\devel\outcdg.ccs
```

CDSize

Function

Specifies the size of the target CD in minutes.

Syntax

CDSize <NumMinutes>

Default

The CDSize defaults to the standard CD length of 74 minutes.

Remarks

The value specified by this command is used to test for CD overflow. This command can be used to adjust the checked length of the CD. The input value is not checked against any specific criterion except for being a valid positive integer.

Example

```
CDSize 20
```

EndDisc

Function

Marks the end of the disc definition.

Syntax

EndDisc

Remarks

The EndDisc statement should be the last statement in any control file. Only at the end of the control file will the image be built.

Example

```
EndDisc
```

LeadIn

Function

Marks the start of a leadin track definition.

Syntax

LeadIn <TrackType>

Remarks

Valid track types are:

CDDA, CDAUDIO or AUDIO	Audio track
Mode0, Data0, ROM0 or CDROM0	Mode0 data track
Mode1, Data1, ROM1 or CDROM1	Mode1 data track
Mode2, Data2, ROM2 or CDROM2	Mode2 data track

Each disc has one leadin track that must be the first track defined. The leadin track contains the disc's table of contents (TOC) in the Q subcode channel. It is generally expected that the data portion of the track contains nothing but zeros. A minimum of length of 150 frames is allowable, although 500 is recommended as a minimum and upwards of 1500 is usual. The LeadIn statement enables track commands until an EndTrack statement is encountered.

Example

```
LeadIn Mode0
```

See also: EndTrack, Empty

LeadOut

Function

Marks the start of a leadout track definition.

Syntax

LeadOut <TrackType>

Remarks

Valid track types are:

CDDA, CDAUDIO or AUDIO	Audio track
Mode0, Data0, ROM0 or CDROM0	Mode0 data track
Mode1, Data1, ROM1 or CDROM1	Mode1 data track
Mode2, Data2, ROM2 or CDROM2	Mode2 data track

Each disc has one leadout track that must be the last track defined. The leadout track contains a disc end marker in the Q subcode channel. It is generally expected that the data portion of the track contains nothing but zeros.

A minimum of length of 150 frames is allowable, although 500 is recommended as a minimum.

The LeadOut statement enables track commands until an EndTrack statement is encountered.

Example

```
LeadOut Mode0
```

See also: EndTrack, Empty

MapFile

Function

Specifies the name of the default text map file.

Syntax

MapFile <FileName>

Remarks

No map file will be written unless the `-m` command line parameter is used when the program is invoked. Any text map file name specified in the command line will override this map file name.

Example

```
MapFile c:\devel\outmap.txt
```

Track

Function

Marks the start of a standard track definition.

Syntax

Track <TrackType>

Remarks

Valid track types are:

CDDA, CDAUDIO or AUDIO	Audio track
Mode0, Data0, ROM0 or CDROM0	Mode0 data track
Mode1, Data1, ROM1 or CDROM1	Mode1 data track
Mode2, Data2, ROM2 or CDROM2	Mode2 data track

Up to 99 standard tracks can defined on a disc.

The Track statement enables track commands until an EndTrack statement is encountered.

Example

```
Track Mode0
```

See also: EndTrack

Track Commands

Track commands define the highest level of track structure. They fall into the following categories:

Defining the component blocks of a track.	`Empty`, `Pause`, `PostGap`, `PreGap`, `Source`, `Volume`, `XASource`
Track label definitions.	`Index`, `ISRC`
Subcode source file specification.	`SubcEmpty`, `SubcSource`
Audio track specific definitions.	`Channels`, `Copy`, `PreEmphasis`
Track definition termination.	`EndTrack`

Channels

Function

Specifies the number of audio channels for an audio track.

Syntax

Channels <NumberOfChannels>

Default

The number of channels defaults to 2.

Remarks

This command is only valid within an audio track and specifies the number of audio channels that the data has been recorded for. This information is included in the Q subcode channel. Valid numbers of channels are:

2 or 4

This command is recognized but has not been implemented.

Example

```
Channels 4
```

Copy

Function

Specifies the state of the copy protection flag on an audio track.

Syntax

Copy <Boolean>

Default

The copy protection flag defaults to false.

Remarks

This command is only valid within an audio track.

Valid boolean values are:

- True, on, yes or 1
- False, off, no or 0

Example

```
Copy On
```

Empty

Function

Specifies a section of track to contain empty frames.

Syntax

Empty <NumFrames>

Remarks

This command specifies that the next <NumFrames> frames of data will be zero filled.

This command is particularly useful for defining the length of leadin and leadout tracks that technically shouldn't contain any data.

Example

```
Empty 150
```

EndTrack

Function

Marks the end of the track definition.

Syntax

EndTrack

Remarks

The EndTrack statement closes the definition of a track and returns control to the previous command level.

Example

```
EndTrack
```

Index

Function

Places an index point into the track at this point.

Syntax

Index

Remarks

Each index point increments the index count by 1. The index count is automatically set to 1 at the start of the track. The index count can have a maximum value of 99.

Indices must be separated by at least 1 frame of data.

Indices can only be defined in audio tracks that are not leadin or leadout tracks.

Example

```
Index
```

ISRC

Function

Allows for the definition of a track ISRC number to be placed in the Q subcode channel.

Syntax

ISRC <IsrcNumber>

Remarks

If no ISRC number is defined, one will not be included within the disc's Q subcode channel. The ISRC number is 12 digits long, made up of the following fields:

Country code	2 digits or uppercase letters
Owner code	3 digits or uppercase letters
Recording Year	2 digits
Serial Numbe	5 digits

An ISRC cannot be defined in leadin or leadout tracks.

Example

```
ISRC US2PD9443001
```

Pause

Function

Specifies a Pause section for the track.

Syntax

Pause <NumFrames>

Remarks

This command specifies that the track will be preceded by a pause of <NumFrames> frames of data. Pause frames are always zero filled and will always be placed at the start of the track.

A Pause cannot be defined in leadin or leadout tracks.

A Pause must always be defined in the first standard track of the disc and in any track where the preceding track is of a different data type.

A Pause must be a minimum of 150 frames in length, and it is recommended that Pauses are greater than 1000 frames.

Example

```
Pause 1500
```

PostGap

Function

Specifies a PostGap section for the track.

Syntax

PostGap <NumFrames>

Remarks

This command specifies that the track will be followed by a gap of <NumFrames> frames of data. PostGap frames are always zero filled and will always be placed at the end of the track.

A PostGap cannot be defined in audio tracks.

A PostGap must be a minimum of 150 frames in length.

Example

```
PostGap 150
```

PreEmphasis

Function

Specifies the state of the pre-emphasis flag on an audio track.

Syntax

PreEmphasis <Boolean>

Default

The pre-emphasis flag defaults to false.

Remarks

This command is only valid within an audio track.

Valid boolean values are:

- True, on, yes or 1
- False, off, no or 0

Example

```
PreEmphasis On
```

PreGap

Function

Specifies a PreGap section for the track.

Syntax

PreGap <NumFrames>

Remarks

This command specifies that the track will be started by a gap of <NumFrames> frames of data. PreGap frames are always zero filled and will always be placed at the beginning of the track.

A PreGap cannot be defined in leadin, leadout or audio tracks.

A PreGap must be a minimum of 150 frames in length.

Example

```
PreGap 150
```

Source

Function

Specifies that the data from the specified file should be placed in the disc image at this point.

Syntax

Source <FileName>

Remarks

Any incomplete frame of data at the end of the file will be padded with zeros.

There are 3 data types for which a Source command is allowable. These are:

Audio	Each frame of data is 2352 bytes long
Mode1	Each frame of data is 2048 bytes long
Mode2	Each frame of data is 2336 bytes long

This command is not allowed in an XA track at this level.

Example

```
Source c:\develop\data\psyqace.bin
```

See also: XASource

Volume

Function

Marks the start of the definition of a track volume and defines a primary output file for the volume.

Syntax

Volume <VolumeType> [<TrackFileName>]

Remarks

No track file will be written unless the `-t` command line parameter is used when the program is invoked and the track file name has been specified within this definition. No track file name can be specified in the command line to override this track file name.

The only valid volume type is ISO9660.

The Volume statement enables the Volume commands until an EndVolume statement is encountered.

Example

```
Volume ISO9660
```

XASource

Function

Specifies that the data from the specified file should be placed in the disc image at this point and treated as XA data.

Syntax

XASource <FileName>

Remarks

The contents of the file are 2336 byte fixed-length records of any combination of either Form1 or Form2 records. The records consist of the following fields:

Form1	Submode+Data+ECC+EDC
Form2	Submode+Data+EDC

This command is only valid in an XA track.

Example

```
XASource c:\develop\data\psyqace.mxa
```

See also: Source

Volume Commands

Volume commands define the highest level of ISO9660 track volume structure. They fall into the following categories:

Volume descriptor definitions.	`PrimaryVolume`
System area definition.	`SystemArea`
Volume definition termination.	`EndVolume`

EndVolume

Function

Marks the end of the volume definition.

Syntax

EndVolume

Remarks

The EndVolume statement closes the definition of a volume and returns control to the previous command level.

Example

```
EndVolume
```

PrimaryVolume

Function

Marks the start of a primary volume definition.

Syntax

PrimaryVolume

Remarks

An ISO volume must contain one and only primary volume definition.

The PrimaryVolume statement enables the Primary Volume commands until an EndPrimaryVolume statement is encountered.

Example

```
PrimaryVolume
```

See also: EndPrimaryVolume

SystemArea

Function

Specifies that the data from the specified file should be placed in the ISO9660 system area.

Syntax

SystemArea <FileName>

Default

This area will be zero filled if no input file is specified.

Remarks

This command indicates that contents of the specified file should be placed into the ISO9660 system area. The system area is the first 16 logical sectors of the volume. If the specified file is too long then its contents will be truncated. If it is too short, then it will be zero padded.

The ISO standard does not specify what data should be placed in this area. It is up to system designers to choose what they wish to be placed here.

Example

```
SystemArea c:\psyq\boot.bin
```

PRIMARY Volume Commands

PrimaryVolume commands define the structure of the content of an ISO9660 Primary Volume descriptor. They fall into the following categories:

Field value definitions.	`AbstractFileIdentifier`, `ApplicationIdentifier`, `ApplicationUse`, `BibliographicFileIdentifier`, `CopyrightFileIdentifier`, `DataPreparerIdentifier`, `LogicalBlockSize`, `PublisherIdentifier`, `SystemIdentifier`, `VolumeCreationDate`, `VolumeEffectiveDate`, `VolumeExpirationDate`, `VolumeIdentifier`, `VolumeModificationDate`, `VolumeSetIdentifier`
XA specific field value definitions.	`XAStartDirectory`
Volume level definitions.	`DescriptorWrites`
Volume contents definitions.	`Hierarchy`, `Lpath`, `MPath`, `OptionalLPath`, `OptionalMPath`
Volume definition termination.	`EndPrimaryVolume`

AbstractFileIdentifier

Function

Fills the abstract file identifier field with the specified root file name.

Syntax

AbstractFileIdentifier <CdRootFileName>

Default

If no command is encountered, then the field will be blank filled.

Remarks

This file name must be specified in the root directory of the volume's hierarchy definition.

The file name must conform to ISO level 1 standards. That is, it must have no more than 8 file name characters and 3 file extension characters.

Example

```
AbstractFileIdentifier ABSTRACT.DOC
```

ApplicationIdentifier

Function

Fills the application identifier field with the specified string.

Syntax

ApplicationIdentifier <String>

Default

If no command is encountered, then the field will be blank filled.

Remarks

The field holds a maximum of 128 characters. If the string's first character is an underscore ('_' - 0x5F), the remainder of the string is taken to be the name of a file in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level 1 standards. That is, it must have no more than 8 file name characters and 3 file extension characters.

The string, if not a file name, is made up of up to 128 'a' characters. For a list of valid 'a' characters see *Appendix B, ISO Character Sets*.

Example

```
ApplicationIdentifier _APP_ID.DOC
ApplicationIdentifier PSYQ APPLET
```

ApplicationUse

Function

Fills the application use field with the contents of the specified file.

Syntax

ApplicationUse <FileName>

Default

If no command is encountered, then the field will be zero filled.

Remarks

This statement specifies the name of a file, the contents of which will be loaded into the application use field of the volume descriptor. The field is 512 bytes long. If the input file is too long, then the contents will be truncated. If the file is too short, then the field will be padded with zeros.

Example

```
ApplicationUse c:\devel\psyq\appuse.bin
```

BibliographicFileIdentifier

Function

Fills the bibliographic file identifier field with the specified root file name.

Syntax

BibliographicFileIdentifier <CdRootFileName>

Default

If no command is encountered, then the field will be blank filled.

Remarks

The file name must be specified in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level 1 standards. That is, it must have no more than 8 file name characters and 3 file extension characters.

Example

```
BibliographicFileIdentifier BIB.DOC
```

CopyrightFileIdentifier

Function

Fills the copyright file identifier field with the specified root file name.

Syntax

CopyrightFileIdentifier <CdRootFileName>

Default

If no command is encountered, then the field will be blank filled.

Remarks

The file name must be specified in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level 1 standards. That is, it must have no more than 8 file name characters and 3 file extension characters.

Example

```
CopyrightFileIdentifier COPYR.DOC
```

DataPreparerIdentifier

Function

Fills the data preparer identifier field with the specified string.

Syntax

DataPreparerIdentifier <String>

Default

If no command is encountered, then the field will be blank filled.

Remarks

The field holds a maximum of 128 characters. If the string's first character is an underscore ('_' - 0x5F), the remainder of the string is taken to be the name of a file in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level 1 standards. That is, it must have no more than 8 file name characters and 3 file extension characters.

The string, if not a file name, is made up of up to 128 'a' characters. For a list of valid 'a' characters see *Appendix B, ISO Character Sets*.

Example

```
DataPreparerIdentifier _DATID.DOC
DataPreparerIdentifier SNSYSTEMS
```

DescriptorWrites

Function

Specifies the number of times that the volume descriptor is to be written to the output.

Syntax

DescriptorWrites <Number>

Default

If no command is encountered, then the descriptor will be written once.

Remarks

The ISO standard allows for the descriptor to be written out as many times as required. Each descriptor record takes up one sector of disc space.

PSX Restriction: With PSX, volume descriptors can only be written once (i.e., DescriptorWrites can only be set to a value of 1).

Example

```
DescriptorWrites 1
```

EndPrimaryVolume

Function

Marks the end of the primary volume definition.

Syntax

EndPrimaryVolume

Remarks

The EndPrimaryVolume statement closes the definition of a primary volume and returns control to the previous command level.

Example

```
EndPrimaryVolume
```

Hierarchy

Function

Marks the start of a directory hierarchy.

Syntax

Hierarchy

Remarks

The must be one and only one hierarchy definition within each primary volume definition.

The Hierarchy statement enables the hierarchy commands until an EndHierarchy statement is encountered.

Example

```
Hierarchy
```

See also: EndHierarchy

LogicalBlockSize

Function

Specifies the logical block size of the volume descriptor.

Syntax

LogicalBlockSize <BlockSize>

Default

The logical block size defaults to 2048.

Remarks

Valid block size values are 512, 1024 and 2048.

This command Is recognized but has not been implemented.

Example

```
LogicalBlockSize    2048
```

LPath

Function

Specifies the position of an L type path table.

Syntax

LPath

Remarks

A path table is a directory quick reference to allow an executing program to find a particular directory record without having to recurse all the way down the directory structure.

There are 2 types of path tables; L and M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

There must be one and only one LPath definition within each primary volume definition. Within a PSX disc definition, the LPath will be automatically placed correctly.

Example

```
Lpath
```

See also: MPath

MPath

Function

Specifies the position of an M type path table.

Syntax

MPath

Remarks

A path table is a directory quick reference to allow an executing program to find a particular directory record without having to recurse all the way down the directory structure.

There are 2 types of path tables; L and M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

There must be one and only one MPath definition within each primary volume definition. Within a PSX disc definition, the MPath will be automatically placed correctly.

Example

```
Mpath
```

See also: Lpath

OptionalLPath

Function

Specifies the position of an optional additional L type path table.

Syntax

OptionalLPath

Default

No additional L path table will be written into the image if this command is not encountered.

Remarks

The additional path table is identical in all respects to the original. A path table is a directory quick reference to allow an executing program to find a particular directory record without having to recurse all the way down the directory structure.

There are 2 types of path tables; L and M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

Ther can be at most only one OptionalLPath definition within each primary volume definition. Within a PSX disc definition, the OptionalLPath will be automatically placed correctly.

Example

```
OptionalLpath
```

See also: LPath, OptionalMPath

OptionalMPath

Function

Specifies the position of an optional additional M type path table.

Syntax

OptionalMPath

Default

No additional M path table will be written into the image if this command is not encountered.

Remarks

The additional path table is identical in all respects to the original. A path table is a directory quick reference to allow an executing program to find a particular directory record without having to recurse all the way down the directory structure.

There are 2 types of path tables; L and M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

There can be at most only one OptionalMPath definition within each primary volume definition. Within a PSX disc definition the OptionalMPath will be automatically placed correctly.

Example

```
OptionalMpath
```

See also: MPath, OptionalLPath

PublisherIdentifier

Function

Fills the publisher identifier field with the specified string.

Syntax

PublisherIdentifier <String>

Default

If no command is encountered, then the field will be blank filled.

Remarks

The field holds a maximum of 128 characters. If the string's first character is an underscore ('_' - 0x5F), the remainder of the string is taken to be the name of a file in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level 1 standards. That is, it must have no more than 8 file name characters and 3 file extension characters.

The string, if not a file name, is made up of up to 128 'a' characters. For a list of valid 'a' characters see *Appendix B, ISO Character Sets*.

Example

```
PublisherIdentifier _PUBID.DOC
PublisherIdentifier SNSYSTEMS
```

SystemIdentifier

Function

Fills the system identifier field with the specified system name string.

Syntax

SystemIdentifier <SystemNameString>

Remarks

This field specifies the system identifiers that can act upon the contents of the System Area contents.

The must be one and only one SystemIdentifier definition within each primary volume definition.

The string is made up of up to 32 'a' characters. For a list of valid 'a' characters see *Appendix B, ISO Character Sets*.

For a PSX disc, the system identifier must be defined as PLAYSTATION.

Example

```
SystemIdentifier DOS
```

VolumeCreationDate

Function

Fills the volume creation field with the specified date and time.

Syntax

VolumeCreationDate <Date> <Time> [<Offset>]

Default

If no creation date and time are given, then the date and time of creation of the volume descriptor will be used with the default Greenwich offset.

Remarks

This statement specifies the date, time and specific Greenwich offset of the volume descriptor creation. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command. If no offset is specified then the default Greenwich offset will be used.

Example

```
VolumeCreationDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

VolumeEffectiveDate

Function

Fills the volume effective field with the specified date and time.

Syntax

VolumeEffectiveDate <Date> <Time> [<Offset>]

Default

If no effective date and time are given, then the field in the volume descriptor will be set to:
00/00/0000 00:00:00:00 0

Remarks

This statement specifies the date, time and specific Greenwich offset of the volume descriptor becoming effective. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command. If no offset is specified then the default Greenwich offset will be used.

Example

```
VolumeEffectiveDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

VolumeExpirationDate

Function

Fills the volume expiration field with the specified date and time.

Syntax

VolumeExpirationDate <Date> <Time> [<Offset>]

Default

If no expiry date and time are given, then the field in the volume descriptor will be set to:
00/00/0000 00:00:00:00 0

Remarks

This statement specifies the date, time and specific Greenwich offset of the volume descriptor expiring. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command. If no offset is specified then the default Greenwich offset will be used.

Example

```
VolumeExpirationDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

VolumeIdentifier

Function

Fills the volume identifier field with the specified string.

Syntax

VolumeIdentifier <String>

Remarks

The must be one and only one VolumeIdentifier definition within each primary volume definition.

The string is made up of up to 32 ‘d’ characters. For a list of valid ‘d’ characters see *Appendix B, ISO Character Sets*.

Example

```
VolumeIdentifier MEGAGAME1
```

VolumeModificationDate

Function

Fills the volume modification field with the specified date and time.

Syntax

VolumeModificationDate <Date> <Time> [<Offset>]

Default

If no modification date and time are given, then the date and time of creation of the volume descriptor will be used with the default Greenwich offset.

Remarks

This statement specifies the date, time and specific Greenwich offset of the last volume descriptor modification. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command. If no offset is specified then the default Greenwich offset will be used.

Example

```
VolumeModificationDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

VolumeSetIdentifier

Function

Fills the volume set identifier field with the specified string.

Syntax

VolumeSetIdentifier <String>

Default

If no command is encountered, then the field will be blank filled.

Remarks

The string is made up of up to 128 ‘d’ characters. For a list of valid ‘d’ characters see *Appendix B, ISO Character Sets*.

Example

```
VolumeSetIdentifier MEGAGAME1
```

XAStartDirectory

Function

Fills the start directory field with the specified CD directory name.

Syntax

XAStartDirectory <String>

Default

If no command is encountered, then the field will be zero filled.

Remarks

The string is made up of up to 8 ‘d’ characters and must be a valid subdirectory of the CD root directory hierarchy.

This statement is only valid within an XA track.

For a list of valid ‘d’ characters see *Appendix B, ISO Character Sets*.

Example

```
XAStartDirectory STARTUP
```

Directory Hierarchy Commands

Directory hierarchy commands define the structure of the root directory of an ISO9660 Primary or Supplementary volume. The commands fall into the following categories:

Directory space allocation definitions.	`AddLength`, `MinLength`, `Gap`
Directory attribute definitions.	`Attributes`, `RecordingDate`
XA specific directory attribute definitions.	`XAAudioAttributes`, `XAFileAttributes`, `XAFilePermissions`, `XAOwnerGroup`, `XAOwnerUser`, `XAVideoAttributes`
Directory object definitions.	`Directory`, `File`, `SourceDirectory`, `XAInterleavedFile`
Volume definition termination.	`EndHierarchy`

AddLength

Function

Specifies the number of bytes to be added to the length of the generated directory record.

Syntax

AddLength <Length>

Default

If no command is encountered, then AddLength defaults to zero.

Remarks

This statement specifies a number of zero filled bytes to be added onto the end of the directory structure. This directory record may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore a directory of say 10 files will produce a directory structure of about 400 bytes. An AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final directory structure will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
AddLength 512
```

See also: MinLength

Attributes

Function

Specifies the directory attribute flags.

Syntax

Attributes <Attribute>

Default

If no command is encountered, then the Attributes default to NotHidden.

Remarks

The valid attributes are:

- Hidden
- NotHidden

If the Hidden attribute is defined, then the directory will not be made known to the user.

Example

```
Attributes Hidden
```

Directory

Function

Marks the start of a directory definition.

Syntax

Directory <DirectoryName>

Remarks

The Directory statement specifies the start of a directory definition with the name specified.

Example

```
Directory PSYQSRC
```

See also: EndDirectory

EndHierarchy

Function

Marks the end of a hierarchy definition.

Syntax

EndHierarchy

Remarks

The EndHierarchy statement closes the definition of a hierarchy record and returns control to the previous command level.

Example

```
EndHierarchy
```

File

Function

Marks the start of a file definition.

Syntax

File <ISOFileName> [<Version>]

Default

If the version number is not specified, then it defaults to 1.

Remarks

The File statement specifies the start of a ISO file definition with the name specified.

The File statement enables the File commands until an EndFile statement is encountered.

For an XA disc, the version must be 1.

Example

```
File MONSTERS.GPH 2
```

See also: EndFile

Gap

Function

Places empty sectors within the hierarchy definition.

Syntax

Gap <Num Empty Sectors>

Remarks

The Gap statement places an empty area within the hierarchy definition of length NumEmptySectors. This feature can be used to position files explicitly within the image.

Example

```
Gap 20
```

MinLength

Function

Specifies the minimum number of bytes of the directory record.

Syntax

MinLength <Length>

Default

If no command is encountered, then MinLength defaults to zero.

Remarks

This statement specifies a minimum size of directory structure that the actual record will be blank filled to.

This directory record may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore a directory of say 10 files will produce a directory structure of about 400 bytes. An AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final directory structure will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
MinLength 512
```

See also: AddLength

RecordingDate

Function

Logs the recording date of the directory with the specified date and time.

Syntax

RecordingDate <Date> <Time> [<Offset>]

Default

If no recording date and time are given, then the date and time of creation of the directory record will be used.

If no offset is specified then the default Greenwich offset will be used.

Remarks

This statement specifies the date, time and specific Greenwich offset of the directory record. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command.

Example

```
RecordingDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

SourceDirectory

Function

Searches the specified directory, and files within it, to recreate the structure and contents in the image.

Syntax

SourceDirectory <DOSDirectory> [<Att1>] [<Att2>] [...]

Default

The command has the following defaults:

- Subroutines will not be recursed.
- All files will be included.
- No additional spare space will be generated.

Remarks

This statement can be used to copy whole directory hierarchies directly into the present directory.

If the keyword 'SubDirectories' appears as one of the attributes, then all subdirectories of the named DOS directory will be searched recursively, copying all the valid filenames to the image.

If the keyword 'IncludeWilds' appears, then the command will *only* retrieve files from the directory and sub-directories complying to the format(s) of those attributes that are wildcard definitions.

If the keyword 'ExcludeWilds' appears, then the command will *only* retrieve files from the directory and sub-directories *not* complying to the format(s) of those attributes that are wildcard definitions.

If the keyword 'SpareSpace' appears, then the command will add in extra zeros at the end of the files' contents. The attribute must be followed by an equals sign and then a number specifying a percentage of additional space to be added. This value can be from 0 to 400. The % sign is optional.

Example

```
SourceDirectory c:\tmp SubDirectories
SourceDirectory c:\tmp ExcludeWilds *.prj *.td
SourceDirectory c:\tmp *.c SpareSpace=25%
```

XAAudioAttributes

Function

Specifies the default audio attribute flags.

Syntax

XAAudioAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the Attributes default to Emphasis Off, ADPCM Level B, Mono.

Remarks

The valid attributes are:

- Emphasis_On and Emphasis_Off
- ADPCM_B and ADPCM_C
- Mono and Stereo

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAAudioAttributes Emphasis_On Stereo
```

XAFileAttributes

Function

Specifies the default file attribute flags.

Syntax

XAFileAttributes <Attribute> [<Attribute>]

Remarks

The valid attributes are:

- Form1 or 1 and Form2 or 2
- Audio, Video and Data

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAFileAttributes Form1 Data
```

XAFilePermissions

Function

Specifies the default file permission flags.

Syntax

XAFilePermissions <Attribute> [<Attribute>] [...]

Remarks

The valid attributes are:

- Owner_Read
- Owner_Execute
- Group_Read
- Group_Execute
- World_Read
- World_Execute

Any combination of attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAFilePermissions Owner_Read
```

XAInterleavedFile

Function

Marks the start of an XA interleaved file definition.

Syntax

XAInterleavedFile <ISOFileName> [<Filename>]

Remarks

The File statement specifies the start of the definition of an XA file with interleaved channels.

The XAInterleavedFile statement enables the Interleaved File commands until an XAEndInterleaveFile statement is encountered.

Example

```
XAInterleavedFile MONSTERS.GPH c:\games\output.str
```

See also: XAEndInterleavedFile

XAOwnerGroup

Function

Specifies the default file Owner Group Identifier.

Syntax

XAOwnerGroup <GroupIdentifier>

Default

The owner group defaults to zero.

Remarks

The owner group identifier is a number between 0 and 65535.

This statement is only valid within an XA track.

Example

```
XAOwnerGroup 27
```

XAOwnerUser

Function

Specifies the default file Owner User Identifier.

Syntax

XAOwnerUser <UserIdentifier>

Default

The owner user defaults to zero.

Remarks

The owner user identifier is a number between 0 and 65535.

This statement is only valid within an XA track.

Example

```
XAOwnerUser 15
```

XAVideoAttributes

Function

Specifies the default video attribute flags.

Syntax

XAVideoAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the Attributes default to Application Specific.

Remarks

The valid attributes are:

- ApplicationSpecific

or a combination of

- 640x480 and 320x240
- Clut1, Clut2, Clut4 and Clut8

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAVideoAttributes ApplicationSpecific
XAVideoAttributes 640x480 Clut2
```

Directory Commands

Directory commands define the structure of a sub-directory of an ISO9660 Primary or Supplementary volume. The commands fall into the following categories:

Directory space allocation definitions.	`AddLength`, `MinLength`
Directory attribute definitions.	`Attributes`, `RecordingDate`
XA specific directory attribute definitions.	`XAAudioAttributes`, `XAFileAttributes`, `XAFilePermissions`, `XAOwnerGroup`, `XAOwnerUser`, `XAVideoAttributes`
Directory object definitions.	`Directory`, `File`, `SourceDirectory`, `XAInterleavedFile`
Command definition termination.	`EndDirectory`

AddLength

Function

Specifies the number of bytes to be added to the length of the generated directory record.

Syntax

AddLength <Length>

Default

If no command is encountered, then AddLength defaults to zero.

Remarks

This statement specifies a number of zero filled bytes to be added onto the end of the directory structure. This directory record may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore a directory of say 10 files will produce a directory structure of about 400 bytes. An AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final directory structure will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
AddLength 512
```

See also: MinLength

Attributes

Function

Specifies the directory attribute flags.

Syntax

Attributes <Attribute>

Default

If no command is encountered, then the Attributes default to NotHidden.

Remarks

The valid attributes are:

- Hidden
- NotHidden

If the Hidden attribute is defined then the directory will not be made known to the user.

Example

```
Attributes Hidden
```

Directory

Function

Marks the start of a directory definition.

Syntax

Directory <DirectoryName>

Remarks

The Directory statement specifies the start of a directory definition with the name specified.

The Directory statement enables the Directory commands an EndDirectory statement is encountered.

Example

```
Directory PSYQSRC
```

See also: EndDirectory

EndDirectory

Function

Marks the end of a directory definition.

Syntax

EndDirectory

Remarks

The EndDirectory statement closes the definition of a directory record and returns control to the previous command level.

Example

```
EndDirectory
```

File

Function

Marks the start of a file definition.

Syntax

File <ISOFileName> [<Version>]

Default

If the version number is not specified, then it defaults to 1.

Remarks

The File statement enables the File commands until an EndFile statement is encountered.

For an XA disc, the version must be 1.

Example

```
File MONSTERS.GPH 2
```

See also: EndFile

Gap

Function

Places empty sectors within the hierarchy definition.

Syntax

Gap <Num Empty Sectors>

Remarks

The Gap statement places an empty area within the hierarchy definition of length NumEmptySectors. This feature can be used to position files explicitly within the image.

Example

```
Gap 20
```

MinLength

Function

Specifies the minimum number of bytes of the directory record.

Syntax

MinLength <Length>

Default

If no command is encountered, then MinLength defaults to zero.

Remarks

This statement specifies a minimum size of directory structure that the actual record will be blank filled to.

This directory record may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore a directory of say 10 files will produce a directory structure of about 400 bytes. An AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final directory structure will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
MinLength 512
```

See also: AddLength

RecordingDate

Function

Logs the recording date of the directory with the specified date and time.

Syntax

RecordingDate <Date> <Time> [<Offset>]

Default

If no recording date and time are given, then the date and time of creation of the directory record will be used.

If no offset is specified, then the default Greenwich offset will be used.

Remarks

This statement specifies the date, time and specific Greenwich offset of the directory record. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command.

Example

```
RecordingDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

SourceDirectory

Function

Searches the specified directory and files within it to recreate the structure and contents in the image.

Syntax

SourceDirectory <DOSDirectory> [<Att1>] [<Att2>] [...]

Default

The command has the following defaults:

- Subroutines will not be recursed.
- All files will be included.
- No additional spare space will be generated.

Remarks

This statement can be used to copy whole directory hierarchies directly into the present directory.

If the keyword 'SubDirectories' appears as one of the attributes, then all subdirectories of the named DOS directory will be searched recursively, copying all the valid filenames to the image.

If the keyword 'IncludeWilds' appears, then the command will *only* retrieve files from the directory and sub-directories complying to the format(s) of those attributes that are wildcard definitions.

If the keyword 'ExcludeWilds' appears, then the command will *only* retrieve files from the directory and sub-directories *not* complying to the format(s) of those attributes that are wildcard definitions.

If the keyword 'SpareSpace' appears, then the command will add in extra zeros at the end of the files' contents. The attribute must be followed by an equals sign and then a number specifying a percentage of additional space to be added. This value can be from 0 to 400. The % sign is optional.

Example

```
SourceDirectory c:\tmp SubDirectories
SourceDirectory c:\tmp ExcludeWilds *.prj *.td|
SourceDirectory c:\tmp *.c SpareSpace=25%
```

XAAudioAttributes

Function

Specifies the default audio attribute flags.

Syntax

XAAudioAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the attributes default to the parent directory default.

Remarks

The valid attributes are:

- Emphasis_On and Emphasis_Off
- ADPCM_B and ADPCM_C
- Mono and Stereo

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAAudioAttributes Emphasis_On Stereo
```

XAFileAttributes

Function

Specifies the default file attribute flags.

Syntax

XAFileAttributes <Attribute> [<Attribute>]

Default

If no command is encountered, then the attributes default to the parent directory default.

Remarks

The valid attributes are:

- Form1 or 1 and Form2 or 2
- Audio, Video and Data

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAFileAttributes Form1 Data
```

XAFilePermissions

Function

Specifies the default file permission flags.

Syntax

XAFilePermissions <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the attributes default to the parent directory default.

Remarks

The valid attributes are:

- Owner_Read
- Owner_Execute
- Group_Read
- Group_Execute
- World_Read
- World_Execute

Any combination of attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAFilePermissions Owner_Read
```

XAInterleavedFile

Function

Marks the start of an XA interleaved file definition.

Syntax

XAInterleavedFile <ISOFileName> [<Interleaved Output File>]

Remarks

The File statement specifies the start of the definition of an XA file with interleaved channels.

The XAInterleavedFile statement enables the Interleaved File commands until an XAEndInterleaveFile statement is encountered.

The Interleaved Output File will be output only if the `-g` is specified on the command line. The contents of each of the interleaved channels are amalgamated into a single interleaved output file with all parity and CRC values set to zero. This file output can then be used as input to an XAF source command.

Example

```
XAInterleavedFile MONSTERS.GPH
```

See also: XAEndInterleavedFile

XAOwnerGroup

Function

Specifies the default file Owner Group Identifier.

Syntax

XAOwnerGroup <GroupIdentifier>

Default

If no command is encountered, then the owner group defaults to the parent directory default.

Remarks

The owner group identifier is a number between 0 and 65535.

This statement is only valid within an XA track.

Example

```
XAOwnerGroup 27
```

XAOwnerUser

Function

Specifies the default file Owner User Identifier.

Syntax

XAOwnerUser <UserIdentifier>

Default

If no command is encountered, then the user identifier defaults to the parent directory default.

Remarks

The owner user identifier is a number between 0 and 65535.

This statement is only valid within an XA track.

Example

```
XAOwnerUser 15
```

XAVideoAttributes

Function

Specifies the default video attribute flags.

Syntax

XAVideoAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the attributes default to the parent directory default.

Remarks

The valid attributes are:

- ApplicationSpecific

or a combination of

- 640x480 and 320x240
- Clut1, Clut2, Clut4 and Clut8

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAVideoAttributes ApplicationSpecific
XAVideoAttributes 640x480 Clut2
```

File Commands

File commands define the contents of an ISO9660 File. The commands fall into the following categories:

File space allocation definitions.	`AddLength`, `MinLength`
File attribute definitions.	`Attributes`, `RecordingDate`
XA file attribute definitions.	`XAAudioAttributes`, `XAEndOfRecord`, `XAFileAttributes`, `XAFilePermissions`, `XAOwnerGroup`, `XAOwnerUser`, `XATrigger`, `XAVideoAttributes`
File object definitions.	`Source`, `XASource`
Command definition termination.	`EndFile`

AddLength

Function

Specifies the number of bytes to be added to the length of the file contents.

Syntax

AddLength <Length>

Default

If no command is encountered, then AddLength defaults to zero.

Remarks

This statement specifies a number of zero filled bytes to be added onto the end of the file contents. This file may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore if we have a file of say 400 bytes, an AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final record will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
AddLength 512
```

See also: MinLength

Attributes

Function

Specifies the file attribute flags.

Syntax

Attributes <Attribute> [<Attribute>]

Default

If no command is encountered, then the attributes default to NotHidden.

Remarks

The valid attributes are:

- Hidden or NotHidden
- Record or NotRecord

If the Hidden attribute is defined, then the file will not be made known to the user.

If the Record attribute is defined, then the file will be flagged as a record.

Example

```
Attributes Hidden NotRecord
```

EndFile

Function

Marks the end of a file definition.

Syntax

EndFile

Remarks

The EndFile statement closes the definition of a file record and returns control to the previous command level.

Example

```
EndFile
```

MinLength

Function

Specifies the minimum number of bytes of a file record.

Syntax

MinLength <Length>

Default

If no command is encountered, then MinLength defaults to zero.

Remarks

This statement specifies a minimum size of file record. If the file is smaller than this, then it will be blank filled to the specified size.

The file may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore if we have a file of say 400 bytes, an AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final record will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
MinLength 512
```

See also: AddLength

RecordingDate

Function

Logs the recording date of the file with the specified date and time.

Syntax

RecordingDate <Date> <Time> [<Offset>]

Default

If no recording date and time are given, then the DOS date and time of creation of the source file will be used.

If no offset is specified then the default Greenwich offset will be used.

Remarks

This statement specifies the date, time and specific Greenwich offset of the file. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command.

Example

```
RecordingDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

Source

Function

Specifies that the contents of the file are to be copied from the file specified.

Syntax

Source <FileName>

Remarks

This command indicates that contents of the specified file should be used as the contents of the ISO file. Any incomplete frame of data at the end of the file will be padded with zeros.

Example

```
Source c:\develop\data\psyqace.bin
```

XAAudioAttributes

Function

Specifies the file's audio attribute flags.

Syntax

XAAudioAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the attributes default to the parent directory default.

Remarks

The valid attributes are:

- Emphasis_On and Emphasis_Off
- ADPCM_B and ADPCM_C
- Mono and Stereo

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track for a file defined as being Audio.

Example

```
XAAudioAttributes Emphasis_On Stereo
```

XAEndOfRecord

Function

Specifies the positions of End Of Record flags within the file.

Syntax

XAEndOfRecord <EOROffset> [<EOROffset>] [...] [EndOfFile]

Remarks

This command can be called more than once for a file.

The EndOfRecord Offset is the number of frames into the file that the end-of-record marker is to be set at.

The EndOfFile parameter allows the End Of Record flag to be placed at the end of the file (*after* adjustment by AddLength and MinLength!) without precalculation of position.

This statement is only valid within an XA track.

Example

```
XAEndOfRecord 15 30 45 90
```

XAFileAttributes

Function

Specifies the file attribute flags.

Syntax

XAFileAttributes <Attribute> [<Attribute>]

Remarks

The valid attributes are:

- Form1 or 1 and Form2 or 2
- Audio, Video and Data

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAFileAttributes Form1 Data
```

XAFilePermissions

Function

Specifies the default file permission flags.

Syntax

XAFilePermissions <Attribute> [<Attribute>] [...]

Remarks

The valid attributes are:

- Owner_Read
- Owner_Execute
- Group_Read
- Group_Execute
- World_Read
- World_Execute

Any combination of attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAFilePermissions Owner_Read
```

XAOwnerGroup

Function

Specifies the default file Owner Group Identifier.

Syntax

XAOwnerGroup <GroupIdentifier>

Default

The owner group defaults to zero.

Remarks

The owner group identifier is a number between 0 and 65535.

This statement is only valid within an XA track.

Example

```
XAOwnerGroup 27
```

XAOwnerUser

Function

Specifies the default file Owner User Identifier.

Syntax

XAOwnerUser <UserIdentifier>

Default

The owner user defaults to zero.

Remarks

The owner user identifier is a number between 0 and 65535.

This statement is only valid within an XA track.

Example

```
XAOwnerUser 15
```

XASource

Function

Specifies that the data from the specified file is to be copied from the specified file.

Syntax

XASource <FileName>

Remarks

This command indicates that contents of the specified file should be used as the contents of the CD file within the XA track.

The contents of the file are 2336-byte fixed length records of any combination of either Form1 or Form2 records. The records consist of the following fields:

Form1	Submode+Data+ECC+EDC
Form2	Submode+Data+EDC

Because the input files already have all the submode data set up within them this command cannot appear within a File specification with the following XA commands:

- XAAudioAttributes
- XAEndOfRecord
- XAFileAttributes
- XATrigger
- XAVideoAttributes

This command is only valid in an XA track.

Example

```
XASource c:\develop\data\psyqace.mxa
```

See also: Source

XATrigger

Function

Specifies the positions of trigger flags within the file.

Syntax

XATrigger <TrigOffset> [<TrigOffset>] [...] [EndOfFile]

Remarks

This command can be called more than once for a file.

The Trigger Offset is the number of frames into the file that the trigger marker is to be set at.

The EndOfFile parameter allows a trigger to be placed at the end of the file (*after* adjustment by AddLength and MinLength!) without precalculation of position.

This statement is only valid within an XA track.

Example

```
XATrigger 15 30 45 90
```

XAVideoAttributes

Function

Specifies the default video attribute flags.

Syntax

XAVideoAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the Attributes default to Application Specific.

Remarks

The valid attributes are:

- ApplicationSpecific

or a combination of

- 640x480 and 320x240
- Clut1, Clut2, Clut4 and Clut8

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track.

Example

```
XAVideoAttributes ApplicationSpecific
XAVideoAttributes 640x480 Clut2
```

XA Interleaved File Commands

XA Interleaved File commands define the contents of an XA File with interleaved channels. The commands fall into the following categories:

File attribute definitions	`Attributes`, `RecordingDate`
XA file attribute definitions	`XAFilePermissions`, `XAOwnerGroup`, `XAOwnerUser`
File object definitions	`XAChannel`
Interleave definitions	`XAChannelInterleave`
Command definition termination	`XAEndInterleavedFile`

Attributes

Function

Specifies the file attribute flags.

Syntax

Attributes <Attribute> [<Attribute>]

Default

If no command is encountered, then the Attributes default to NotHidden.

Remarks

The valid attributes are:

- Hidden or NotHidden
- Record or NotRecord

If the Hidden attribute is defined, then the file will not be made known to the user.

If the Record attribute is defined, then the file will be flagged as a record.

Example

```
Attributes Hidden NotRecord
```

RecordingDate

Function

Logs the recording date of the file with the specified date and time.

Syntax

RecordingDate <Date> <Time> [<Offset>]

Default

If no recording date and time are given, then the DOS date and time of creation of the source file will be used.

If no offset is specified then the default Greenwich offset will be used.

Remarks

This statement specifies the date, time and specific greenwich offset of the file. Any valid date and time can be input.

Dates must be specified in the following format:
MM/DD/YYYY

Times must be specified in the following format:
HH:MM:SS:hh

The specific Greenwich offset must be specified in exactly the same format as for the GreenwichOffset command.

Example

```
RecordingDate 01/26/1963 06:45:00:00 0
```

See also: GreenwichOffset

XAChannel

Function

Marks the start of an XA channel definition.

Syntax

XAChannel <ChannelNumber>

Remarks

The XAChannel statement specifies the start of the definition of an XA channel within an XA Interleaved file.

The Channel Number must be a number between 1 and 32 inclusive. Only one channel definition for each channel number is allowed.

The XAChannel statement enables the Channel commands until an XAEndChannel statement is encountered.

Example

```
XAChannel 3
```

See also: XAEndChannel

XAChannelInterleave

Function

Defines the format of the XA channel interleave.

Syntax

XAChannelInterleave <l'type> [<arg>] [<l'type> [<arg>]]

Remarks

The XAChannelInterleave statement specifies how the channels within the XA file are to be interleaved.

The valid interleave types are:

- TimeCritical
- Explicit
- Proportional
- Even
- PaddedEven

TimeCritical and Explicit

These two interleave types are for the most part the same, as they must be both followed by an argument string that defines the order in which key channels are to be interleaved. The only difference between the two types is that all explicitly defined channels within a TimeCritical argument string will be marked as Real Time.

The argument string is a list of channels separated by hyphens. Gaps can be defined by including Xs in the definition, which are to be filled by a further interleave definition. Padding blanks can be specified by including Bs in the string. The string will be used repeatedly until all channels have been fully output.

Only 2 levels of interleave definition are allowed, and only TimeCritical and Explicit interleaves allow for a further level of definition. TimeCritical interleaves can only appear as the first interleave, and no Xs can appear in a secondary Explicit definition.

Proportional

A proportional interleave uses the length of the data within each channel to calculate the order of sectors. Channels with more data will be added to the image more regularly than channels with less. In this way all channels will start and end at approximately the same point within the image.

Even

An even interleave outputs one sector from each channel in sequence until all channels have been fully output.

PaddedEven

A padded-even interleave outputs one sector from each channel in sequence until all channels have been fully output, just like an even interleave. However it will intersperse the packets of data for each channel with blank sectors to ensure that, like the proportional interleave, all channels start and end at approximately the same point.

There must be one and only one XAChannelInterleave statement within XAInterleavedFile definition.

Example

```
XAChannelInterleave Even
XAChannelInterleave TimeCritical 1-2-X-X PaddedEven
```

XAEndInterleavedFile

Function

Marks the end of an XA interleaved file definition.

Syntax

XAEndInterleavedFile

Remarks

The XAEndInterleavedFile statement closes the definition of an XA interleaved file record and returns control to the previous command level.

Example

```
XAEndInterleavedFile
```

XAFilePermissions

Function

Specifies the default file permission flags.

Syntax

XAFilePermissions <Attribute> [<Attribute>] [...]

Remarks

The valid attributes are:

- Owner_Read
- Owner_Execute
- Group_Read
- Group_Execute
- World_Read
- World_Execute

Any combination of attributes is allowed.

Example

```
XAFilePermissions Owner_Read
```

XAOwnerGroup

Function

Specifies the default file Owner Group Identifier.

Syntax

XAOwnerGroup <GroupIdentifier>

Default

The owner group defaults to zero.

Remarks

The owner group identifier is a number between 0 and 65535.

Example

```
XAOwnerGroup 27
```

XAOwnerUser

Function

Specifies the default file Owner User Identifier.

Syntax

XAOwnerUser <UserIdentifier>

Default

The owner user defaults to zero.

Remarks

The owner user identifier is a number between 0 and 65535.

Example

```
XAOwnerUser 15
```

XA Interleaved Channel Commands

XA Interleaved Channel commands define the contents of a channel within an XA File. The commands fall into the following categories:

File space allocation definitions	`AddLength`, `MinLength`
XA file attribute definitions	`XAAudioAttributes`, `XAEndOfRecord`, `XAFileAttributes`, `XATrigger`, `XAVideoAttributes`
File object definitions	`Source`
Command definition termination	`XAEndChannel`

AddLength

Function

Specifies the number of bytes to be added to the length of the channel contents.

Syntax

AddLength <Length>

Default

If no command is encountered, then AddLength defaults to zero.

Remarks

This statement specifies a number of zero filled bytes to be added onto the end of the channel contents. This channel may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore if we have a channel of say 400 bytes, an AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final record will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
AddLength 512
```

See also: MinLength

MinLength

Function

Specifies the minimum number of bytes of a file record.

Syntax

MinLength <Length>

Default

If no command is encountered, then MinLength defaults to zero.

Remarks

This statement specifies a minimum size of channel record. If the channel is smaller than this, then it will be blank filled to the specified size.

The channel may then be further zero filled to take it to a frame boundary.

AddLength will always be performed before MinLength. Therefore if we have a channel of say 400 bytes, an AddLength of 1800 bytes will increase this size to 2200 bytes. A MinLength of 2048 will have no effect but the final record will be written in 2 x 2048 bytes blocks (i.e. 2 frames).

Example

```
MinLength 512
```

See also: AddLength

Source

Function

Specifies that the contents of the channel are to be copied from the file specified.

Syntax

Source <FileName>

Remarks

This indicates that contents of the specified file should be used as the contents of the channel. Any incomplete frame of data at the end of the channel will be padded with zeros.

There can be one and only one sourced file within a channel definition

Example

```
Source c:\develop\data\psyqace.bin
```

XAAudioAttributes

Function

Specifies the channel's audio attribute flags.

Syntax

XAAudioAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the attributes default to the parent directory default.

Remarks

The valid attributes are:

- Emphasis_On and Emphasis_Off
- ADPCM_B and ADPCM_C
- Mono and Stereo

Any combination of the non-mutually exclusive attributes is allowed.

This statement is only valid within an XA track for a channel defined as being Audio.

Example

```
XAAudioAttributes Emphasis_On Stereo
```

XAEndChannel

Function

Marks the end of a channel definition.

Syntax

XAEndChannel

Remarks

The XAEndChannel statement closes the definition of an XAChannel record and returns control to the previous command level.

Example

```
XAEndChannel
```

XAEndOfRecord

Function

Specifies the positions of End Of Record flags within the channel.

Syntax

XAEndOfRecord <EOROffset> [<EOROffset>] [...]

Remarks

This statement specifies the positions of end of record flags within the channel. This command can be called more than once for a channel.

The EndOfRecord Offset is the number of frames into the channel (not the record) that the end-of-record marker is to be set at.

Example

```
XAEndOfRecord 15 30 45 90
```

XAFileAttributes

Function

Specifies the file attribute flags.

Syntax

XAFileAttributes <Attribute> [<Attribute>]

Remarks

The valid attributes are:

- Form1 or 1 and Form2 or 2
- Audio, Video and Data

Any combination of the non-mutually exclusive attributes is allowed.

Each channel must have form and data types defined explicitly within the channel definition.

Example

```
XAFileAttributes Form1 Data
```

XATrigger

Function

Specifies the positions of trigger flags within the channel.

Syntax

XATrigger <TrigOffset> [<TrigOffset>] [...]

Remarks

This command can be called more than once for a channel.

The Trigger Offset is the number of frames into the channel that the trigger marker is to be set at.

Example

```
XATrigger 15 30 45 90
```

XAVideoAttributes

Function

Specifies the default video attribute flags.

Syntax

XAVideoAttributes <Attribute> [<Attribute>] [...]

Default

If no command is encountered, then the Attributes default to Application Specific.

Remarks

The valid attributes are:

- ApplicationSpecific

or a combination of

- 640x480 and 320x240
- Clut1, Clut2, Clut4 and Clut8

Any combination of the non-mutually exclusive attributes is allowed.

Example

```
XAVideoAttributes ApplicationSpecific
XAVideoAttributes 640x480 Clut2
```

Chapter 5:
Tools and Techniques for Development with the CD Emulator

.CTI File Creation

An easy method to create emulator control (*.CTI) files is to use the CDGEN software to generate your CD-R image. Then save out the *.CCS file and convert it to a *.CTI file by using the routine CCS2CTI.EXE.

CCS2CTI is an MS-DOS command that converts CD-ROM format data from CCS to CTI. The format is simply:

```
ccs2cti [-o <file>] [-v] <ccs-filename>

o <file>     Output to file value of <file>.  By default, this is
             <ccs-filename>.cti.

-v           Verbose mode
```

such as

```
ccs2cti foo.ccs
```

This will produce `foo.cti`, which you can then use to build your CD-ROM image on the hard-drive.

You can also use the "gencti" program described in further detail below.

Refer to the following control (.CTI) file for an example of Playstation CD-ROM disc with interleaved audio and video. For more examples, refer to Appendix A.

File with interleaved audio and video

```
Disc CDROMXA
      MapFile vid.map
      LeadIn XA
            Empty        1000
            PostGap      150
      EndTrack
      Track XA
            Pause        150
            Volume             ISO9660      cd.ISO
                  PrimaryVolume
                        SystemIdentifier                "PSX"
                        VolumeIdentifier                "PSXTEST"
                        VolumeSetIdentifier             "PSXTEST"
                        PublisherIdentifier             "SONY"
                        DataPreparerIdentifier          "SONY"
                        ApplicationIdentifier           "SONY"
                        Lpath
                        Mpath
                        Hierarchy
                              XAFileAttributes    Form1  Data
                              File   PSX.EXE
                                     Source PSX.EXE
                              EndFile
                              Gap 26749
                              File Mov.str
                                     XASource mov.str
                              EndFile
                        EndHierarchy
                  EndPrimaryVolume
            EndVolume
      EndTrack
      LeadOut XA
            Empty        500
      EndTrack
EndDisc
```

Warning: In the CTI format, be careful about the layout of files on a CD The contents of a sub-directory must be physically contiguous.

Use the “layout-mode” of the CDGEN program to see the actual location layout.

Here’s an acceptable layout:

```
\DIR1\
\DIR1\AAA.DAT
\DIR1\BBB.DAT
\CCC.DAT
```

This is acceptable because the contents of directory "DIR1" immediately follow the declaration of "DIR1".

The next layout is unacceptable:

```
\DIR1\
\DIR1\AAA.DAT
\DDD.DAT
\DIR1\BBB.DAT   <---split contents of DIR1
\CCC.DAT
```

This is not acceptable because the contents of "DIR1" are scattered among the declarations of other directories and files.

CDBoot6x.bin Usage

Run the program CDMON, which will install a TSR that displays information from the CD emulator in the top right of a standard text screen. This information will continuously update as the emulation progresses. It can be switched on and off by holding down the *left* hand control and shift keys and pressing C.

This display will only work if:

(1) You install version 6.x of the boot code and

(2) You have the PC hardware interrupt configured on the CD emulator board and you have informed CDBIOS of the interrupt number.

If you install version 6.x of the boot code but do not have the interrupt set up properly or use an old version of CDBIOS then the emulator will fail to function correctly.

2MByte Emulation

Emulation sessions can be successfully executed with 8MByte RAM images, but the production Playstation and the debugging stations can only handle 2MByte RAM images. Make sure that your application emulates correctly with 2MBYTE, before going to GOLD DISC. Also make sure to remove pollhost and other debugging hooks (PC READS and WRITES) before going to GOLD DISC.

CDEmuPrm Instructions

This utility allows you to select normal or zero seek timings, and also to control errors within CD streams.

Both settings are unaffected by RESETPS (which causes the CD emulator's microcontroller to reload CDBOOT from the SCSI HD).

Requirements:

Use at least CDBOOT35/CDBOOT65.

Seek Control:

To select zero seek times:

```
CDEMUPRM s 0
```

To reset the seek time to the default algorithm parameters:

```
CDEMUPRM s 1
```

Error Control:

General usage:

```
CDEMUPRM e [prm1,,prmN]
```

The error insertion algorithm is activated by the DEX issuing a CdlReadS command to the CD sub-system. This command is used to playback streams. If an error occurs, then the DEX will receive an interrupt, but the sub-system won't retry to read the sector.

Errors can be inserted at a fixed frequency or randomly via seed and threshold parameters. A RESETPS will reset the internal parameters to what you originally specified, so that error patterns can be repeated on your CD sector stream.

The error algorithm is done before a CD sector is signalled ready for the DEX to receive it. This means that error patterns can be reproduced, since during seeks (which vary for the same seeks!) the Random() fn isn't called.

```
Rnd = Random()

err_Count--
       if(err_Count==0)
       {
              err_Count=err_Period       //Reset the Count

       if(Threshold>=Rnd)
              Signal to DEX that this sector contains an error.
       }
```

CDEmuPrm ERROR parameters, default values:

(UInt32)PERIOD	0000 0000
(UInt16)THRESHOLD	0000
(UInt47)SEED	7FFF FFFF FFF8

PERIOD is in CD sectors.
THRESHOLD is a 16 bit unsigned WORD.
SEED is a 47 bit WORD.

To obtain a fixed frequency:

Specify a PERIOD only.

```
CDEMUPRM e 100
```

This example will set up the system to produce errors every 256 sectors.

IMPORTANT: Use a PERIOD greater than 1.

To obtain a random error pattern:

Specify a PERIOD, a THRESHOLD and an optional SEED.

```
CDEMUPRM e 10 1000
```

This example will set up the system to produce an error every 16 sectors, but only if the current random value is >= 0x1000.

Some values that could be useful:

PERIOD	THRESHOLD	# of sectors read after 0x80 errors:
0x10	0xF000	0x4000
0x10	0x8000	0x0F00
0x10	0x1000	0x0880

To stop errors occurring:

```
CDEMUPRM e
```

This will set all parameters to their defaults.

Important Notes

The way the parameters are modified is fundamentally:

- Open door
- Modify parameters
- Close door

Parameters can't be modified on the fly during streaming playback. You should set the parameters up at the start of your emulation session and only adjust them at suitable times (i.e., after RESETPS 1).

cti2cd Instructions

"cti2cd" generates a CD without having to use CDGen. To use "cti2cd", follow these steps:

1. Write a .cti file.
2. Run cti2cd.

```
cti2cd <file> -c<cutter> -r<cutter id> -x<speed> -t
```

<file> specifies the .cti file
-t specifies test mode (use this option the first time around)
-c specifies the cutter type: Sony(1), Yamaha(2), Kodak(3)
-r is the SCSI ID of the cutter

Be sure the correct ASPI driver is installed for your adapter.

This has only been tested on 1540 compatibles.

CutCD Version 1.0

CutCD provides the user with the abilty to produce a Gold Disc CD from a CD emulator image stored on a SCSI HD. The program currently supports two CDR devices: the Sony CDW-900E and the Yamaha CDR-100 (firmware version1.08 or higher). To use the program, you must invoke it from the command line:

```
CutCD -s<hd id:partition no.> [options [-t] [-d<CDR id>] [-x<1,2 or 4> ]]
```

Note: / as well as - can be used to specify options.

Options.

S - This must be present. This gives where the image is located on the CD emulator. The number preceding the colon must be the SCSI ID of the HD. The number after the colon will be the particular partition where the image is located. This is the same as the option used to specify the image location using BuildCD/BCDFlat.

T - This is optional, although recommended the first time you use the program for a new image. This option enables the test mode. In this mode, the burning of the CD will be simulated, without actually burning a disc. The read power rather than the write power of the laser is used. This option is useful for setting up the program and determining at which speed it is capable of cutting a CD.

D - This is optional. This option will specify the location (the SCSI ID) of the device you wish to use to cut the image. If this option is not present, the program will attempt to locate a device from those it finds on the SCSI bus. If you are sure of the device's location, then program speed will be greatly enhanced if the location is specified. If more than one cutter is found on the bus, you will be asked to specify which device you wish to use, from a list of options.

X - This enables the selection of the cutting speed. The default option is single speed cutting.

Once the program has begun you will be shown a detailed breakdown of information concerning the image you are cutting and the cutting device you have specified. This is accompanied by an image completion counter. Note: if the cutter is stopped or too fast a cutting speed is selected for your computer, any disc not finished will, in most instances, be unusable. It is recommended that this program not be run whilst multi-tasking under Windows.

!! IMPORTANT !!

Futhermore, the image must have been built using BuildCD or BCDFlat version 2.32 *or* greater. This also means that the complementary tools CDDisk and UpdateCD *must* be compatible. The lowest versions of these that should be used are 1.17 and *1.18,* respectively. If an older release is used, the programs will report the disc *image* to be of an invalid mode or form. Also note that cutter and HD are *both* connected to the SDevTC (formerly known as PSY-Q) scsi card, i.e., the program does not yet run off ASPI.

```
cutcd -s0:2 -d2 -t
```

This command will perform a test cut to the cutter, located at SCSI ID 2, from partition 2 of the HD located at SCSI ID 0. Once this test run has been completed successfully, the command

```
cutcd -s0:2 -d2
```

will cut the disc properly.

Note: Please make sure before cutting that you have modified the size of the partition on which the image is located so that it is of minimum size. Otherwise the program will write any excess to the disc. You will probably get better performance if you run the program natively in DOS, although it has been used in a Windows DOS box. When you load CDBios, use the /d option to specify a DMA channel to increase performance. There will be less likelihood of the buffer in the CDW or CDR machines underflowing. Note also that the CDW will take a noticeable time between starting the image write (where it writes the Lead-In

Track) and when it starts to use the image data. There is a small problem with setting the timeout in this part of the program, due to selecting x1 or x2 speed. Most developers have expressed that x2 cutting on the CDW can be unreliable. Therefore the timeout is currently optimized for single speed cutting to ensure reliabilty of data transfered. The writer ID of the CDW-900e must be 0 (zero).

Finally, if you are cutting an image shorter than 4 minutes, add some dummy data to the image using AddLength or Minlength commands.

GenCTI Instructions

GenCTI is designed to make to creation of .cti files (.cti being the usual extension for CD emulator and CD cutting software control files) easier for those people who have not constructed one before. It is not the aim of this program to generate fully any but the most simple and straightforward of control files. Saying that, the program does provide a certain amount of flexibility, such as the specification of wildcard inclusions.

There are two programs which rely on these control files, namely, BuildCD (along with a flat model version known as BCDFlat) and CTI2CD. BuildCD is for use with the CD emulator, and CTI2CD will cut CD gold discs (presently on the Sony CDW-900E and Yamaha CDR-100 dev ices), based on a control file.

GenCTI is a DOS command line program. To run it and display a full list of options type:

```
>gencti
```

at the command prompt.

The following will be displayed:

```
GenCTI version 1.0 Beta
Copyright (c) SN Systems Ltd. 26 March 1996

Format: GenCTI outfile [options]
Purpose: Automatically generates a cti file from the file contained in the directory
and subdirectories from where the program is invoked. Directory can be
overridden, see options.'outfile' given default extension .cti
Options:
        /t=target           Playstation - PlayStation specific
                            saturn - saturn specific
                            (default mode 1, data)
        /dir=<dir>          specify directory to be root on CD
        /system=<file>      specify filename for system file
        /nosubdirs          do not recurse subdirectories
        /wildcard=<file>    specify wildcard list
        /audio=<dir>        specify directory for audio tracks

Note: Only the first letter of each option needs to be specified.
      - or / may be used as switches.
```

The following are definitions of the options:

Option **t=target**. This option explicitly defines the target machine and generates a .cti file suitable for the machine.

Option **dir=<directory>**. By default, the program will use the directory from where it is called as the root directory in the .cti file. To override this, specify a different directory, (full path) with this option.

Option **system=<system file>.** The system file is a special file, specific to individual machines, territories or developers, which is positioned before the main disc data. This option enables the location and name of this system file to be specified and correctly included.

Option **nosubdirs**. This option speaks for itself, sub directories will not be included. The control file will contain only one directory, the root.

Option **wildcards**. This option is a bit more involved than the others, relying on the creation of a wildcard file. Just create a text file including the wildcard definitions, each separated by a newline/return and save. The wildcard definitions can be 31 characters maximum. This file is then added as the file specified in the option.

The following is an example of a wildcard file: wild.wcd.

```
*.bin
```

```
s*.tex
game?.aud
t*.ma*
*.dat
lev*.00?
```

The command line option would then appear as

```
/wildcard=wild.wcd
```

Presently a maximum of 15 definitions are allowed.

Option **audio**. This option requires further explanation to avoid including audio files twice. The directory specified for the location of audio tracks should not lie as a subdirectory off the directory specified as the root. Using this will generate a separate audio track for each file present in the audio directory.

Note: wildcards apply throughout the program if specified. If wildcards are specified and an audio directory is specified, the wildcards must contain inclusion information for the audio files. E.g., if the audio tracks all have the extension .wav, then *.wav must be included in your wildcard list.

MultiCD Instructions

MultiCD can be used to change to a 'new CD' without using CDDisk to write a new partition offset onto the HD or having to reset the target. The 'new CD' can be another partition on the current HD or on any other HD on the SCSI bus that contains valid CD images.

For DEX boards, use at least CDBoot30.BIN or CDBoot60.BIN.

For Saturn, use at least SatCD12.BIN or SatCD22.BIN.

Use at least CDBIOS.COM v1.04.

For full feedback using CDBoot60.BIN, use CDMon v1.02. (This will display the partition baseblock on an emulator reset.)

Usage: `MultiCD [id [partition]]`

```
MultiCD id partition
```

This simulates an open door for one second, then changes to the new CD at the specified SCSI ID and partition.

If an ID is selected with no HD, then you will receive a 'Target did not respond' message, but as a side effect the emulator will begin open door simulation. To close it, either reissue the command with a valid ID and partition or close it via the MultiCD main menu.

Chapter 6:
Modifying the Emulation Image

Modifying the Emulation Image-Using UPDATECD

BuildCD will generate the whole CD image each time it is run. However, if only one file has changed then modification can be a very lengthy process. UpdateCD, however, will determine which files have changed since the last creation/modification took place and will only write those to the HD. It requires the name of the map file that was generated by BuildCD as its parameter.

If BuildCD was run using the following command line:

```
BUILDCD-mexample.map -s0:1 example.cti
```

then the file `example.map` will contain all the relevant information to allow UpdateCD to write only files that changed since last time.

So if immediately after the above example the following was run

```
UPDATECD example.map
```

then nothing will be updated.

If, however, one of the files written by BuildCD was modified, then UpdateCD will detail the consequences of the change and ask the user if this modification should be made.

There will be cases where a file increases in length so much that it will affect following files. Then the user will need to modify the space allocation in the original control file and rebuild using BuildCD.

UpdateCD takes the following command line form:

```
UPDATECD [options] [map filename]
```

Available options are:

`-e<num>`	To restrict the number of warnings messages given.
`-w`	To suppress all warnings.

The mapfile contains **all** the information needed to do its updating, including the ID and partition number of the disk that the original image was created on.

Chapter 7:
Connecting Multiple Emulation Hard Drives

CD Emulator

Hard Drive Connections

As you know, this emulator system is based upon its local implementation of a SCSI bus. This electronics standard allows for up to 8 devices to share a common data pathway. However, to avoid reflection of signals from the ends of the bus, each end needs to be terminated, a termination simply being a resistor connection between each of the signal lines on the bus and ground.

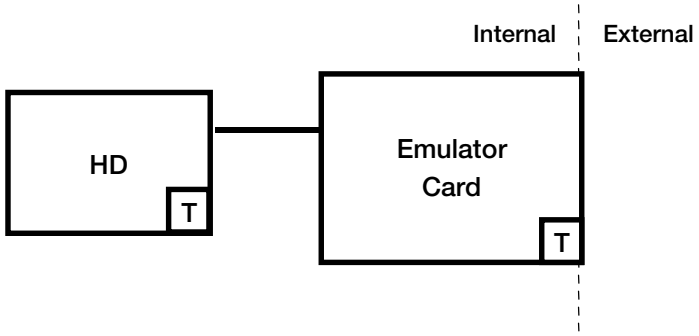
Hard drives usually have a configurable termination, and are generally shipped with this active. The emulator board also has this termination fitted and can be considered as 'one end' of the bus. Therefore, with only one internal drive fitted 'both ends' of the bus are terminated. This also applies if an external drive is fitted. However, if you wish to use more than one hard drive, then you need to determine which devices are 'on the ends' and remove the terminations from the ones 'between'.

Refer to the manual supplied with your internal hard drive to determine how to remove the termination.

External hard drives generally have two connectors, one of which is connected to the SCSI cable and the other connected to a termination pack.

So with an internal hard drive, schematically the system is:

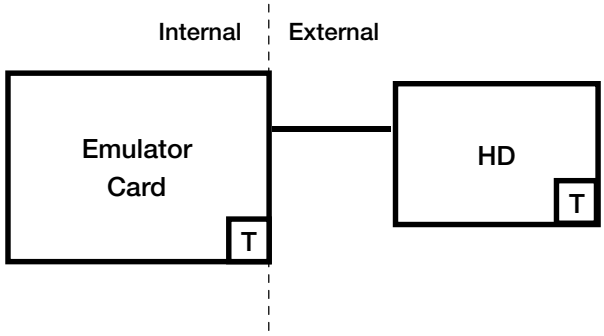
Figure 7–1: Emulator with Internal Hard Drive



with both ends terminated (T).

With an external hard drive:

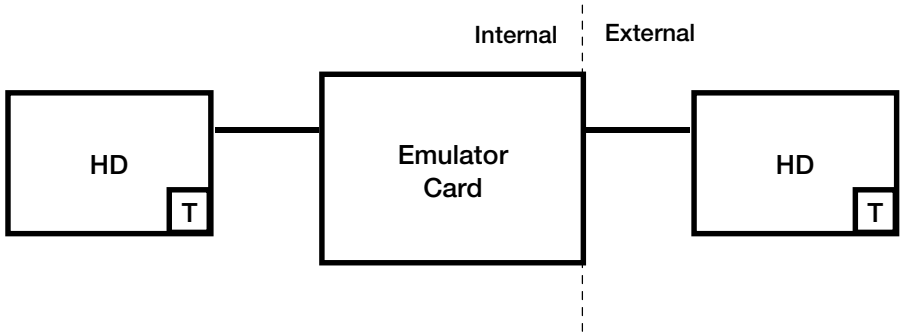
Figure 7–2: Emulator with External Hard Drive



Again, both ends are terminated.

If the requirement is for an internal and an external drive then:

Figure 7–3: Emulator with Internal and External Hard Drives

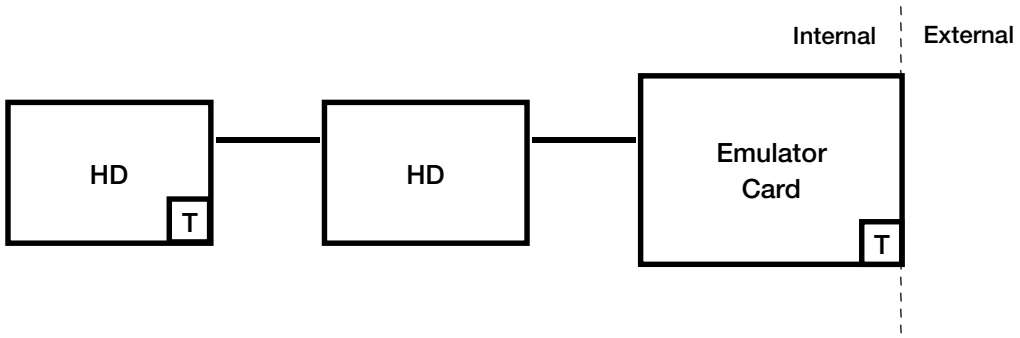


is how the chain should be configured.

NOTE: The termination is missing from the emulator board (see later for how to remove the termination from the card).

For two internal drives:

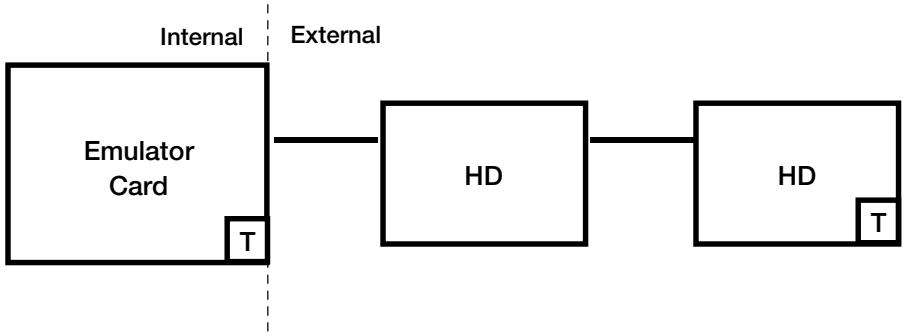
Figure 7–4: Emulator with Two Internal Hard Drives



NOTE: The termination has now been disabled from the middle device (instructions to disable the termination from a hard drive will be in the manual supplied with it).

For two external hard drives:

Figure 7–5: Emulator with Two External Hard Drives

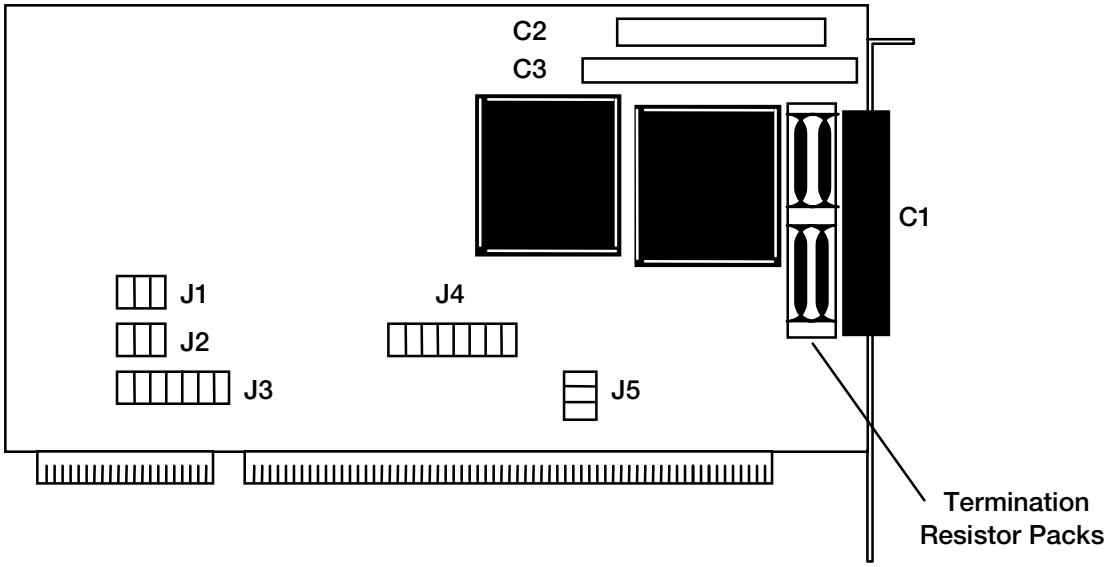


For configurations that need more than two hard drives (either internal, external or a mixture of both) the above still applies. It is only the devices that are on each end of the bus that need to have termination.

Removing the Termination Resistors from the CD Emulator Board

In some of the above examples the emulator board did not require termination. However, it is supplied with termination resistors fitted.

Figure 7–6: Emulator Termination Resistors



Each of the resistor packs are fitted into sockets to allow for easy removal. However, not all the packs are of the same value. Mark and note the position and orientation of each of the packs prior to removal. This will facilitate their easy replacement. Be very careful not to bend the pins of these packs during removal.

CD Emulator

Appendix A:
Sample Control Files

Sample Control Files

An example of a high-level disc-defining control file. Note how it includes example file 2, below, and therefore processes it.

```
Echo “Building Image”

Define              PsyQ c:\psyQ\dev\
ShowDefines

Disc CDROM state.img

CatalogNumber       47646
MapFile             [PsyQ]state.map
BInaryFileOutput    [PsyQ]state.cdb

LeadIn              AUDIO
Empty               150
EndTrack

Track               MODE1
Pause               150
SubcSource          [PsyQ]subc\subc1.sbc
                    Include state.ctl;  Volume Definition
SubcEmpty
EndTrack

Track               MODE1
SubcSource          [PsyQ]subc\dsubcs.sll
Source              [PsyQ]state.trk
SubcEmpty
PostGap             150
EndTrack

Track               Audio
Pause               150
Source              [PsyQ]state1.trk
EndTrack

LeadOut             AUDIO
Empty               150
EndTrack

EndDisc
```

An example of a volume-defining control file. Although this file can be included in file 1 above, it can also be used on its own.

```
Echo    Control file test for directory structures

Volume ISO9660 state.trk

    SystemArea        [PsyQ]share.exe

    PrimaryVolume

        AbstractFileIdentifier          AUTOEXEC.BAT
        ApplicationIdentifier           “_AUTOEXEC.BAT”
        ApplicationUse                  [PsyQ]appuse.txt
        BibliographicFileIdentifier     AARON.PTT
        CopyrightFileIdentifier         CONFIG.SYS
        DataPreparerIdentifier          _AUTOEXEC.BAT;1
        DescriptorWrites                1

        PublisherIdentifier             “SNSYSTEMS”
        SystemIdentifier                CD
        VolumeCreationDate              10/17/198908:12:00:00 00
        VolumeEffectiveDate             09/18/1990 09:13:00:00 00
        VolumeExpirationDate            10/04/1991 07:15:22:00 -12
        VolumeIdentifier                TOME
        VolumeModificationDate1         12/25/1993 00:00:00:00 00
```

```
        VolumeSetIdentifier             TICTACTOE

        Hierarchy

           File       AUTOEXEC.BAT;1
           Source     \autoexec.bat
           EndFile

           File       CONFIG.SYS;1
           Source     \config.sys
           EndFile

              Directory    CHKDEMO
                SourceDirectory \chkdemo SubDirectories SpareSpace=100%
              EndDirectory

              Directory    GRAPHICS

                File       FRED.PTT       3
                Source     \setnil.bat
                EndFile

                File       GINGER.PTT     4
                Source     \setmap.bat
                EndFile

                Directory  MONGEESE

                   File    ARNIE.PTT      5
                   Source  \setnml.bat
                   EndFile

                EndDirectory

                File       AARDVARK.PTT   6
                Source     \config.nil
                EndFile

              EndDirectory

              Directory    GRAPHICS1

                MinLength  2045
                AddLength  200

                File       FRED.PTT     3
                Source     \config.nml
                EndFile

                File       GINGER.PTT   4
                Source     \config.ptt
                EndFile

                Directory  MONGEESE

                   File    ARNIE.PTT    5
                   Source  \setmap.ptt
                   AddLength2048
                   EndFile

                EndDirectory

                File       AARDVARK.PTT 6
                Source     \setmap.blr
                EndFile

              EndDirectory

              File    AARON.PTT
              Source  \setmap.blt
              EndFile

        EndHierarchy

     Lpath
     MPath
```

```
    EndPrimaryVolume

    VolumePartition

        DescriptorWrites                3
        SystemIdentifier                CD
        SystemUse                       \setmap.crd
        VolumePartitionData             \snlpt.ddt
        VolumePartitionIdentifier       PARTITION1

    EndVolumePartition

    BootRecord

        BootIdentifier                  V1.0
        DescriptorWrites                2
        SystemIdentifier                CD
        SystemUse                       \setmap.crg

    EndBootRecord

EndVolume
```

Example XA Control File

demo.cti

```
Define              PsyQ c:\psyQ\
ShowDefines

Disc                CDROMXA demo.img

CatalogNumber       47646
MapFile             demo.map
BInaryFileOutput    demo.cdb

LeadIn              XA
Empty               1000
PostGap             150
EndTrack

Track               XA
Pause               150
SubcSource          \autoexec.bat
Include             demo.ctl          ; Track definition
SubcEmpty
PostGap             150
EndTrack

Track               Audio
ISRC                UKSNS9400001
SubcSource          bcdmap.obj
Source              \dev\trk\state1.trk
SubcEmpty
EndTrack

Track               Audio
Copy                On
PreEmphasis         On
Source              \dev\trk\state1.trk
EndTrack

LeadOut             AUDIO
Empty               150
EndTrack
EndDisc
```

demo.ctl

```
Volume              ISO9660      demo.trk

SystemArea          [psyq]sysarea.exe

PrimaryVolume
```

```
AbstractFileIdentifier           ABS.TXT
ApplicationIdentifier            "_APP.TXT"
ApplicationUse                   [psyq]txt\appuse.cdn
BibliographicFileIdentifier      BIB.TXT
CopyrightFileIdentifier          CPY.TXT
DataPreparerIdentifier           _DTP.TXT
DescriptorWrites                 1

PublisherIdentifier              "SNSYSTEMS"
SystemIdentifier                 CD
VolumeCreationDate               10/17/1989 08:12:00:00 00
VolumeEffectiveDate              09/18/1990 09:13:00:00 00
VolumeExpirationDate             10/04/1991 07:15:22:00 -12
VolumeIdentifier                 Vol_Id
VolumeModificationDate           12/25/1993 00:00:00:00 00
VolumeSetIdentifier              Vol_Set_One

Hierarchy

XAFileAttributes    Form1        Data
XAVideoAttributes   640x480      Clut4
XAAudioAttributes   Emphasis_On  ADPCM_B Mono

XAFilePermissions   World_Read   World_Execute
XAOwnerGroup        56
XAOwnerUser         1

File        ABS.TXT
   Source[psyQ]txt\abstract.txt
EndFile
File        APP.TXT
   Source[psyQ]txt\app_id.txt
EndFile
File        BIB.TXT
   Source[psyQ]txt\biblio.txt
EndFile
File        CPY.TXT
   Source [psyQ]txt\cpyright.txt
EndFile
File        DTP.TXT
   Source[psyQ]txt\dataprep.txt
EndFile

; File with channels interleaved within the BuildCD program
;
  XAInterleavedFile              autoexec.bat;1

     XAChannelInterleave         TimeCritical    2-X-2-X   Even

     XAChannel  1
            Source               C:\dev\buildcd\bcdmain.c
            XAFileAttributes     Form1  Data
         AddLength 2332
     XAEndChannel

     XAChannel  2
            Source               C:\dev\buildcd\bcdbasic.c
            XAFileAttributes     Form2  Audio
         XAEndOfRecord           1 8 EndOfFile
         XAAudioAttributes       Emphasis_Off
     XAEndChannel

     XAChannel  3
            Source               C:\dev\buildcd\bcdfront.c
            XAFileAttributes     Form1  Video
     XAEndChannel

  XAEndInterleavedFile

; File defined within XA with no interleaves
```

```
;
   File CONFIG.SYS;1

      XAFileAttributes           Form2  Audio
      XATrigger  0 2 4
      XAEndOfRecord              EndOfFile 1

      Source  \config.sys
      MinLength 10000

   EndFile

   AddLength 2048
   MinLength 8192

   Directory CHKDEMO
      SourceDirectory \chkdemo   SubDirectories SpareSpace=100%
   EndDirectory

   Directory  GRAPHICS

; File with channel interleaving already performed
;
            File                  FRED.PTT    3
               XASource           \setnil.bat
            EndFile

            File                  GINGER.PTT  4
               Source             \setmap.bat
            EndFile

            Directory             MONGEESE
               File               ARNIE.PTT   5
               Source             \setnml.bat
               EndFile

            EndDirectory

         EndDirectory

   EndHierarchy

   Lpath
   MPath

   EndPrimaryVolume

   VolumePartition

      DescriptorWrites           3
      SystemIdentifier           CD
      SystemUse                        \setmap.bat
      VolumePartitionData        \dev\buildcd\buildcd.exe
      VolumePartitionIdentifier  PARTITION1

   EndVolumePartition

   BootRecord

      BootIdentifier             V1.0
      DescriptorWrites           2
      SystemIdentifier           CD
      SystemUse                  \setmap.bat

   EndBootRecord

EndVolume
```

The following is a sample map file.

```
================================================================================
= Generated Map File  C:\DEV\BUILDCD\STATE1.MAP for Image File
                      C:\DEV\BUILDCD\STATE1.IMG
=
================================================================================
Image  C:\DEV\BUILDCD\STATE1.IMG 03/10/1994 14:31:48   2967552
       (CAT:0000000047646)
================================================================================
= Command             Fr Time     Len    LBN    Additional
================================================================================

LeadIn 0                                        XA
  Empty               00:00:00    150
  PostGap             00:02:00    150

Track  1                                        XA
  Pause               00:04:00    150
  TrackDef
     SystemArea       00:06:00    16     0      09/04/1991  05:00:00  10912
        Form1 C:\SHARE.EXE
     PrimVol          00:06:16    1      16
     TermVol          00:06:17    1      17

     Lpath            00:06:18    1      18
     OptLpath         00:06:19    1      19
     Mpath            00:06:20    1      20
     OptMpath         00:06:21    1      21
     Dir()            00:06:22    1      22
     Dir(GRAPHICS)    00:06:23    1      23
     File(FRED.PTT)   00:06:24    1      24     00:06:23  31/03/1994  13:42:30  62
        Form1 C:\SETNIL.BAT
     File(GINGER.PTT) 00:06:25    1      25     00:06:23  02/08/1994  16:25:56  320
        Form1 C:\SETMAP.BAT
     File(AARDVARK.PTT) 00:06:26  1      26     00:06:23  24/03/1994  12:20:58  111
        Form1 C:\CONFIG.NIL
     Dir(MONGEESE)    00:06:27    1      27
     File(ARNIE.PTT)  00:06:28    1      28     00:06:27  21/04/1994  13:52:12  62
        Form1 C:\SETNML.BAT
  EndTrackDef
  PostGap             00:06:29    150

Track  2                                        Audio(COPY:OFF PRE_EMPH:OFF)
  Pause               00:08:29    150
  Source              00:10:29    73           02/10/199418:00:12170563
       C:\DEV\BUILDCD\BCDWRITE.C

Track  3                                        Audio(COPY:OFF PRE_EMPH:OFF)
  Pause               00:11:27    150
  Source              00:13:27    7            03/01/1994 02:20:00
16384  H:\AMEOL\DOWNLOAD\WINSORT.EXE

LeadOut 4                                       Audio(COPY:OFF PRE_EMPH:OFF)
  Empty               00:13:34    150
```

Appendix B:
ISO Character Sets

ISO Character Sets

Figure B–1: ISO 'd' characters (shown unshaded)

least significant nibble	0	1	2	3	4	5	6	7
	0	NUL	DLE	SP	0	@	P		p
	1	SCH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EDT	DC4		4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	'	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	}
	C	FF	IS4	,	<	L	\	l	|
	D	CR	IS1	-	=	M	]	m	}
	E	SQ	1S4	.	>	N	^	n	~
	F	S1	1S3	/	?	O	_	o	DEL

Figure B–2: ISO 'a' characters (shown unshaded)

least significant nibble	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P		p
1	SCH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EDT	DC4		4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	}
C	FF	IS4	,	<	L	\	l	|
D	CR	IS1	-	=	M	]	m	}
E	SQ	1S4	.	>	N	^	n	~
F	S1	1S3	/	?	O	_	o	DEL

Appendix C:
SCSI Errors

SCSI Errors

The hundreds and thousands places in a hexadecimal error code can be interpreted as the following:

Table C–1: SCSI Error Definition

Sense Key	Description
00	No sense
01	Recovered error
02	Not ready
03	Medium error
04	Hardware error
05	Illegal request
06	Unit attention (could be unit has been reset)
07	Data protect (write protected)
08	Blank check
09	Vendor specific
0A	Copy aborted
0B	Aborted command
0C	Equal (for search data requests)
0D	Volume overflow
0E	Miscompare
0F	Reserved