CD Emulator

Developer

Reference

Series

September 1995

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

Publication date: February 1995
Reprinted: September 1995

Contents

1 INTRODUCTION

This CD Emulator has been designed for use exclusively with the Sony Playstation Development tool DTL-H2000. 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 it is connected to.

We recommend that you install the DTL-H2000, libraries and Psy-Q development software first, and take time to familiarise yourself with how it functions prior to installing the CD-emulator.

Please take a few minutes to read the following chapter describing the emulators design and function. An understanding of the system components and how they function will lead to a quicker and easier installation.

As part of our commitment to you the developer, we have designed this system with flexibility and upgradability in mind. We have tried to include all the functions we think you may require in the development of CD based products. However, if there are functions that you feel are missing, the Playstation development support group for your territory will be happy to forward your suggestions to us. We cannot guarantee that all suggestions will be included in future updates but we will do our best to try.

2 SYSTEM OVERVIEW

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 would go directly to a CD-Rom drive, which would respond and retrieve data from a CD. However, the DTL-H2000 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 connected to the emulator.

This emulator consists of three (logically) different sub sections:-

the emulation hard drive(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 which connects each of the above sections in the following way:-

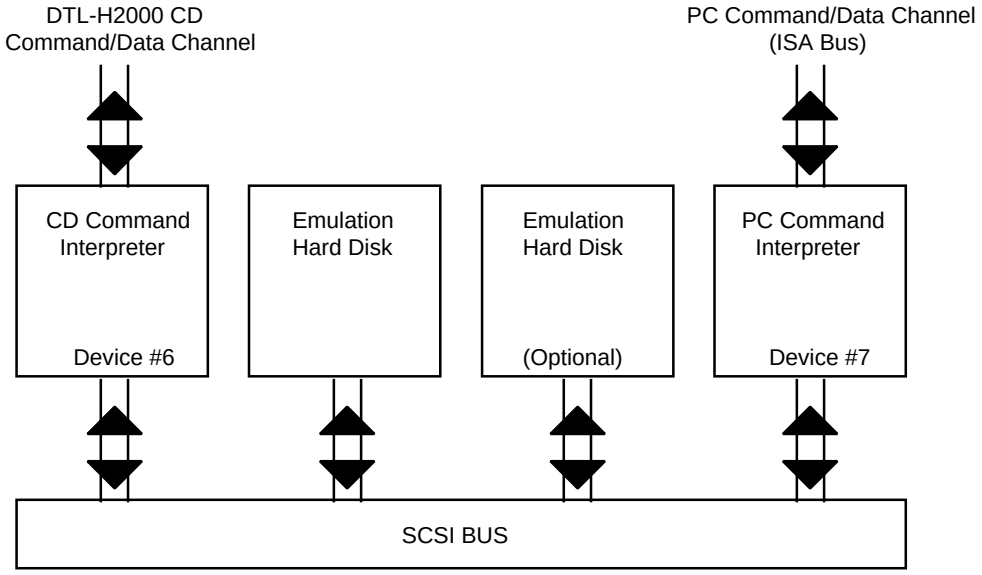


Fig 1.

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, leaving six free for connection to emulation hard drives. We do not anticipate many users will need more that one hard drive (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, however, 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 double speed CD Drive a certain data transfer rate needs to be maintained between the emulation HD and the DTL-H2000 CD command channel. We have found that some hard drives cannot maintain this data transfer rate. The manufacturer's specifications for HDs declare an average rate which may seem enough to satisfy the emulators requirements. However, most drives need to re-calibrate themselves periodically to account for thermal changes. During this period the data rate on some drives fall off and lead to stalls in the emulation data flow. This is observed as glitches in film playback or interruptions in sound replay.

Currently the only ones we have tested and found reliable are :-

Micropolis 4110AV
*IBM Spitfire (0662)
*IBM Pegasus

*Please ensure that the auto start jumper is set prior to use of any IBM drive or it will not appear to work.

Drive manufacturers are constantly updating their range of drives; please contact Sony Technical Support for a list of currently available drives.

(Initially we recommended the Digital DSP3107L hard drive. However during extended testing we have found that after resetting, the drive occasionally re-calibrates after 30 seconds. This causes a once only stall in the data flow, after this the drive operates within specifications.)

Now 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:-

i how to configure your emulator card.
ii plugging the card into the PC
iii connecting the emulation HD
iv connecting the emulator to the DTL-H2000
v installing the software
vi setting up the emulation HD ready for image building and emulation
vii building and testing the example program.

and once complete the CD emulator is ready for use.

3 INSTALLING THE SYSTEM

The Psy-Q CD emulator board should be fitted into an empty 16 bit slot in the host PC.

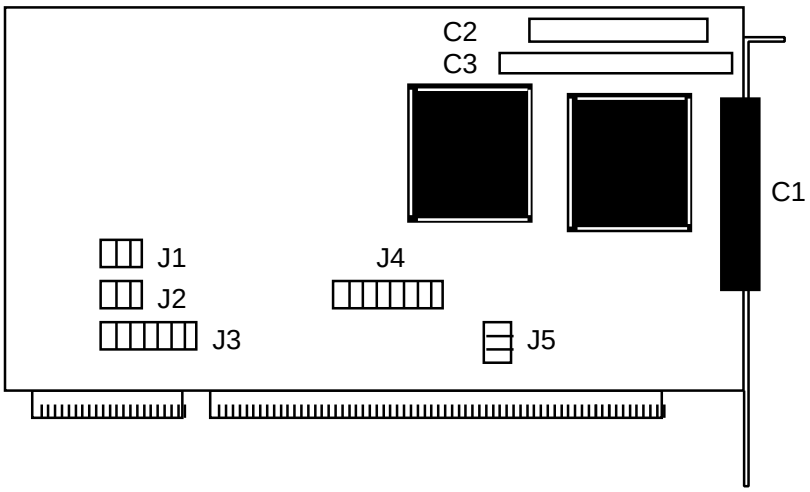


Fig 2. (Not all chips shown)

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)
C3	internal SCSI connector
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 and 5
J4	I/O base address select. Possible settings are (from left to right) 300,308,310,318,380,390,398
J5	SCSI ID device number (also software selectable) Possible settings are from 7-0. This is a 3 bit binary selection so with the top two jumpers closed the ID will be 6

[J5 is set to 7 (all three jumpers connected). This is the PC command channel device number. See Fig 1. Note: this is also configurable from the command line.]

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

Turn off your PC and disconnect from the supply. Remove the cover and check all the cards in your system noting down their base addresses, irq settings and dma channel usage. Determine which channels/addresses are free and set the jumpers on the emulator card appropriately.

Install the emulator board in an ISA slot adjacent to the PIO board of the DTL-H2000. Referring to Fig 2. connect between C2 and the similar socket on the top edge of the PIO board using the small ribbon cable provided.

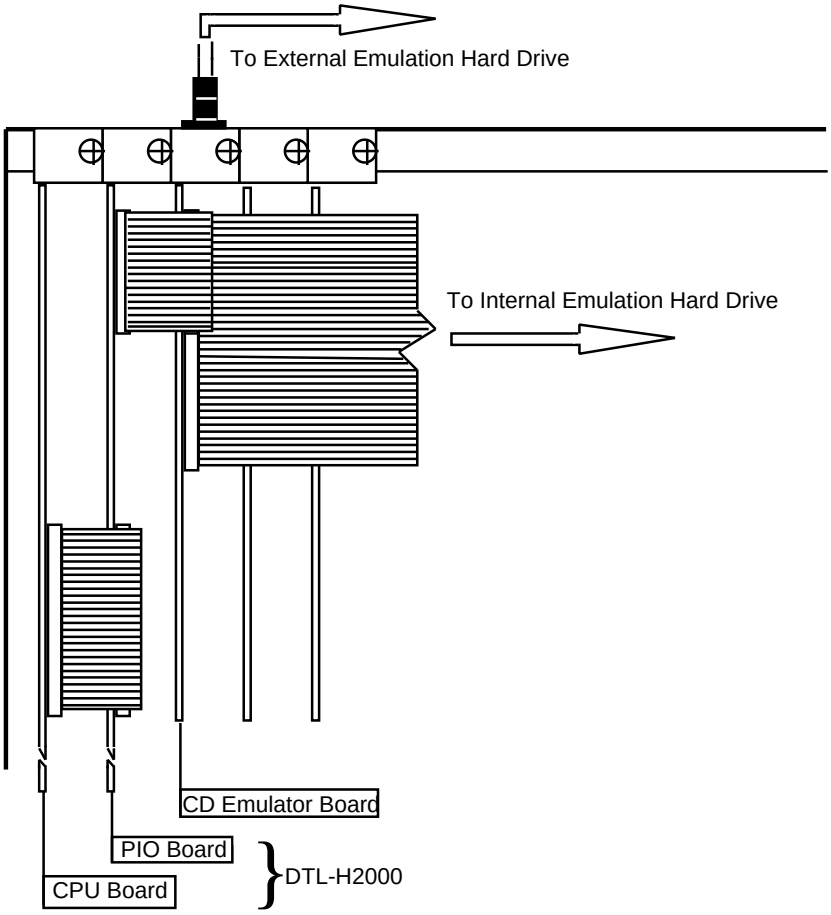


Fig 3.

If you intend to use an internal HD with the emulator, then now is the time to 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 this to the power socket of the HD. Note down 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 then connect this now.

Note
NEVER connect or disconnect any hard drives without first removing ALL sources of power.

3.1 Installing the software

You will have already created a PSYQ directory into which you have copied the assemblers etc. Copy all the files from the CD-Emulator disk into this directory.

e.g. COPY A:*.* C:\PSYQ

DEXBIOS.COM is a driver which you will be familiar with. It allows the code development system to communicate with the DTL-H2000. Similarly, we have provided CDBIOS.COM to provide communication services to the CD emulator (in addition to DEXBIOS.COM, not as a replacement).

You will probably have installed DEXBIOS.COM by adding a line to your AUTOEXEC.BAT file which automatically loads DEXBIOS.COM. The addition for CDBIOS is of the form:-

CDBIOS /aADDR /dDMA /iIRQ, where ADDR is the I/O address, DMA is the dma channel and IRQ is the interrupt request number of the board. (Refer to Fig 2.)

For Example:-.
CDBIOS /a308 /d7 /i15

will install the driver to use the emulator board with its factory settings.

Preparing for Emulation

4 PREPARING FOR EMULATION

How to use CDDISK.

4.1 CDDISK Introduction

CDDisk is the general emulation disk manager program, performing these functions -

Initialisation of the HD
Installation of the emulator card '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, initialises the partition sector

and

'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 card. However, when using more than one HD then special installation instructions need to be followed (see chapter 7) and its ID must be different from each of the 'devices' on the local SCSI bus.

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

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

So when the system is first installed, or a new HD is added, then it must be initialised 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 will be destroyed.

4.1.1 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 -7. It is looking to identify an emulation HD and 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. Booting in this way allows for future upgrades to the emulation system should it be necessary.

Only one of these 'boot' partitions needs to exist in multiple HD configurations and it is recommend 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:\PSYQ\CDBOOT.BIN 0
```

If this command was completed successfully then 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 CDBOOT.BIN

```
e.g.    C:\PSYQ\CDBOOT.BIN
```

Providing all is well no errors will be reported.

4.1.2 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 or one CD of up to 99 mins. (there is little point in defining a partition larger than this because the CD directory structure does not cater for this.)

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 Min CD will require a partition size of 270,000 frames or 1,350,000 sectors or 660 Mb.

So 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 min CD needing 270000 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 being the boot partition and partition 1 being 60 Min CD.

4.1.3 Setting the active partition

The act of creating this partition (above) has made this the currently active partition. this means that when the DTL-H2000 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 and can be viewed as being equivalent to inserting a real CD into a drive. If you have more than one HD then 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 de-selects all partitions on that HD.

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

```
CDDISK -a1 0
```

or CDDISK 0
select the 'Select active partition' option from the menu
and type 1

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

4.1.4 Modifying partition sizes

This can only be achieved through the option menus. Select the partition you wish to resize and enter the new size (only in sectors!). NOTE re-sizing a non-active partition will NOT set it as the active partition.

4.1.5 Deleting a partition

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

Generating Emulation Images

5 GENERATING EMULATION IMAGES - How to use BUILDCD

5.1 BuildCD Introduction

BuildCD is a program to create PsyQ 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.

5.1.1 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 that an audio data requires 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 then 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 this. The last track is another dedicated track called the leadout track, this is just a marker to flag the end of the disc. Neither the leadin or leadout tracks should have any data in them.

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

Mode0 sectors 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 (such as executable programs) where data integrity has to be maintained, 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 (such as graphics) where data integrity needed not be worried about too much, Mode2 frames can be used.

XA frames are a modification to the mode2 frame standard and allow two types of data record that can be freely interleaved.

Form1 mirrors mode1 in that it can hold up to 2048 bytes of data and also automatically includes cyclic redundancy checks and error correction.

Form2 mirrors mode2 in that it can hold 2324 bytes of data but again has no error correction.

5.1.2 BuildCD's output

The 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 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:

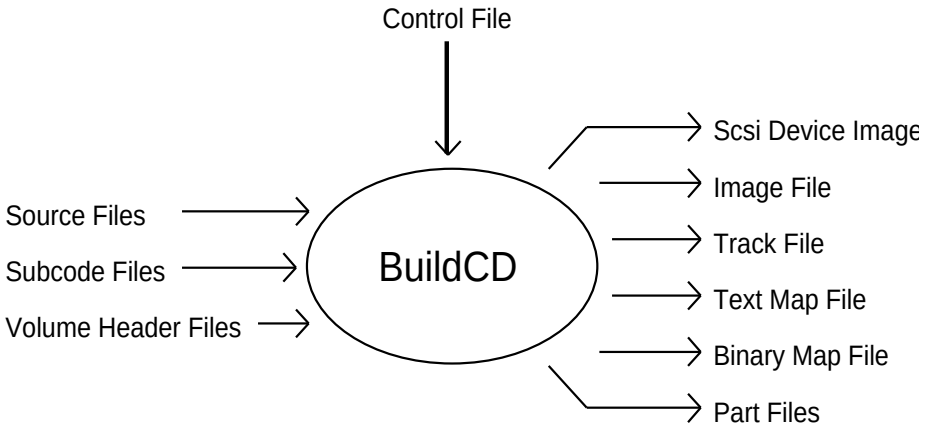


Fig1. BuildCD program I/O.

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 here 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 it's contents were to change then 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 which are already resident on disk). In fact these are binary representations of the CD Directories, PathTables and Volume headers. These Part files 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.

Sony CD Generator-Save File - The CD generator save files can be used by Sony's CD Generator program to import the CD specification creation of CD's prior to direct writing from within the BuildCD program.

Interleaved Output File - Interleaved output files are in the same format as those files imported using an XASource command within the control file command language. Interleave files are produced only when a -g command line argument is used and the output filename is specified in the XAInterleavedFile control file command.

5.1.3 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.

5.1.4 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 echo the case of characters within the control file faithfully.

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 then 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 it's 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 are used.

Another 'trick' that has been used by developers is 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. When the files have been created they need only be copied over the dummy files for the UpdateCD program to correctly load them.

5.2 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 in-sensitive. 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.

i.e. `Define PsyQ ""`

See also Page 5-3 - Command line arguments

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

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.

See also Define

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
```

5.3 OUTERMOST LEVEL COMMANDS

Outermost level commands are the highest level of command for a control file, only global commands can be recognised 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 mode 1 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 the program.

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
```

5.4 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 The CDGeneratorFile statement specifies the default save file name. 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 filename.

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 criteria 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 The LeadIn statement specifies the start of the leadin track definition.

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 sector 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.

See also EndTrack, Empty

Example
```
LeadIn  Mode0
```

LeadOut

Function Marks the start of a LeadOut track definition.

Syntax LeadOut <TrackType>

Remarks The LeadOut statement specifies the start of the leadout track definition.

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 sector 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.

See also EndTrack, Empty

Example
```
LeadOut  Mode0
```

MapFile

Function Specifies the name of the default text map file.

Syntax MapFile <FileName>

Remarks The MapFile statement specifies the default text map file name. 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 filename.

See also Chapter

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

Track

Function Marks the start of a standard track definition.

Syntax Track <TrackType>

Remarks The Track statement specifies the start of a standard track definition.

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.

See also EndTrack

Example

```
Track  Mode0
```

5.5 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 RECOGNISED 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 and specifies the state of the track's copy protection flag.

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. A frame is $^{1}/_{75}$th of a second of data.

This command is particularly useful for defining the length of LeadIn and LeadOut tracks that shouldn't technically 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 The Index statement positions an index point within the track. 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 then it 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 uppercase letters or digits
Owner code	3 uppercase letters or digits
Recording Year	2 digits
Serial Number	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 this 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 frame is $^{1}/_{75}$th of a second of data.

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, 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 this 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 frame is $^{1}/_{75}$th of a second of data.

A PostGap cannot be defined in audio tracks.
A PostGap must be a minimum of 150 frames.

Example
```
PostGap   150
```

PreEmphasis

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

Syntax PreEmphasis <Boolean>

Default The pre-emphasis defaults to FALSE.

Remarks This command is only valid within an audio track and specifies the state of the track's pre-emphasis flag.

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 this 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 end of the track. A frame is $^{1}/_{75}$th of a second of data.

A PreGap cannot be defined in LeadIn, Leadout or audio tracks.

A PreGap must be a minimum of 150 frames.

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 This indicates that contents of the specified file should be placed into the image. 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.

See also XASource

Example

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

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 The Volume statement specifies a track output file name. 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 This indicates that contents of the specified file should be placed into the image. 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.

See also Source

Example

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

5.6 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 The PrimaryVolume statement specifies the start of a primary volume definition. 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.

See also EndPrimaryVolume

Example
```
PrimaryVolume
```

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 to long then it's 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
```

5.7 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 statement specifies the name of the abstract file identifier. This file name must be specified in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level1 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 This statement specifies the application identifier within the volume descriptor. 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 level1 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 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 This statement specifies the name of the bibliographic file identifier. This file name must be specified in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level1 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 This statement specifies the name of the copyright file identifier. This file name must be specified in the root directory of the volume's hierarchy definition.

The filename must conform to ISO level1 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 This statement specifies the data preparer identifier within the volume descriptor. 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 level1 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 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 This statement specifies the number of times that the volume descriptor is to be written out. 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 (ie. 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 Hierarchy statement specifies the start of a directory hierarchy within a primary volume definition.

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.

See also EndHierarchy

Example

```
Hierarchy
```

LogicalBlockSize

Function Specifies the logical block size of the volume descriptor.

Syntax LogicalBlockSize <BlockSize>

Default The logical block size defaults to 2048.

Remarks This statement specifies the size of the volume set logical block size. Valid block size values are:

512, 1024 and 2048

THIS COMMAND IS RECOGNISED BUT HAS NOT BEEN IMPLEMENTED.

Example

```
LogicalBlockSize  2048
```

LPath

Function Specifies the position of an L type path table.

Syntax LPath

Remarks This statement specifies the position of an L type path table. 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 & M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

The 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.

See also MPath

Example

```
Lpath
```

MPath

Function Specifies the position of an M type path table.

Syntax MPath

Remarks This statement specifies the position of an M type path table. 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 & M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

The 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.

See also LPath

Example

```
MPath
```

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 This statement specifies the position of an additional L type path table. 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 & M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

The 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.

See also LPath, OptionalMPath

Example

```
OptionalLpath
```

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 This statement specifies the position of an additional M type path table. 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 & M. An L path table is set out least significant byte first, while an M path table is set out most significant byte first.

The 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.

See also MPath, OptionalLPath

Example
```
OptionalMpath
```

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 This statement specifies the publisher identifier within the volume descriptor. 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 level1 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 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 statement specifies the name of the system identifier with the string specified. 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 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 is 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 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.

See also GreenwichOffset

Example

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

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 is 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 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.

See also GreenwichOffset

Example

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

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 is 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 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.

See also GreenwichOffset

Example

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

VolumeIdentifier

Function Fills the volume identifier field with the specified string.

Syntax VolumeIdentifier <String>

Remarks This statement specifies the name of the volume identifier with the string specified.

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

The string is made up of 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 is 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 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.

See also GreenwichOffset

Example

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

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 This statement specifies the name of the volume set identifier with the string specified.

The string is made up of 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 This statement specifies the name of the XA disc's startup directory to be placed in the Primary Volume descriptor.

The string is made up of 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
```

5.8 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).

See also MinLength

Example

```
AddLength  512
```

Attributes

Function Specifies the directory attribute flags.

Syntax Attributes <Attribute>

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

Remarks This statement specifies the directory attributes. The valid attributes are:

Hidden or 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.

See also EndDirectory

Example
```
Directory  PSYQSRC
```

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.

See also EndFile

Example
```
File  MONSTERS.GPH  2
```

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 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).

See also AddLength

Example
```
MinLength  512
```

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 is 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 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.

See also GreenwichOffset

Example

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

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 This statement specifies the default XA audio attributes. 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 This statement specifies the default XA file attributes. 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 This statement specifies the default XA file permissions. 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.

See also XAEndInterleavedFile

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

XAOwnerGroup

Function Specifies the default file Owner Group Identifier.

Syntax XAOwnerGroup <GroupIdentifier>

Default The owner group defaults to zero.

Remarks This statement specifies the default XA owner group identifier. 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 This statement specifies the default XA owner user identifier. 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 This statement specifies the default XA video attributes. The valid attributes are:

ApplicationSpecific

or a combination of

640x480 and 320x200
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
```

5.9 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).

See also MinLength

Example

```
AddLength  512
```

Attributes

Function Specifies the directory attribute flags.

Syntax Attributes <Attribute>

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

Remarks This statement specifies the directory attributes. The valid attributes are:

Hidden or 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.

See also EndDirectory

Example

```
Directory  PSYQSRC
```

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 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.

See also EndFile

Example

```
File  MONSTERS.GPH  2
```

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 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).

See also AddLength

Example

```
MinLength  512
```

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 is 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 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.

See also GreenwichOffset

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

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 This statement specifies the default XA audio attributes. 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 This statement specifies the default XA file attributes. 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 This statement specifies the default XA file permissions. 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.

See also XAEndInterleavedFile

Example
```
XAInterleavedFile  MONSTERS.GPH
```

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 This statement specifies the default XA owner group identifier. 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 This statement specifies the default XA owner user identifier. 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 This statement specifies the default XA video attributes. The valid attributes are:

ApplicationSpecific

or a combination of

640x480 and 320x200
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
```

5.10 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, XAEOR, 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 had 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).

See also MinLength

Example

```
AddLength  512
```

Attributes

Function Specifies the file attribute flags.

Syntax Attributes <Attribute> [<Attribute>]

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

Remarks This statement specifies the directory attributes. 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 had 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).

See also AddLength

Example

```
MinLength  512
```

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 is 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 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.

See also GreenwichOffset

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

Source

Function Specifies that the contents of the file 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 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 This statement specifies the XA audio attributes. 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 position of End Of Record flags within the file.

Syntax XAEndOfRecord <TrigOffset> [<TrigOffset>] [...] [EndOfFile]

Remarks This statement specifies the position of end of record flags within the file. 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 Add & 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 This statement specifies the default XA file attributes. 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 This statement specifies the default XA file permissions. 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 This statement specifies the default XA owner group identifier. 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 This statement specifies the default XA owner user identifier. 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 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 setup within them this command cannot appear within a File specification with the following XA commands:

XAAudioAttributes,
XAEOR,
XAFileAttributes,
XATrigger,
XAVideoAttributes

This command is only valid in an XA track.

See also Source

Example

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

XATrigger

Function Specifies the position of Trigger flags within the file.

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

Remarks This statement specifies the position of trigger flags within the file. 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 Add & 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 This statement specifies the default XA video attributes. The valid attributes are:

ApplicationSpecific

or a combination of

640x480 and 320x200
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
```

5.11 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 This statement specifies the directory attributes. 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 is 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 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.

See also GreenwichOffset

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

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.

See also XAEndChannel

Example
```
XAChannel  3
```

XAChannelInterleave

Function Defines the format of the XA channel interleave.

Syntax XAChannelInterleave <I'type> [<arg>] [<I'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 & Explicit
These two interleave types are for the most part the same as they must be both followed by an argument string that define 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 a hyphen. Gaps can be defined by including Xs into the definition 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 This statement specifies the default XA file permissions. 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 This statement specifies the default XA owner group identifier. 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 This statement specifies the default XA owner user identifier. The owner user identifier is a number between 0 and 65535.

Example

```
XAOwnerUser  15
```

5.12 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, XAEOR, 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 had 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).

See also MinLength

Example

```
AddLength  512
```

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 had 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).

See also AddLength

Example
```
MinLength  512
```

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 This statement specifies the XA audio attributes. 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 position of End Of Record flags within the channel.

Syntax XAEndOfRecord <TrigOffset> [<TrigOffset>] [...]

Remarks This statement specifies the position 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 This statement specifies the default XA channel attributes. 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 from and data types defined explicitly within the Channel definition.

Example
```
XAFileAttributes  Form1  Data
```

XATrigger

Function Specifies the position of Trigger flags within the channel.

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

Remarks This statement specifies the position of trigger flags within the channel. This command can be called more than once for a channel.

The Trigger Offset is the number of frames into the file 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 This statement specifies the default XA video attributes. The valid attributes are:

ApplicationSpecific

or a combination of

640x480 and 320x200
Clut1, Clut2, Clut4 and Clut8

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

Example

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

Modifying the Emulation Image

6 MODIFYING THE EMULATION IMAGE - How to use UpdateCD

BuildCD will generate the whole CD image each time it is run, however, if only one file has changed then this can be a very lengthy process. UpdateCD, however, will determine which files has changed since the last creation/modification took place and only write those to the HD. It requires 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 re-build 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 information needed to do its updating including the ID and partition number of the disk that the original image was created on.

Connection of Multiple Emulation Hard Drives

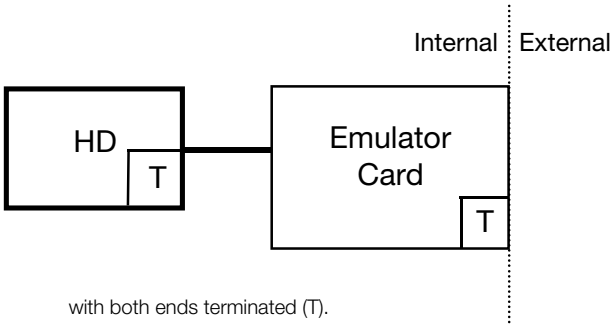
7 CONNECTION OF MULTIPLE EMULATION HARD DRIVES

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. This 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 card 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 device is 'on the end' and remove the termination from the ones 'between'

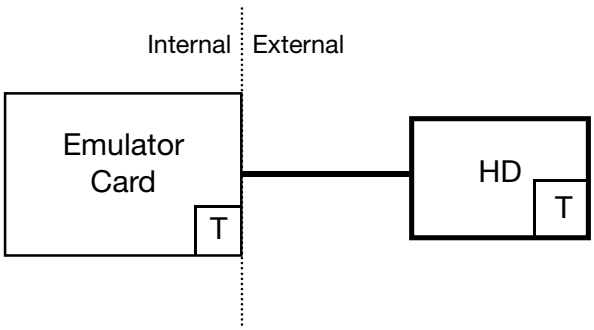
Refer to the manual supplied with your internal hard drive to determine how to remove the termination. External hard drives generally have two connectors, one to which is connected the SCSI cable and to the other is connect a termination pack.

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



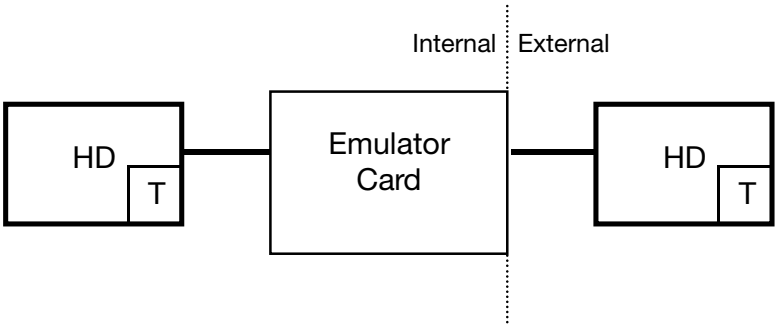
with both ends terminated (T).

With an external hard drive-



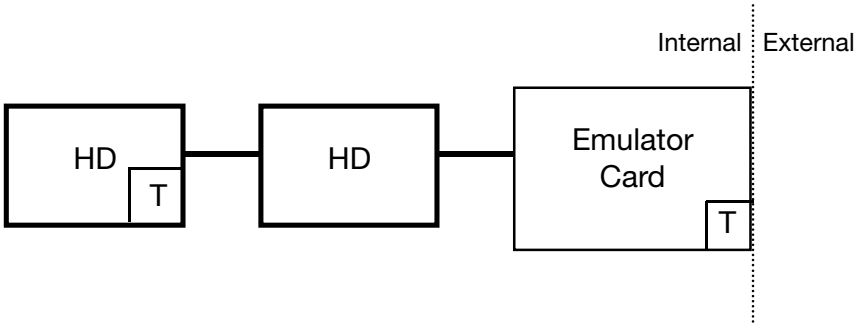
Again, both ends are terminated.

If the requirement was for an internal and an external drive then -



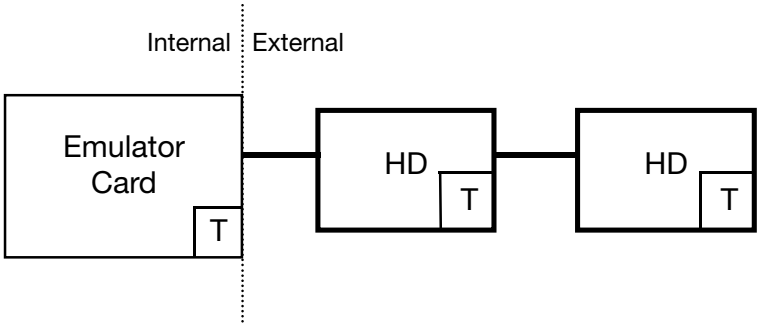
is how the chain should be configured. Note the termination is missing from the emulator card (see later for how to remove the termination from the card).

Or, for two internal drives -



Note that 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).

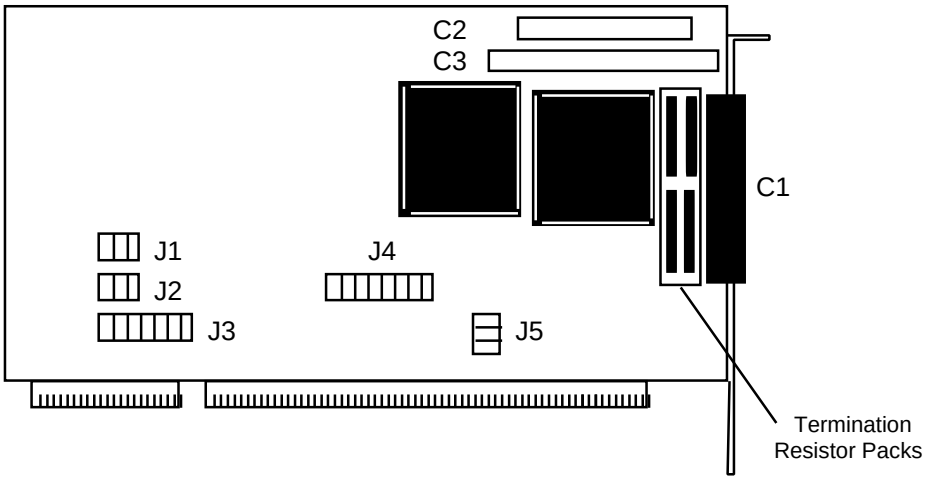
Or, for two external hard drives -



For configurations which 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.

7.1 Removal of the Termination Resistors from the CD Emulator Card

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



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.

Appendix A

A SAMPLE CONTROL FILES

An example of a high level disc defining control file. Note how it includes the example file 2 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. Note that although this can be included in file 1 above it can also be used on it's 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           10/17/1989 08:12:00:00 00
    VolumeEffectiveDate          09/18/1990 09:13:00:00 00
    VolumeExpirationDate         10/04/1991 07:15:22:00 -12
    VolumeIdentifier             TOME
    VolumeModificationDate       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
    AddLength 2048
    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                        02/10/1994 18:00:12   170563
        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

B ISO CHARACTER SETS

ISO 'd' characters - shown unshaded

least / significant / nibble	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P		p
1	SOH	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	IS2	.	>	N	^	n	~
F	S1	IS3	/	?	O	_	o	DEL

ISO 'a' characters - shown unshaded

least / significant / nibble	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P		p
1	SOH	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	IS2	.	>	N	^	n	~
F	S1	IS3	/	?	O	_	o	DEL

Appendix C

C SCSI ERRORS

The hundreds and thousands of a hex error code can be interpreted as the following:

Sense Key	Description
00	No Sense. Okay.
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

Index