CTrac™ BuildDisc ©

A compact disc image building tool.





Contents

Chapter 1 Introduction to BuildDisc	
1.1 About BuildDisc	2
1.2 Compact Disc Standards	
1.3 Running BuildDisc	//////
1.3.1 Command Line Options	
Chapter 2 Control Files	
2.1 Command and File Format	
2.2 An Example Control File	7
2.3 Control File Commands	9
2.3.1 Global Commands	 11
2.3.2 Outermost Level Commands	
2.3.3 Disc Commands	19
2.3.4 Track Commands	27
Appendices Appendix A Parsies Algorithm	42
Appendix R Warning Algorithm	
Appendix B Warning and Error Messages	44
Index	47

Credits

Software:

Documentation:

Todd Squires Donna Manning

This manual was produced using Microsoft Word from Microsoft Corp. and the index was produced using Sonar Bookends from Virginia Systems, Inc.

Copyright © 1991 ICOM Simulations, Inc. All rights reserved.

Your rights of ownership are subject to the limitations and restrictions imposed by copyright laws as described below.

It is against the law to copy, reproduce or transmit including without limitation, electronic transmission over any network any part of the manual or program except as permitted by the Copyright Act of the United States, Title 17, United States Code. Under the law copying includes translation into another language or format. However, you are permitted by law to write the contents of the program into the machine memory of your computer so that the program may be executed provided, however, that in no event shall two users use the program at the same time, as in shared-disk systems. You are also permitted by law to make working copies of the program, solely for your own use, subject to the following restrictions. (1) Working copies must be treated in the same way as the original copy; (2) If you ever sell, lend or give away the original copy of the program, all working copies must also be sold, lent or given to the same person, or destroyed; (3) No copy (original or working) may be used while any other copy (original or working) is in use. If you make working copies of the program you should place on them the copyright notice that is on the original copy of the program.

The above is not an inclusive statement of the restriction imposed on you under the Copyright Act. For a complete statement of the restrictions imposed under the copyright laws of the United States of America see Title 17, United States Code.

With your convenience in mind, BuildDisc has been supplied without copy protection. Remember that our continued support of this software clies upon your respect for its creator's efforts.

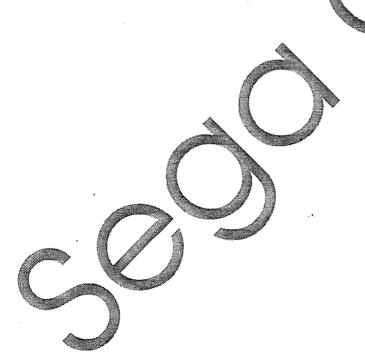
The following are trademarks of ICOM Simulations, Inc.: CTrac, ICOM Simulations and the ICOM Simulations, Inc. logo

The following is a trademark of Bell Laboratories: UNIX

ICOM Simulations, Inc. 648 South Wheeling Road Wheeling, IL 60090

Chapter 1 Introduction to BuildDisc

- 1.1 About BuildDisc
- 1.2 Compact Disc Standards
- 1.3 Running BuildDisc
 - 1.3.1 Command Line Options



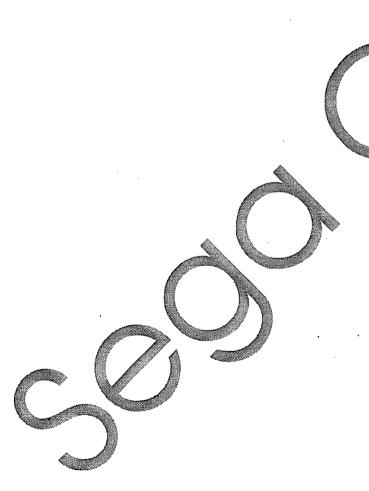
PROPERTY OF SEGA

1.1 About BuildDisc

BuildDisc is a tool which is used to create entire CD disc images for the CTrac emulation system. It combines all of the track images, along with the proper subcode information, into one composite disc image. BuildDisc operates at a relatively low level. It requires its user to have a fair knowledge of how CD's are put together. Because of this, BuildDisc can be complex to use, however, the tradeoff is that BuildDisc will allow its users almost total control over the final image produced.

The layout, format and contents of the final disc image are described in a control file with a series of commands. BuildDisc takes the control file as input and outputs a file that is the disc image. Other tools, such as the CTrac CD-ROM Emulator, can then make use of the image.

This documentation assumes a general understanding of the structure of a CD ROM disc. For more information on CD-ROM, refer to the N.V. PHILIPS CD Red Book or Yellow Book standards and the International Standard for CD-ROM Information Processing booklet (reference number ISO 9660: 1988 (E))



1.2 Compact Disc Standards

Although it is beyond the scope of this document to describe the entire structure of a compact disc, a few relevant details should be touched upon. Again, refer to the N.V. PHILIPS CD Red Book or Yellow Book standards and the International Standard for CD-ROM Information Processing booklet (reference number ISO 9660: 1988 (E)) for further information.

Data contained on compact discs are layed out in tracks. There may be up to 99 tracks on any given CD. With certain restrictions, each track may contain an arbitrary amount of data. However, the total amount of data on all tracks can not exceed the total amount of data allowed on a CD. The data within any track is grouped into frames. One frame corresponds to 1/75th of a second of data. There are 2352 bytes of data in every frame.

There are two dedicated tracks on a CD, the "leadin" and "leadout" tracks. The leadin track is always at the start of the CD and contains the table of contents (TOC) information for the CD. The table of contents is stored in the Q subcode area of the leadin track. The leadin is also referred to as the "TOC." The table of contents describes the number of tracks in use, the type of each track and where each track starts on the disc. The leadout track is always at the end of the disc and is used to tell CD drives that they have reached the end of the disc.

The tracks of a CD are either formatted as data (CD-ROM) or as audio (CD-DA). If a track is formatted as data, it is further divided into one of three modes, Mode 0, 1, or 2. All data tracks allocate 16 bytes per frame for sync and header information. The remaining bytes of every frame of Mode 0 tracks are all 0's. Mode 1 tracks contain 2048 bytes of data in every frame, using the remaining bytes for error detection and correction. Mode 2 tracks use all of the remaining space in each frame to data, storing 2336 bytes per frame.

Compact discs have a second data channel, called the subcode channel. It is stored on the CD "along side" of the data channel and is read out at the same time. There are 98 bytes of subcode for every frame on the CD. The subcode information is further broken down into 8 channels of 12 bytes each. These 8 channels are named P, Q, R, S, T, U, V and W. Channels P and Q are defined by the *PHILIPS CD Red Book* and *Yellow Book* standards. CD+G data and CD+Mid data are stored in channels R through W.

Based on information given in the control file, BuildDisc generates the subcode P and Q channels for all tracks in the disc image. During leadin, the Q channel contains the table of contents information. BuildDisc will allow the specification of the R-W channels via the subcsource command. For more information on the Subcsource command, see section 2.3.4 Track Commands.

1.3 Running BuildDisc

BuildDisc takes as input a control file that defines how the data is to be arranged in the image. The control file further specifies all source files that contain the actual data to be placed in the image. When BuildDisc is executed, it parses through the control file and creates, in memory, all of the data structures needed to build the disc image. The disc image is not created, however, until the disc definition in the control file has been completely read in and validated. Once validated, BuildDisc creates the disc image, writing its contents into an output file.

To execute BuildDisc, use the following command format:

BuildDisc [options] controlFileName [options]

On systems that do not allow 9 character file names, use the command to execute BuildDisc. The invocation command, the control file name and any command line options must be separated be at least one space.

1.3.1 Command Line Options

All options are case sensitive. If an option is specified incorrectly, a listing of the options and a brief description of each will appear at the command prompt. At least one space must be used between a command line option and its parameters. If multiple options are specified, they must be separated by at least one space. Extra spaces are ignored.

-b < numFrames>

Use a buffer of the number of frames specified when building the disc image. Each frame requires 2450 bytes of memory. If the -b command is not used, 20 frames are used by default. Setting the number of frames higher improves build time but uses more memory.

-d <variable> <value>

Defines the variable to the value specified. This option is useful for importing variable values into the control file. See also section 2.3.1 Global Commands for the Define command.

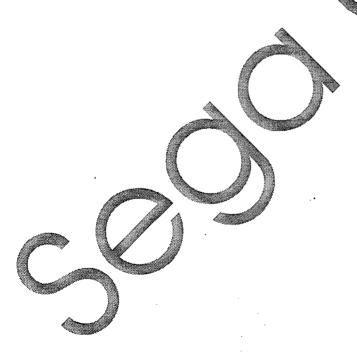
Prints diagnostic messages to the standard error device. This option is used mainly as a debugging tool and is not needed for normal operation.

Do not report warning messages. For a list of the warning messages that are generated by BuildDisc, see Appendix B Warning and Error Messages.

4

Chapter 2 Control Files

- 2.1 Command and File Format
- 2.2 An Example Control File
- 2.3 Control File Commands
 - 2.3.1 Global Commands
 - 2.3.2 Outermost Level Command
 - 2.3.3 Disc Commands
 - 2.3.4 Track Commands



2.1 Command and File Format

The control file provides the means by which the disc contents and layout are described. The disc type is described first followed by a description of all the tracks on the disc in succession. For each track, the data and subcode can be defined.

Command Format

All lines in the control file have the same structure. There must be at least one space between the control file command, the first parameter and any subsequent parameters. Extra spaces are ignored. Commands and their parameters are not case sensitive.

command [parameter1] [parameter2] [parameter3]...

omment

A comment begins immediately after a semi-colon (;). For a complete list and explanation of other BuildDisc special characters, see Appendix A Parsing Algorithm.

File Format

The control file for BuildDisc takes the following general format.

Disc discType outputFileName

Commands that further define the disc are specified here.

LeadIn trackType

Commands that define the leadin track go here.

EndTrack

Track trackType

Commands that define track 1 go here. Tracks are placed on the disc in the order that they appear in the control file. Up to 99 tracks may be defined.

EndTrack

Track trackType

Commands that define track n go here.

EndTrack

LeadOut trackType

Commands that define the leadout track go here.

EndTrack

Endiso

CONFIDENTIAL

2.2 An Example Control File

The following is a sample control file for building a CD-ROM disc image. The resulting image will be a disc that contains one Mode 1 track and two audio tracks, one of which contains an index.

```
Echo
      "building disc image"
Define
            TWOSEC
                        150
                                           ; num of frames in 2 se
Define
            SRCPATH
                        "C:\myData\"
                                           ; path to data for disc
Disc CDROM output.dsc
                                     ; a CD-ROM dis
CatalogNumber 0123456789012
                                     ; 13 digits of catalog
                                                             umber
LeadIn MODE1
                         ; information about the leadin area
Empty 4500
                         ; length of the leadin area (in frames)
PostGap 150
                         ; need postcap for CD-ROM leadin
EndTrack
Track MODE1
                               ; fight track is Mode 1
Pause [TWOSEC]
                               wait 2 seconds before data begins
Source [PATH] myTrk
                              ; file that contains data for track
PostGap 150
                               ; postgap for switching track modes
EndTrack
                               ; this track is done
Track AUDIO
      ISUSA9001234
                              ; place ISRC code into subcode
Source [PATH] Mozart
                              ; place this audio on this track
Track AUDIO
Source [PATH]Bach.1
                              ; part one of Bach music
Index
                              ; place index in here
Source [PATH]Bach.2
                              ; part two
EndTrack
```

LeadOut AUDIO

Empty 500

; amount of time to use for leadout

EndTrack

EndDisc



8

BuildDisc Reference Manual

2.3 Control File Commands

There are four types of commands that are valid in a BuildDisc control file. They are described briefly below. If a command is used incorrectly in the control file, a warning or an error message will be printed to the standard error device. If this occurs, BuildDisc will continue parsing through the control file until it reaches the end. It will continue printing all error or warning messages as necessary. If any error messages are generated, the disc image is not created. For a list of the warning and error messages, that are generated by BuildDisc, see Appendix B Warning and Error Messages.

Global Commands

Valid at any point in the control file.

Outermost Level Commands

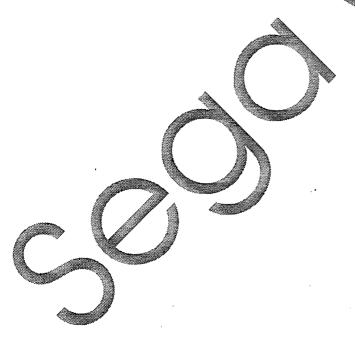
Valid before the beginning of a disc definition only

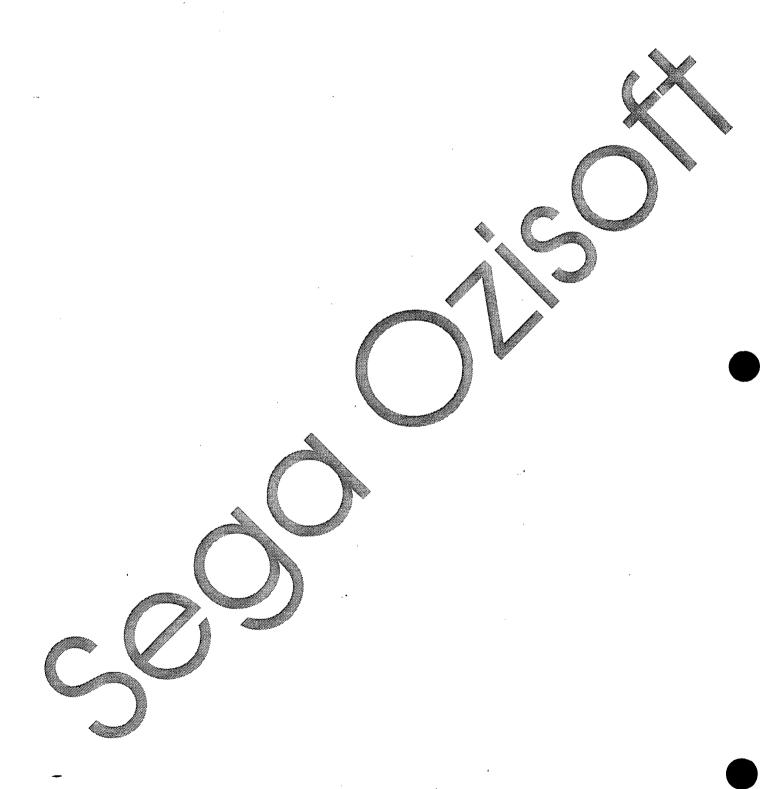
Disc Commands

Valid within a disc definition only.

Track Commands

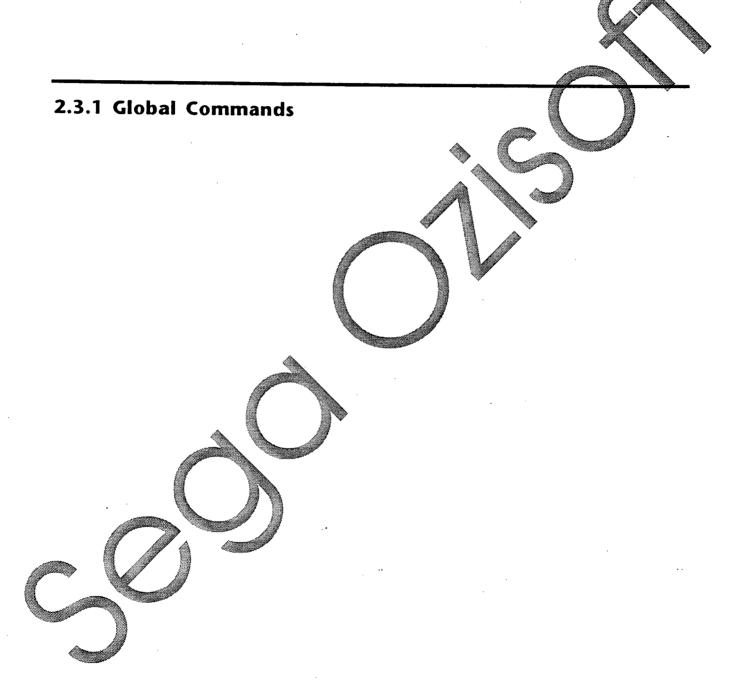
Valid within a track definition only.





10

BuildDisc Reference Manual



Define

Syntax

Define <variable> <value> [<variable> <value>] [...]

Example

Define path C: IMAGES

Description

Defines one or more variables to values. The variable and value can be any arbitrary ASCII strings. The variables and values are sensitive.

- Define can be useful for aliasing values that would be repeated often in a control file or for importing values into the build. For more information, see section 1.3.1 Command Line Options for the -d option.
- To remove (undefine) a variable that was previously defined, define it to the empty string (****). For example, the command
 Define path "" undefines the variable in the example above.

Defaults

When BuildDisc starts, the following variables are predefined:

program

version

edition

second

minute

hour

day

month

ywai

weekday

yearday

name of the executing program

current version number

- current edition

Me second of the minute

- the minute of the hour

- the hour of the day

- the day of the month

- the month of the year

- the year

- the day of the week

- the day of the year

Appendix A Parsing Algorithm for an explanation of the notation used for accessing variables and for declaring strings.

ee also

CONFIDENTIAL

Echo

Syntax

Echo [<argument1>] [<argument2>] [...]

Example

Echo this is a test

Description

Reports all arguments to the console.

- Echo is normally used to display variables or messages while the control file is being processed.
- Using the ECHO command without an argument results in a carriage return being echoed.
- Δ Note: Due to buffering of data sent to the console, it is possible that the output of ECHO may not appear until the BuildDisc program has stopped running



Include

Syntax

Include <controlFileName1> [<controlFileName2>] [...]

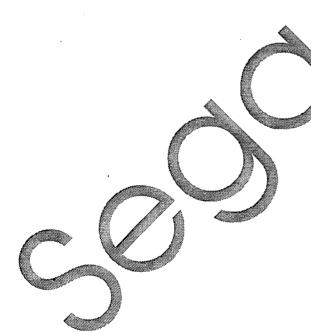
Example

Include MYFILE.CTL

Description

Includes the contents of each of the listed control files into the build at the point where the INCLUDE statement occurred. Each of the listed files is included in the order in which it appears in the list

- Include files can be nested.
- To prevent recursive nesting of include files that could cause BuildDisc to hang up while parsing the control file, there is an imposed limit of 25 nested levels deep. If the control file attempts to nest include files to a level deeper than this, an error will be generated.



CONFIDENTIAL

ShowDefines

Syntax

ShowDefines

Example

ShowDefines

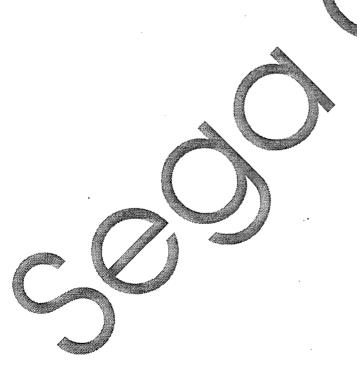
Description

Reports all defined variables and their values to the console.

- This command can be useful when trying to determine what a
 variable is getting set to in a complex expression or to show the
 current state of variables from anywhere within the control file.
- A Note: Due to buffering of data sent to the console, it is possible that the output of SHOWDEFINES may not appear until the BuildDisc program has stopped running.

See also

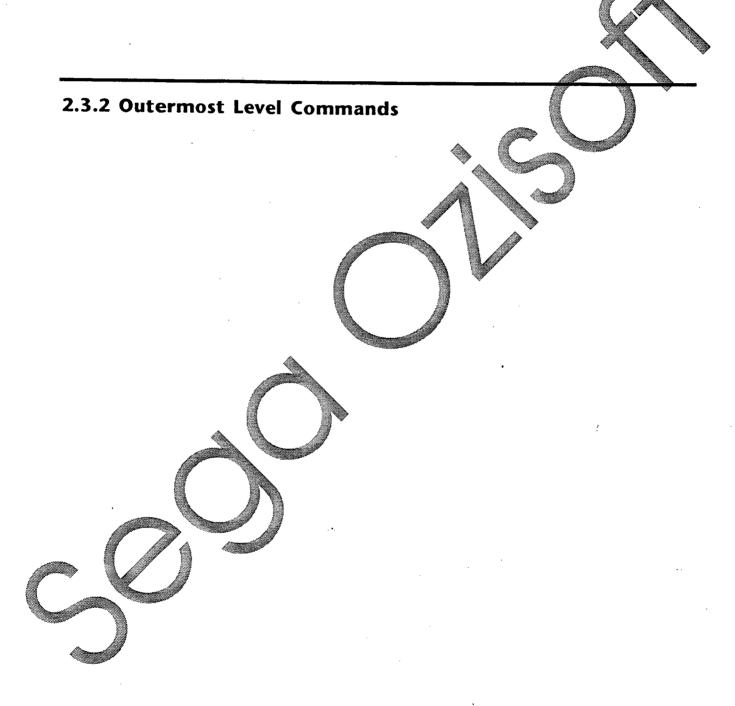
Appendix A Parsing Algorithm for an explanation of the notation used for accessing variables and for declaring strings.





16

BuildDisc Reference Manual



BuildDisc Reference Manual

Disc

Syntax

Disc <discType> <outputFileName>

Example

Disc CDROM output.dsc

Description

Begins the definition of a disc of the type specified. The disc image is

written to the file specified. Valid disc types are as follows:

CDDA, CDAudio or Audio - audio discs

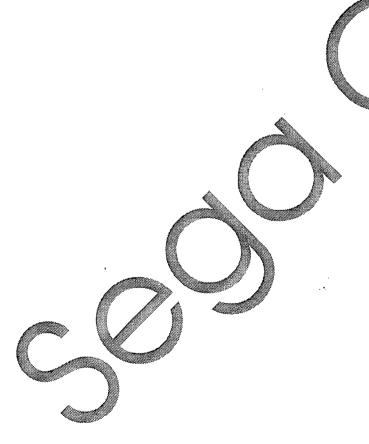
CDROM or ROM

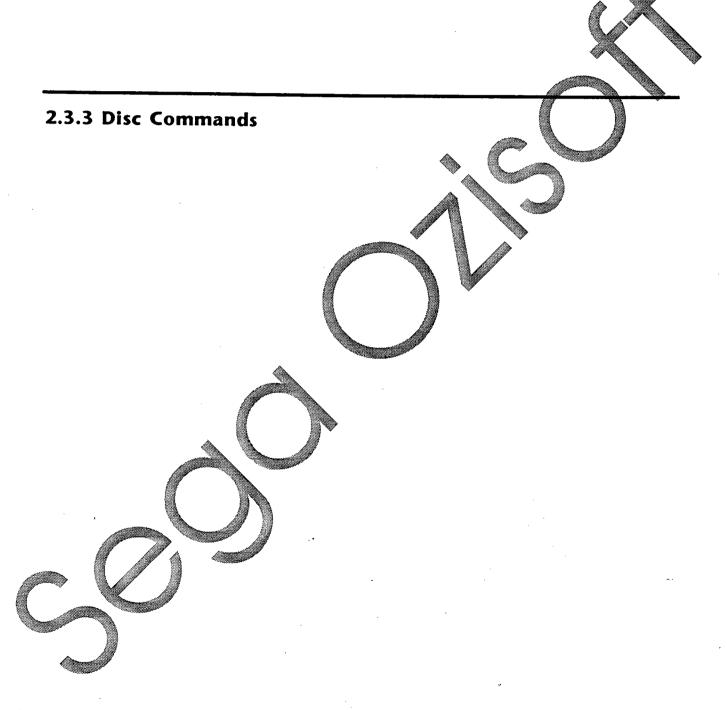
- discs with Mode 0, Mode 1 or Mode 2

CDI

- CDI discs

Note: Discs of type CDROM or ROM may also contain audio Δ tracks.





CatalogNumber

Syntax

CatalogNumber <theNumber>

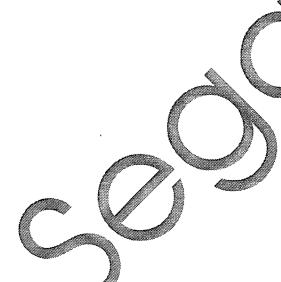
Example

CatalogNumber 01234

Description

The compact disc standard allows for the Q subcode channel to contain a catalog number that describes the disc. This command allows the catalog number for this disc to be specified. If a catalog number is not specified for a disc, it will not contain one.

 The number parameter can be up to 13 ASCII digits. If it is less than 13 digits, the leading digits are padded with 0s.



EndDisc

Syntax

EndDisc

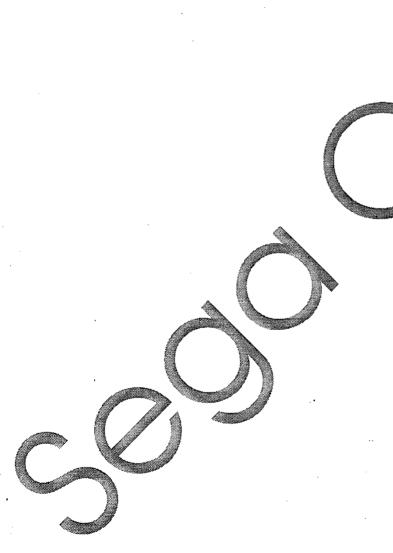
Example

EndDisc

Description

This marks the end of the current disc definition. When this command is encountered and no errors have occurred, the actual

disc build takes place.



LeadIn

Syntax

LeadIn <trackType>

Alt. Syntax

TOC <trackType>

Example

LeadIn CDDA

Description

Specifies that the Leadin track is beginning for the following track types are valid:

CDDA, CDAudio or Audio

- audio track

Mode0, Data0, ROM0 or CDROM0

- Mode @ data track

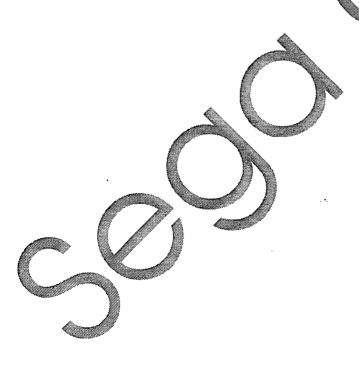
Model, Datal, ROM1 or CDROM1

- Mode I data track

Mode2, Data2, ROM2 or CDROM2

Mode 2 data track

Δ Note: The Leadin track contains the table of contents (TOC) information for the disc. Although the Red Book does not specify a minimum length for the Leadin track, 500 frames is recommended. Actual CD's usually contain upwards of 1500 frames of Leadin.



LeadOut

Syntax

LeadOut <trackType>

Example

LeadOut MODE1

Description

Specifies that the leadout track is beginning for this disc. The following track types are valid:

CDDA, CDAudio or Audio

- audio track

Mode0, Data0, ROM0 or CDROM0

- Mode 0 data track

Mode1, Data1, ROM1 or CDROM1

- Mode I data track

Mode2, Data2, ROM2 or CDROM2

- Mode 2 data trac

A Note: Although the Red Book does not define the minimum length of the Leadout track, 500 frames of "empty" data is recommended. For more information, see the EMPTY command.



MapFile

Syntax

MapFile <fileName>

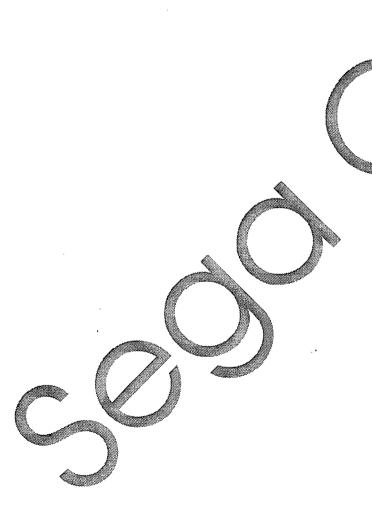
Example

MapFile disc.map

Description

Places a "map" of the disc image into the file specified. The ASCII map file describes each track in the disc image. The track information includes the data length of each track and when me time, each track begins. PREGAP, POSTGAP and PAUSE parameters for each track are also listed.

 The map file output can be used to determine where data on a track begins or how long the created disc image is. It can also be used with tools that modify and maintain disc image files



Track

Syntax

Track <trackType>

Example

Track MODE2

Description

Specifies that a new track is beginning. The new track will have a track number that is 1 greater than that of the previous track. If there were no previous tracks, the track number will be 1. The track numbers do not include the Leadin and Leadout tracks. The following track types are valid:

CDDA, CDAudio or Audio

- audio track

Mode0, Data0, ROM0 or CDROM0

- Mode 0 data trac

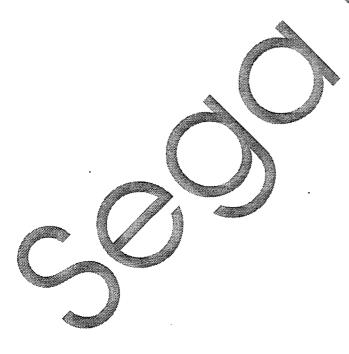
Mode1, Data1, ROM1 or CDROM1

- Mode I data track

Mode2, Data2, ROM2 or CDROM2

Mode 2 data tracl

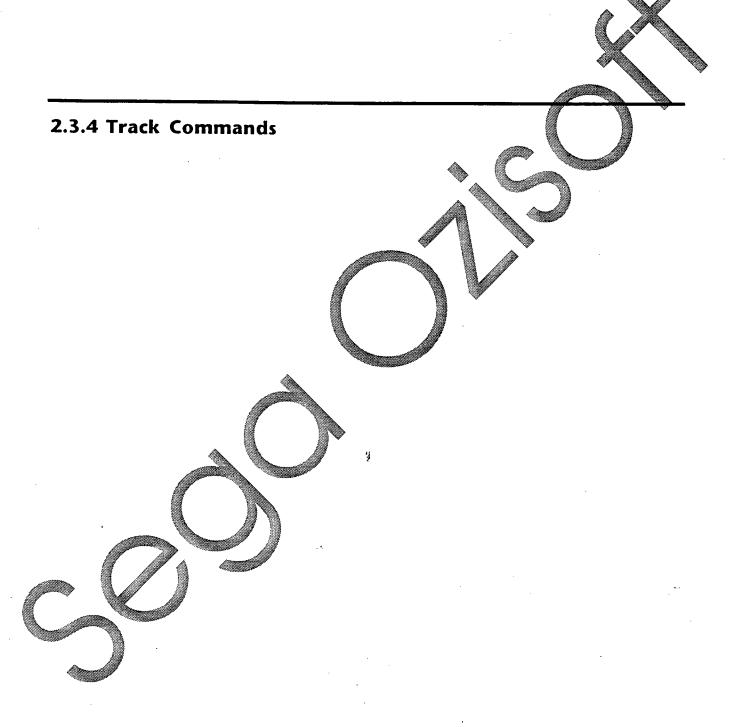
- Up to 99 tracks (not including the Deadin and Leadout tracks) may be specified.
- The command is not valid unless the command LEADIN has been previously defined.





26

BuildDisc Reference Manual



BuildDisc Reference Manual

Channels

Syntax

Channels <2 or 4>

Example

Channels 4

Description

If the track is of type CDDA, this command specifies if it has 2 or 4

channels (this is contained within the Q subcode information).

Default

When unspecified, the default is 2 channels.



Copy

Syntax

Copy <boolean>

Example

Copy true

Description

Indicates the copy protect state of the audio data for the current track. If this is set to true, digital copying will be allowed.

 The boolean parameter can be either true, on, res, or 1 for TRUE or false, off, no, or 0 for FALSE.

Default

FALSE



Empty

Syntax

Empty < numFrames>

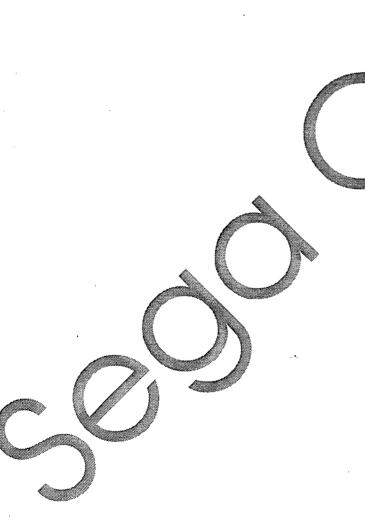
Example

Empty 200

Description

For the number of frames specified, 0's will be written into the output when this command is encountered. One frame corresponds to 1/75th of a second of data. There are 2352 bytes of data in every frame.

 This command can be useful when defining the Leadin and Leadout tracks, since they usually do not contain data. The EMPTY command can be used to define their contents.



EndTrack

Syntax

EndTrack

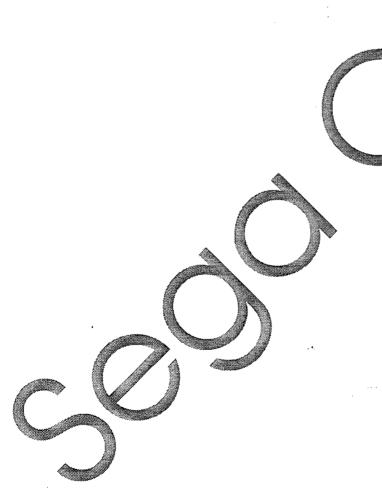
Example

EndTrack

Description

Marks the end of the current track definition. All track definitions must end with this command, including the Leadin and Leadout

tracks.



Index

Syntax

Index

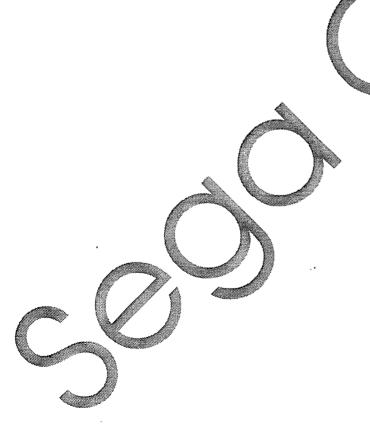
Example

Index

Description

Places an index into the track at the point the command is encountered. When a track is defined, it starts at index 1. Each index command increases the number by 1.

- Up to 99 indices may be specified for a single track.
- Indices must be separated by at least 1 frame of data.
- Warning: The INDEX command cannot be used within the following track type definitions: Leadin, LeadOut and CDROM.



ISRC

Syntax

ISRC <theNumber>

Example

ISRC USICM9100001

Description

Defines an ISRC number to be placed in the Q subcode for the current track. The ISRC number looks as follows: CCOOOYYS SSS where CC is the country code (2 uppercase letters or digits), OOO is the owner code (3 uppercase letters or digits), YY is the year of the recording (2 digits), and SSSSS is the serial number of the recording (5 digits).

- If an ISRC number is not specified for a track, it will not contain one.
- Warning: The ISRC command cannot be used within the following track type definitions: leadin and LeadOut.



Pause

Syntax

Pause < numFrames>

Example

Pause 150

Description

Specifies the number of frames to pause at the start of the track. One frame corresponds to 1/75th of a second of data. 150 frames (2 seconds) is a standard pause length. All frames of pause contain vis.

A Note: There are instances when a pause is required at the start of a track. When the required pause is not present, BuildDisc will generate an error message indicating the point in the control file where the PAUSE command must be defined and the minimum number of frames required.



PostGap

Syntax

PostGap < numFrames>

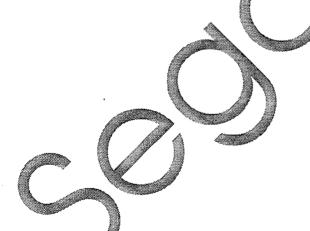
Example

PostGap 150

Description

Specifies the number of frames of postgap for the current track. One frame corresponds to 1/75th of a second of data. All frames of postgap contain 0's.

Note: There are instances when a postgap is required at the end of a track. If the required postgap is not present, BuildDisc will generate an error message indicating the point in the control file where the POSTGAP command must be defined and the minimum number of frames required.



PreEmphasis

Syntax

PreEmphasis <boolean>

Alt. Syntax

Pre-emphasis <boolean>

Example

PreEmphasis on

Description

If the current track type is CDDA, this command indicates if the preemphasis bit in the Q subcode channel should be turned on

• The boolean parameter can be either true, on, wes, or her TRUE or false, off, no, or 0 for FALSE.

Default

OFF



PreGap

Syntax

PreGap < numFrames>

Example

PreGap 150

Description

Specifies the number of frames of pregap information at the start of the current track. One frame corresponds to 1/75th of a second of data. All frames of pregap contain 0's.

Note: There are instances when a pregap is required at the beginning of a track. If the required pregap is not present, BuildDisc will generate an error message indicating the point in the control file where the PRECAP command must be defined and the minimum number of frames required.



Source

Syntax

Source <theFile>

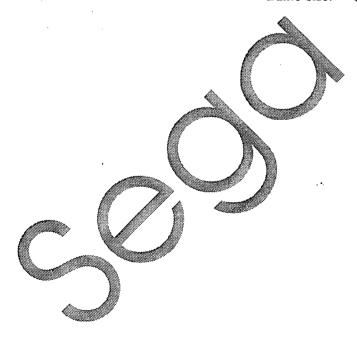
Example

Source myData.asm

Description

Indicates that the data in the file specified should be placed into the disc image at this point. Multiple SOURCE commands are allowed within any track. The source files are individually padded to being each to a multiple of the frame size as described below.

- If the data is CDDA, it should be in the form of 16 bits left channel, 16 bits right channel, until the end of the file. The byte ordering is low-high. If the file is not an integral number of frames long (each audio frame is 2352 bytes), the output will be padded with 0's (audio silence) to bring it to an integral number of frames.
- If the data is Mode 1, the file should just contain the raw binary data. BuildDisc will create the EDC/EC and all needed sync and header information. If the file is not an even multiple of the 2048 byte Mode 1 frame size, it will be padded with 0's to bring it to the nearest frame.
- If the data is Mode2, it will be padded to the 2336 byte Mode 2 frame size.



SubcEmpty

Syntax

SubcEmpty

Example

SubcEmpty

Description

At the point this command is encountered in the control file 0's are

output to the R through W channels of the subcode. This is used to

terminate a SubcSource command.

See also

SubcSource



SubcSource

Syntax

SubcSource <fileName>

Example

SubcSource midi.dat

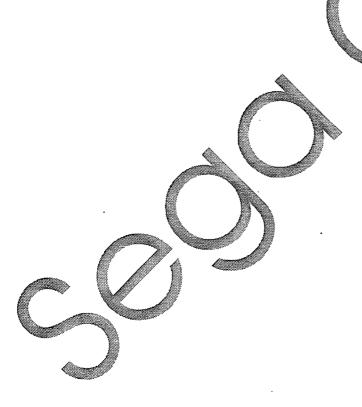
Description

This command is primarily used to place CD+G or CD+Midi data into the subcode of a track. Data from the file specified will be placed in the R through W channels of the subcode in the disc image at the location where the SUBCSOURCE command is located. If the data in the file runs out before the end of the disc, or before another SUBCSOURCE command is encountered, 0's are written into the R—W channels. Each byte of data in the file specified will contain 1 byte of subcode data, with bit 2 = R, bit 3 = S, ... bit T = W. Bits 0 and 1 would be P and Q, but are ignored.

 The SUBCSOURCE command stays in effect across track boundaries.

See also

SubcEmpty





Appendix A Parsing Algorithm
Appendix B Warning and Error Messages



Appendix A Parsing Algorithm

BuildDisc parses through the control file much like a UNIX[™] shell parses input. It recognizes certain characters as special and performs various substitutions. Below is a list of the special characters and their meaning.

<u>Characters</u>	<u>Usage</u>
white space	Separates parameters.
return	Separates commands.
11	Single quotes protect the characters inside from substitution and groups the characters together so they are treated as one argument. Surround single quotes with double quotes to include them in an argument.
11 1 1	Double quotes group all characters inside as a single entity. Substitutions still occur and the results are grouped. Surround double quotes with single quotes to include them in an argument.
{}	Braces force a recursive substitution on the variable/variables inside.
[]	Brackets force a single substitution on the variable/variables inside.
;	Semi-colons begin comments.
Examples	
"{A}"	-will produce 1 argument that is the replacement for the variable A
{A}	- will produce as many arguments as the replacement for A requires
{"A"} {A B}	- is the same as {A} sis the same as {A}{B}
("A B")	requests a substitution of the variable A B
"("A B"}"	- will produce 1 argument that replaces A B (quotes nest through other characters)
"}\"A B'}"	- will do the same as "{"A B"}"
(14))	- is C if $\{A\}$ = B and $\{B\}$ = C
'{A}'	- will result in the argument {A} being passed

- if A is defined as '{B}' and B is defined as "TEST", then {A} yields TEST, while

[A] yields {B}

"a "rip in" time" - will produce 2 arguments a rip and in time

- will yield 1 argument that is an empty string

- will yield 1 argument that is an empty string

"don't do that" - will produce one argument don't do that

Appendix B Warning and Error Messages

BuildDisc can potentially output a great variety of warning and error messages. Most of these are self explanatory and are not further detailed here. Below are listed a number of messages for which some additional information was thought to be helpful. In this section, whenever the characters STR are shown, they refer to a string that is filled in by the program at run time. This is usually that name of a command or file.

Warning Messages

Your lead-in track is somewhat short, (more than STR frames would be nice)

The *Red Book* does not specify how many frames of data are required in the leady area of a disc. This message indicates that the amount specified is too low for most purposes.

```
Previous catalog number replaced 
Previous ISRC number replaced
```

The catalog and ISRC numbers can be specified more than once. The last one specified will be used in the final disc image. This is simply to warn you that you have repeated the specification of the catalog or ISRC, in case it was done inadvertently.

```
Source data in file STR may violate mode 0 data track structure
```

The Yellow Book standard specifies that a Mode 0 data track will have only data bytes containing 0's in the track. BuildDisc allows you to place any data in a Mode 0 track, but will issue this warning when doing so.

Error Messages

```
Could not allocate memory for line input buffer
Could not allocate memory for line output buffer
Could not create copy of file name operand
Could not create lead in information data structure
Could not create new track
Could not create new track definition structure
Could not create new track for STR
Failed to initialize
Failed to initialize file system
Unable to create data structures needed for STR
```

These messages will usually be accompanied by another message indicating that not enough memory was available to perform the required action.

BuildDisc Reference Manual

```
Failed to allocate output buffer for disc build (check the -b option)

Failed to allocate subcode buffer for disc build (check the -b option)
```

The command line option -b allows the user to specify the number of frames to buffer during the build. If too many frames are specified, it might not be possible for BuildDisc to obtain enough memory from the Operating System (OS) to satisfy the request. If you receive either of these messages, reduce the number of frame buffers requested.

```
Could not open control file STR1. OS Reports: STR2
Could not open output file STR1. OS Reports: STR2
Failed to open file STR1. OS Reports: STR2
Failed to open lead-in source file STR1. OS Reports: STR2
Failed to open lead-out source file STR1. OS Reports: STR2
Failed to open subcode source file STR1. OS Reports: STR2
Failed to open track source file STR1. OS Reports: STR2
Failed to read lead-in source file STR1. OS Reports:
Failed to read lead-out source file STR1. OS Reports
Failed to read subcode source file STR1. OS Reports: STR
Failed to read track source file STR1 OS Reports: STR2
Failed to write lead-in blocks to output The STR1,
                                                   S Reports: STR2
Failed to write lead-out blocks to output file STM1. OS Reports: STR2
Failed to write track blocks to output file TR1. OS Reports: STR2
Failed to write track lead-out to output fare STR1. OS Reports: STR2
```

All of these messages indicate that some type of I/O error occurred while attempting to perform the function noted. The phrase after "OS Reports: " will show what type of error the operating system reported to BuildDisc. Consult your OS manual for more information.

Indices must be separates by at least 1 frame

Indices (generated by an INDEX command) must be separated by at least 1 frame of data. If you had two consecutive INDEX commands, this message is generated.

Track is too short, must be at least 300 frames

The *Red Book* forces tracks to be at least 300 frames long. If the track definition is shorter, this plessage will be generated. The track can be lengthened by placing an EMPTY command at the end of it.





46

BuildDisc Reference Manual

Index

Symbols and Operators

' ', 42

" ", 42

{}, 42

[], 42

,, 42

\blacksquare A

Aliasing variables, 12 Audio, 18, 22-23, 25

\blacksquare B

Brackets

square [], 42 curly {}, 42

\blacksquare C

CatalogNumber

disc command, 20

CD+G data, 40

CD+MIDI data, 40

CDAudio, 18, 22-23, 25

CDDA, 3, 18, 22-23, 25, 28, 36, 38

CDI, 18

CDROM, 18

CDROM0, 22-23, 25

CDROM1, 22-23, 25

CDROM2, 22-23, 25

Channels

track command, 28

Comments, 42

Control file, 6

Copy

Frack command, 29

\blacksquare D

Data0, 22-23, 25

Data1, 22-23, 25

Data2, 22-23, 25

Day variable, 12

Define

global command, 12

Disc

outermost level command, 18

Double quotes

" ", 42

\blacksquare E

Echo

global command,

EDC/ECC, 38

Edition variable, 12

Empty

track command, 30

EndDisc

disc command, 21

EndTrack

track command, 31

Error messages, 44

H

Hour variable, 12

Include

global command, 14

Index

track command, 32

ISRC

track command, 33

LeadIn

disc command, 22

LeadOut

disc command, 23

\blacksquare M

MapFile

disc command, 24

Minute variable, 12

Mode0, 22-23, 25

#161

Mode1, 22-23, 25, 38 Mode2, 22-23, 25, 38 Month variable, 12

$\blacksquare P$

Parsing algorithm, 42
Pause
track command, 34
PostGap
track command, 35
PreEmphasis
track command, 36
PreGap
track command, 37
Program variable, 12

■ Q

Quotes single, 42 double, 42

\blacksquare R

Return, 42 ROM, 18 ROM0, 22-23, 25 ROM1, 22-23, 25 ROM2, 22-23, 25

Second variable, 12

Semi-colon
;, 42
ShowDefines
global command, 15
Single quotes
'', 42
Source

source track command, 38 Space, 42 Subclimpty track command, 39 SubcSource track command, 40

\blacksquare T

TOC disc command, 22 Track disc command, 2

Undefining variables, 12

$\blacksquare V$

Version variable, 12

Warning messages, 44 Weekday variable, 12

JY

Year variable, 12 Yearday variable, 12

