CTrac™ Builder
for
Sega® Genesis®

ICOM SIMULATIONS, INC.

User Guide

1. Introduction

CTrac Builder is a set of two software programs designed to allow the user to create CD-ROM images that are compatible with the Sega Genesis game machine and that can be used in conjunction with CTrac Emulator to create, test, and debug CD-ROM products. The two software programs are BuildTrack, which creates ISO 9660 track images, and BuildDisc, which creates the actual CD-ROM disk image.

2. How CTrac Builder works.

The Builder system works in two steps. First the user builds all of the necessary tracks with a BuildTrack appropriate for the type of track he is creating. Since the Towns computer only uses a combination of ISO 9660 tracks and CD-DA tracks, there is only one BuildTrack. Since the format required for a CD-DA track is so simple (one 16 bit word of left channel sound data, followed by a 16 bit word of right channel sound data), a CD-DA BuildTrack is not required. After the user has all of his tracks built, he uses BuildDisc to combine the tracks and create the disc image. Since both the track images and the disc images are standard DOS data files, they can be created on any standard DOS volume. Inputs to BuildDisc are a control file that defines how the data is to be arranged in the image, and source files that specify the actual data to be used.

3. Running BuildTrack and BuildDisc.

Both programs, BuildTrack and BuildDisc are run in the same manner They take as inputs a control file that defines how the data is to be arranged in the image, and source files that specify the actual data to be placed in the image. More than one output image file can be created by a single BuildTrack or BuildDisc control file, although it is generally recommended that each control file should build only one image.

3.1 Running BuildTrack.

To execute BuildTrack, the following command line is used:

buildtrack [options] ControlFileName [options]

The available options are:

-d variable value	Define a variable to the given value.
-b numSectors	Use a buffer of numSectors when building (default = 64). Setting this higher will improve build time.
-w	Do not report warnings.
-p	Print diagnostic messages to the standard error device.
-l	Expect ISO level 1 interchange compatible file names. Warn if a name violates this convention.

When BuildTrack is executed, it will parse through the given control file, and create (in memory) all of the data structures needed to actually build the ISO image. No output is created, however, until the ISO 9660 definition in the control file has been completely read in, and validated. BuildTrack contains a complete knowledge of what it means for an ISO 9660 image to be valid, and will not allow an invalid ISO

9660 image to be created without generating a warning. BuildTrack creates the entire volume structure for an ISO 9660 volume.

ISO 9660 volumes reside on a single track of a CD (usually, track 1). BuildTrack is used to create the entire ISO volume image that would be contained on this track.

ISO 9660 volumes contain a hierarchical directory system, as well as various structures (called descriptors) that define the volume. See the ISO 9660 specification for more information.

The control file that is used as the main input to BuildTrack defines the layout of these structures, as well as the layout of the hierarchical directory system.

Volume descriptors, and the hierarchies that are contained within are written in the order that they are read from the control file definition. Files and directories are written to the output image in the order that they appear in the control file.

Descriptors,directory hierarchies, directories, and files are specified in a hierarchical manner, each only being specified within its applicable scope. (e.g. a directory can only be specified within a hierarchy, or another directory, not at the outside level)

File names are checked for consistency with the ISO 9660 format, and warnings are generated if the file name does not meet the spec. For the most part, BuildTrack will allow non-standard fields within ISO structures as long as there is no ambiguity, or problem fitting the data into the field. Warnings will always be generated when a field that is specified does not meet the standard.

All application and system use fields whose contents are not specified by ISO can be filled from source files during the build process.

3.2 Running BuildDisc.

To execute BuildDisc, the following command line is used:

builddisc [options] ControlFileName [options]

The available options are:

-d variable value	Define a variable to the given value.
-b numBlocks	Use a buffer of numBlocks when building (default = 20). Setting this higher will improve build time.
-w	Do not report warnings.
-p	Print diagnostic messages to the standard error device.

When BuildDisc is executed, it will parse through the given control file, and create (in memory) all of the data structures needed to actually build the disc image. No output is created, however, until the disc definition in the control file has been completely read in, and validated. BuildDisc contains a complete knowledge of what it means for a disc image to be valid, and will not allow an invalid disc image to be created. I t will apply EDC/ECC, sync, and header information as needed, and it generates the data for the subcode P and Q channels.

4. About Control Files.

A control file consists of command tokens followed by zero or more parameters. These tokens are arranged within certain scopes, which define at what level of the hierarchy of information used to define a disc or a track that they are recognized. All of the valid tokens and their scopes will be defined below.

Many of the command tokens and their parameters defined below have a specific meaning and range of values when considered in the context of the definition of a "Yellow Book" standard CD-ROM or the ISO 9660 data format. The complete definition of these standards is beyond the scope of this document. For further information, please refer to the Compact Disc Read Only Memory standards documentation from Philips International, B.V. and the ISO 9660 standard (reference number ISO 9660 : 1988 (E)) available from the International Organization for Standardization.

In the rest of this document, words in **BOLD** are command tokens, words in *italics* are supplemental comments, and **outlined** words represent a parameter that the user would supply for that token.

When executed, BuildDisc and BuildTrack parse through their 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 what they do.

Characters	Explanation
WhiteSpace	Used to separate parameters.
Return	Used to separate commands.
'...'	Single quotes protect the characters inside them from substitution, and group the characters together so they become one argument
"..."	Double quotes group all characters as a single entity. Substitutions still occur, and the results are grouped. Double quotes quote single quotes.
{...}	Braces force a recursive substitution on the variable/variables inside.
[...]	Brackets force a single substitution on the variable/variables inside.
;	Semicolons begin comments.

Some examples to help clarify things:

`"{A}"`	will produce 1 argument that is the replacement for `A`
`{A}`	will produce as many arguments as the replacement for `A` requires
`{"A"}`	is the same as `{A}`
`{A B}`	is the same as `{A}{B}`
`{"A B"}`	requests a substitution of 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"}"`
`{{A}}`	is C if {A} = B and {B} = C
`'{A}'`	will result in the argument {A} being passed
`{A}`	if A is defined as {B} and B is defined as "TEST", then
`[A]`	yields TEST, while
`"a "rip in" time"`	yields {B}
`""`	will produce 2 arguments a_rip and in_time
`''`	will yield 1 argument that is an empty string
`"don't do that"`	will yield 1 argument that is an empty string
	will produce one argument don't_do_that

5. Installing the Builder Software.

The Builder software is contained in a pair of DOS disk files called BuildTrack and BuildDisc. To install the programs, simply copy them to the hard disk on your MS-DOS computer. They can be placed in any convenient directory. The Builder programs and the images that they build do not have to be in the same directory, nor do the control files.

6. The BuildTrack control file.

The control file for BuildTrack takes the following general format:

```
Volume          volumeType      outputFileName
commands that further define the track volume go here

note that the order of the BootRecord, VolumePartition, PrimaryVolume, and
SupplementaryVolume can be changed, but they are written out to the disk in the
order in which they are defined.

BootRecord              ;There can be an arbitrary number of BootRecords
commands that define the bootrecord go here
EndBootRecord

VolumePartition         ;There can be an arbitrary number of VolumePartitions
commands that define the volumepartition go here
EndVolumePartition

PrimaryVolume           ;There can only be one primary volume
commands that further define the primary volume go here.

    Hierarchy           ;There can only be one Hierarchy per PrimaryVolume
    commands that further define the Hierarchy go here.

        Directory theName   ;There can be an arbitrary number of directories
        commands that further define the Directory go here
        other Directory commands can be nested inside of directories, to an ISO
        defined limit of 8 deep.  BuildTrack does not enforce this ISO limit, but will
        warn you if you exceed it.

            File theName [theVersion]
            commands that further define the File go here
            EndFile

            an arbitrary number of files can go here
            File commands can also be placed outside of directories (but inside
            the Hierarchy Command), both before, between, and after a
            Directory command.

        EndDirectory
    EndHierarchy
EndPrimaryVolume
```

SupplementaryVolume ;*There can an arbitrary number of supplementary volumes commands that further define the primary volume go here.*

The usage and layout of Hierarchy, Directory, and File commands inside of the supplementaryvolume follow the same rules as in the primaryvolume

EndSupplementaryVolume

EndVolume

6.1 Global Commands.

Globally recognized command tokens (those that have meaning anywhere in the control file):

Define	**variable**	**value**	Defines a variable to a value (variable and value can be any string). Define can be useful for aliasing values that would be repeated often in a control file, or for importing values into the build (See the -d command line option). When BuildTrack starts, some variables are pre-defined: program - Name of the executing program version - Current version number edition - Current edition second - the second of the minute minute - the minute of the hour hour - the hour of the day day - the day of the month month - the month of the year year - the year weekday - the day of the week yearday - the day of the year
ShowDefines			Shows all currently defined variables.
Echo	**theString**		Echoes the passed string to the standard output device (theString can be anything). Echo can be useful for debugging a control file, or just for status information.
Include	**theFile**		Include another control File at this point (theFile must be a valid path and file name for the machine that BuildTrack is running on).

6.2 Outermost Level Commands.

Command tokens that are active at the outermost level:

Volume	**volumeType**	**outputFile**	Specify that a volume of volumeType is being defined, and that the output should be written to outputFile. Currently the only volume type that BuildTrack knows is ISO9660.

6.3 Volume Commands.

Command tokens that are active within a volume:

SystemArea	theFile	This is used to specify a file that contains the source data that should be placed into the system area of this ISO 9660 volume. The system area of an ISO 9660 volume is made up of the first 16 sectors. If the file contains too little data to fill the first 16 sectors, it will be padded with 0's. If the file contains more data than will fit, an error will be generated.
PrimaryVolume		This marks the beginning of a primary volume descriptor record. There can be only one primary volume in an ISO 9660 image.
SupplementaryVolume		This marks the beginning of a supplementary volume descriptor record. There can be an arbitrary number of supplementary volumes in an ISO9660 image.
VolumePartition		This marks the beginning of a volume partition descriptor record. There can be an arbitrary number of volume partitions in an ISO 9660 image.
BootRecord		This marks the beginning of a boot record. There can be an arbitrary number of boot records in an ISO 9660 image.
EndVolume		This marks the end of a volume definition. At this point, if the definition of the volume is valid, the actual track image will be built.

6.4 Primary Volume Commands.

Command tokens that are active within a primary volume:

DescriptorWrites	numWrites	This is used to specify the number of times that the descriptor record for the current Primary volume will be written to the output. If this is not specified, it defaults to 1.
SystemIdentifier	theString	Use this to define the system identifier name for the current primary volume.
VolumeIdentifier	theString	Use this to define the volume identifier name for the current primary volume
LogicalBlockSize	theSize	Use this to define the logical block size for the volume. Allowed logical block sizes are: 512, 1024, and 2048.
LPath		When this token is seen, BuildTrack will write out the LPath descriptor. The position of this token in the control file defines the position of the LPath table in the output image. Only one LPath is allowed per primary volume, and it must be specified.
MPath		Same as LPath, except this writes out the MPath.
OptionalLPath		When this is seen, the optional L Path table is written. This does not need to be specified, and if it is not, it will be omitted from the output
OptionalMPath		Writes out optional M path table.
VolumeSetIdentifier	theString	Use this to specify the volume set identifier.

PublisherIdentifier	theString	Use this to specify the publisher identifier. If this string is preceded by an underscore (_) it is taken to be a file name on the root of the primary volume that contains publisher identification information.
DataPreparerIdentifier	theString	Use this to specify the data preparer identifier. If this string is preceded by an underscore (_) it is taken to be a file name on the root of the primary volume that contains data preparer identifier information.
ApplicationIdentifier	theString	Use this to specify the application identifier. If this string is preceded by an underscore (_) it is taken to be a file name on the root of the primary volume that contains application identifier information.
CopyrightFileIdentifier	theString	This is used to specify a file on the root level of the primary volume that contains a copyright message.
AbstractFileIdentifier	theString	Used to specify the name of a file on the root level of the primary volume that contains abstract information.
BibliographicFileIdentifier	theString	Used to specify the name of a file on the root level of the primary volume that contains bibliographic information.
VolumeCreationDate	theDate	This can be used to specify the creation date of the volume. Dates are specified in the following format: YYYYMMDDHHMMSSHHGG Where YYYY marks the year, MM the month, DD the day of the month, HH the hour (military format), MM the minute, SS the second, HH the hundredths of a second, and GG the Greenwich offset. If this command is not given, the current date and time will be used.
VolumeModificationDate	theDate	This can be used to specify the last modification date of the volume. If this command is not given, the current date and time are used.
VolumeExpirationDate	theDate	This is used to specify the expiration date for the volume. If it is not given, it defaults to 0000000000000000 which indicates no expiration date.
VolumeEffectiveDate	theDate	This is used to specify the effective date for the volume. If it is not given, it defaults to 0000000000000000 which indicates always effective.
ApplicationUse	theFile	There is an application use field specified by ISO for the primary volume descriptor. If this command is given, theFile will be opened, read, and the data within it will be placed in the application use field. If the file is too short to fit into the field, it will be padded with 0's. If it is too long, an error will be reported.
Hierarchy		This marks the beginning of the directory hierarchy for the primary volume.
EndPrimaryVolume		The definition of the primary volume is complete.

6.5 Supplementary Volume Commands.

Command tokens that are active within a supplementary volume:

DescriptorWrites	numWrites	This is used to specify the number of times that the descriptor record for the current supplementary volume will be written to the output. If this is not specified, it defaults to 1.
SystemIdentifier	theString	See primary volume definition.
VolumeIdentifier	theString	See primary volume definition.
EscapeSequences	???	This is not implemented yet, but is intended to allow for the definition of escape sequences as defined by ISO.
LPath		See primary volume definition.
MPath		See primary volume definition.
OptionalLPath		See primary volume definition.
OptionalMPath		See primary volume definition.
VolumeSetIdentifier	theString	See primary volume definition.
PublisherIdentifier	theString	See primary volume definition.
DataPreparerIdentifier	theString	See primary volume definition.
ApplicationIdentifier	theString	See primary volume definition.
CopyrightFileIdentifier	theString	See primary volume definition.
AbstractFileIdentifier	theString	See primary volume definition.
BibliographicFileIdentifier	theString	See primary volume definition.
ApplicationUse	theFile	See primary volume definition.
Hierarchy		See primary volume definition.
EndSupplementaryVolume		The definition of the supplementary volume is complete.

6.6 Volume Partition Commands.

Command tokens that are active within a volume partition:

DescriptorWrites	numWrites	This is used to specify the number of times that the descriptor record for the current volume partition will be written to the output. If this is not specified, it defaults to 1.
SystemIdentifier	theString	This gives the system identifier for the volume partition.
VolumePartitionIdentifier	theString	This names the volume partition.
SystemUse	theFile	The data in theFile will be placed into the system use field of the volume partition descriptor.
VolumePartitionData	theFile	The build program will use theFile as the data source for the volume partition. The data in theFile will be placed into the volume partition.
EndVolumePartition		When this token is processed, the volume partition description is complete.

6.7 Boot Record Commands.

Command tokens that are active within a boot record:

DescriptorWrites	numWrites	This is used to specify the number of times that the descriptor record for the current boot record will be written to the output. If this is not specified, it defaults to 1.
SystemIdentifier	theString	This gives the system identifier for the boot record.
BootSystemIdentifier	theString	This is synonymous with SystemIdentifier.
BootIdentifier	theString	Use this to specify the boot identifier for the current boot record.
BootSystemUse	theFile	The build program will use theFile as the data source for the system use field of the current boot record.
EndBootRecord		This marks the end of the boot record definition.

6.8 Directory Hierarchy Commands.

Command tokens that are active within a directory hierarchy:

Directory	theName		A directory is beginning. theName will be taken as the name of the directory as it will appear in the ISO image.
File	theName	theVersion	A file is beginning. theName will ba taken as the name of this file as it will appear in the ISO image. theVersion is optional, and if specified, tell the version number of the file. Version numbers can range from 1 to 32767. If not specified, the version number defaults to 1. Also, if the version number is given as 0, no version number will be written out to the file (this is not legal for an ISO 9660 file name, but is sometimes desirable)
Attributes	theAttribs		Attributes are specified for the hierarchy. theAttribs can be one or more of the following separated by whitespace: Hidden, NotHidden. If attributes are not specified, they default to NotHidden.
MinLength	theLength		Minimum number of bytes to use for directory. This can be useful when it is intended that in the future, the directory may have more files added into it. It simply forces the build process to allocate at least theLength bytes for the directory in the output image.
RecordingDate	theDate		This will allow for the direct entry of the recording date for the hierarchy. If this is not specified, current date is used.
EndHierarchy			The current hierarchy is ending.

6.9 Directory Commands.

Command tokens that are active within a directory:

Directory	theName	This specifies that a new directory is beginning within the current one. ISO allows for directories to be nested 8 levels deep. Beyond that point, warnings will be generated.

File	theName	theVersion	A file is beginning.
Attributes	theAttribs		Attributes are specified for the current directory. theAttribs can be one or more of the following separated by whitespace: Hidden, NotHidden. If attributes are not specified, they default to NotHidden.
MinLength	theLength		Minimum number of bytes to use for directory.
RecordingDate	theDate		This will allow for the direct entry of the recording date for the current directory. If this is not specified, current date is used.
EndDirectory			The current directory is ending.

6.10 File Commands.

Command tokens that are active within a file

Source	theFile	This is used to specify the data for the file being defined. theFile is opened, and its contents are placed into the output image. The size of theFile is taken as the size of the file in the image.
Attributes	theAttribs	Attributes are specified for the current file. theAttribs can be one or more of the following seperated by whitespace: Hidden, NotHidden, Record, NotRecord. If attributes are not specified, they default to NotHidden, NotRecord.
MinLength	theLength	Minimum number of bytes to use for this file. This is normally used when it is expected, that in the future, this file will grow, and an updated version of it will be placed into the image. It simply forces the build process to allocate at least theLength bytes in the image for this file. It does not change the size of the file as observed in its directory entry.
RecordingDate	theDate	This will allow for the direct entry of the recording date for the current file. If this is not specified, current date is used.
EndFile		The current file definition is ending.

7. The BuildDisc control file.

The control file for BuildDisc takes the following general format:

Disc discType outputFileName
commands that further define the disc go here

LeadIn trackType
commands that define the leadin track go here
EndTrack

tracks are placed on the disc in the order that they appear in the control file

Track trackType
commands that define track 1 go here

```
EndTrack

up to 99 tracks may be defined

Track                trackType
commands that define the track n go here
EndTrack

LeadOut              trackType
commands that define the leadout area go here
EndTrack

EndDisc
```

7.1 Global Commands.

Globally recognized command tokens (those that have meaning anywhere in the control file):

Define	**variable**	**value**	Defines a variable to a value (variable and value can be any string). Define can be useful for aliasing values that would be repeated often in a control file, or for importing values into the build (See the -d command line option). When BuildDisc starts, some variables are pre-defined: program - Name of the executing program version - Current version number edition - Current edition second - the second of the minute minute - the minute of the hour hour - the hour of the day day - the day of the month month - the month of the year year - the year weekday - the day of the week yearday - the day of the year
ShowDefines			Shows all currently defined variables.
Echo	**string**		Echoes the passed string to the standard output device (string can be anything). Echo can be useful for debugging a control file, or just for status information.
Include	**theFile**		Include another control File at this point (theFile must be a valid path and file name for the machine that buildDisc is running on).

7.2 Outermost Level Commands:

Command tokens that are active at the outermost level (before the beginning of a disc definition):

Disc	**discType**	**outputName**	Begin definition of a disc of type discType with output file name of outputName (Valid discTypes are CDDA for audio discs,CDROM for discs that contain Mode1 or Mode2, as well as audio tracks,CDI for CDI discs).

7.3 Disc Commands.

Command tokens that are active within a disc:

CatalogNumber	**theNumber**	Specify the catalog number for this disc (theNumber is a string of at most 13 ASCII digits). If the catalog number is not specified for a disc, it will not contain one.
LeadIn	**trackType**	Specify that the leadin area is beginning for this disc (trackType is CDDA for audio tracks, Mode0 for mode 0 data tracks, Mode1 for mode 1 data tracks, and Mode2 for mode 2 data tracks).
Track	**trackType**	Specify that a new track is beginning (trackType is the same as that for LeadIn). 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. Up to 99 tracks may be specified. The Track command is not valid unless LeadIn has been specified.
LeadOut	**trackType**	Specify that the leadout area is beginning for this disc (trackType is the same as that for LeadIn).
EndDisc		This marks the end of the current disc definition. When this token is seen, the actual disc build takes place.

7.3 Track Commands.

Command tokens that are active within a track:

Index		Place an index in the track at this point. Up to 99 indices may be specified for a single track.
Pause	**numBlocks**	Specify number of blocks to pause at start of track (numBlocks is the number of 1/75th of a second disc blocks that should be used for the pause (usually 150 (2 seconds)).
Source	**theFile**	This indicates that the data in theFile should be placed into the output image at this point. 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. If the file is not an integral number of blocks long (each audio block is 2352 bytes), then the output will be padded with 0's to bring it to an integral number of blocks. If the data is Mode1, theFile should just contain the raw binary data (BuildDisc will create the EDC/ECC, and all needed sync and header information). If the file is not an even multiple of the 2048 byte Mode1 block size, it will be padded with 0's to bring it to the nearest block. If the data is Mode2, it will be padded to the 2352 byte Mode2 block size.
Empty	**numBlocks**	numBlocks of zeros will be written into the output when this is seen. This can be useful when defining the LeadIn, and LeadOut tracks.

SubcSource	filename	This command is primarily used to place CD+G or CD+M data into the subcode of a track. Data from filename will be placed in the R through W channels of the subcode in the output image at the location where the SubcSource command is located. If the file runs out before the end of the disc, or before another SubcSource command is encountered, 0s are written into the R—W channels. Each byte of data in filename will contain 1 byte of subcode data, with bit 2 = R, bit 3 = S, ... bit 7 = W. Bits 0 and 1 would be P and Q, which are ignored.
SubcEmpty		Output 0s to the R through W channels of the disc image, at the point where the SubcEmpty command is encountered.
PreGap	numBlocks	Specify the number of blocks of PreGap information at the start of this track.
PostGap	numBlocks	Specify the number of blocks of PostGap for this track.
Channels	numChans	If the track is of type CDDA, this tells if it has 2 or 4 channels (Q subcode information). If this is not specified, it defaults to 2.
Preemphasis	boolean	If the track type is CDDA, this tells if the pre-emphasis bit in the Q subcode channel should be turned on (boolean is either TRUE, or FALSE). If not specified, this defaults to FALSE.
Copy	boolean	Tells the copy protect state of the audio data for the given track (if this is set to TRUE, digital copying will be allowed). If this is not specified for a given track, it defaults to FALSE.
ISRC	theNumber	Defines an ISRC number to be placed in the Q subcode for this track (the ISRC number looks as follows: OOCCCYYSSSSS where OO is the owner code (2 uppercase letters or digits), CCC is the country 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 the ISRC number is not specified for a track, it will not contain one.
EndTrack		This marks the end of the current track definition.

9. Warning and Error Messages.

The CTrac Builder system can potentially output a great variety of warning 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.

9.1 BuildTrack Warning Messages.

```
Abstract file Str not located in root of hierarchy
Application identification file Str not located in root of hierarchy
Bibliographic file Str not located in root of hierarchy
Copyright identification file Str not located in root of hierarchy
Data preparer identification file Str not located in root of hierarchy
```

`Publisher identification file Str not located in root of hierarchy`

The ISO 9660 standard assumes that any of the files named above, if they exist, will be found in the root of the hierarchy of the CD. If the file has not been placed in the root during the build, this error will be reported.

`This parameter seems a bit excessive, but if you insist...`

The value given for a particular command seems larger than would normally be expected for a parameter of this type. BuildTrack will allow you to use the large value, but will give you this warning, just in case you have inadvertently used the wrong number.

`Write overshoot (this is an indication of a possible internal malfunction)`

This is an internal diagnostic message which the user should never see. If you receive this message, please contact ICOM immediately.

9.2 BuildTrack Error Messages.

`Directory identifier Str is not unique`
`File identifier Str is not unique`

It is not acceptable to have a duplicate directory names at a given point in the directory hierarchy. Similarly, all files in a specific directory must have unique names. If any two directories, or any two file names, are found to be identical, this error message will be issued.

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

The -b option allows the user to specify the number of memory buffers that will be used by BuildTrack during the build. If two many buffers are specified, it might not be possible for BuildTrack to obtain enough memory from the Operating System (OS) to satisfy the request. If you receive this message, reduce the number of buffers requested.

`Failed to initialize`
`Failed to initialize file system`
`Failed to initialize volume parsing`

These messages will usually be accompanied by another message indicating that not enough memory was available to perform the required action. It they are issued on their own with no other error, than a unexpected malfunction has occured and you should call ICOM.

`Failed to write data area`
`Failed to write system area`

These messages will usually be accompanied by another message indicating that some sort of I/O error has occured. It they are issued on their own with no other error, than a unexpected malfunction has occured and you should call ICOM.

`Could not open control file Str1. OS Reports: Str2`
`Failed to open Str1 during build phase. OS Reports: Str2`
`Failed to open Str1 during partition build phase. OS Reports: Str2`
`Failed to open file Str1. OS Reports: Str2`
`Failed to open output file Str1. OS Reports: Str2`
`Failed to open system data file Str1. OS Reports: Str2`
`Failed to read file Str1. OS Reports: Str2`

```
Failed to read.  OS Reports: Str
Failed to write.  OS Reports: Str
```

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

```
Internal error (an inconsistent state was detected)
```

This is an internal diagnostic message which the user should never see. If you receive this message, please contact ICOM immediately.

9.3 BuildDisc Warning Messages.

```
Your lead-in track is somewhat short, (more than Str blocks would be nice)
```

Although the amount of data specified in your lead-in track is valid with respect to ISO 9660, it is general practice to use more data. The amount specified in the message is the normally accepted size.

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

The catalog and ISRC numbers can be specified more than once, and the last one specified will be used in the final disc image. This is just 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 ISO 9660 standard specifies that a mode 0 data track will have only data bytes containing 0 in the track. BuildDisc allows you to place any data in a mode 0 track, but will issue this warning if the data is not all 0.

9.4 BuildDisc Error Messages.

```
Could not add track to disc image
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 copy output file name
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. If they are issued on their own with no other error, than a unexpected malfunction has occured and you should call ICOM.

```
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 -b option allows the user to specify the number of memory buffers that will be used by BuildDisc during the build. If two many buffers 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 this message, reduce the number of buffers requested.

```
Failed to generate subcode for lead-in
Failed to generate subcode for lead-out
Failed to generate subcode for track
Internal Error in DoSource
Something BAD happened
```

This is an internal diagnostic message which the user should never see. If you receive this message, please contact ICOM immediately.

```
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: Str2
Failed to read lead-out source file Str1.  OS Reports: Str2
Failed to read subcode source file Str1.  OS Reports: Str2
Failed to read track source file Str1.  OS Reports: Str2
Failed to write lead-in blocks to output file Str1.  OS Reports: Str2
Failed to write lead-out blocks to output file Str1.  OS Reports: Str2
Failed to write track blocks to output file Str1.  OS Reports: Str2
Failed to write track lead-out to output file Str1.  OS Reports: Str2
```

All of these messages indicate that some type of I/O error occured 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 separated by at least 1 block
```

Indices (generated by an **Index** command) must come after the first block of data and be seperated by at least 1 block of data from each other. If you had two consecutive **Index** commands, you would get this message.

Sample control file for building an ISO 9660 track

```
;Build TIC-TAC-TOE Game for the Sega Genesis machine.

Define                          WD0             "Sunday"
Define                          WD1             "Monday"
Define                          WD2             "Tuesday"
Define                          WD3             "Wednesday"
Define                          WD4             "Thursday"
Define                          WD5             "Friday"
Define                          WD6             "Saturday"

Echo                            "[program] version [version].[edition]"
Echo                            "It is [hour]:[minute]:[second] on ([WD[weekday]])
                                [month]/[day]/[year]"

;volume information
Volume                          ISO9660         "trk1"      ;output image to be placed in
                                                            file trk1

SystemArea                      "systemCode"                ;want this file in system area

PrimaryVolume

SystemIdentifier                "FM TOWNS"
VolumeIdentifier                "TIC_TAC_TOE"
LogicalBlockSize                2048
VolumeSetIdentifier             "TIC_TAC_TOE"
PublisherIdentifier             "ICOM"
DataPreparerIdentifier          "ICOM CD DATA PREPERATION CENTER"
ApplicationIdentifier           "TTT"
CopyrightFileIdentifier         "TTTCR.DOC"
AbstractFileIdentifier          "TTTAB.DOC"
BibliographicFileIdentifier     "TTTB.DOC"

VolumeCreationDate              1990031615565000000
VolumeModificationDate          1990031615565000000
VolumeExpirationDate            1990031615565000000
VolumeEffectiveDate             0000000000000000000

LPath
MPath                           ;place path tables before the hierarchy

;the directory structure starts here

Hierarchy

File 'IO.SYS'                   ;these are needed to get the towns to boot
source IO.SYS
EndFile
```

```
File 'TBIOS.SYS'
source TBIOS.SYS
EndFile

File 'TBIOS.BIN'
source TBIOS.BIN
EndFile

File 'AUTOEXEC.BAT'
source AUTOEXEC.BAT
EndFile

File 'CONFIG.SYS'
source CONFIG.SYS
EndFile

File 'RUN386.EXE'
source RUN386.EXE
EndFile

File 'TTT.EXP'
source TTT.EXP
EndFile

File 'TTTCR.DOC'
source TTTCR.DOC
EndFile

File 'TTTAB.DOC'
source TTTAB.DOC
EndFile

File 'TTTB.DOC'
source TTTB.DOC
EndFile
;-------------------------------------------------------------------------------

Directory 'GRAPHICS'
     File                    'THEBOARD.PIC'
     source                  'C:\GRAPHICS\THEBOARD.PIC'
     Endfile

     File                    'X.PIC'
     source                  'C:\GRAPHICS\X.PIC'
     Endfile

     File                    'O.PIC'
     source                  'C:\GRAPHICS\O.PIC'
     Endfile
EndDirectory

EndHierarchy

EndPrimaryVolume

EndVolume
```

Sample control file for building a "Yellow Book" CD-ROM

Sample control file for building an ISO 9660 disc which contains 1 ISO 9660 Mode 1 track and 2 audio tracks, one of which contains an index.

```
Echo            "building ISO9660 disc image"

Define          TWOSEC          150             ;number of blocks in 2 seconds
Define          SRCPATH         "C:\myData\"    ;path to data for this disc

Disc            CDROM           output.dat      ;ISO discs are CDROM

CatalogNumber   0123456789012                   ;13 or less digits of catalog
number

LeadIn          MODE1                           ;information about the leadin
area
Empty           4500                            ;length of the leadin area (in
blocks)
PostGap         150                             ;need postgap for CDROM leadin
EndTrack

;-------------------------------------------------------------------------------

Track           MODE1           ;first track is mode1 (contains ISO data)
Pause           [TWOSEC]        ;wait 2 seconds before data begins
Source          [PATH]myTrk1    ;file that contains data for this track
PostGap         150             ;need postgap for switching track modes
EndTrack                        ;this track is done

Track           AUDIO
ISRC            ISUSA9001234    ;place ISRC code into subcode for track
Source          [PATH]Mozart    ;place this audio on this track
EndTrack

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
```