The Jaguar CD-ROM

The Atari CD is a low cost, high capacity data storage device capable of storing 746.9 megabytes of data. The Atari drive is double speed (≈353 kb/sec.). The uncorrectable error rate is less than 1 in 10^{-11}. All errors are flagged by the system so damaged blocks may be re-read.

There are a few differences between the Jaguar CD and other systems that you may be familiar with. These fall into two areas: performance and arrangement.

The Jaguar CD subsystem is high performance. For example, a MPC (Multimedia PC) has a minimum performance requirement that states that, "The drive must be capable of maintaining a sustained transfer rate of 150 kb/sec, without consuming more than 40% of the CPU bandwidth in the process." This data rate is half that of the Atari CD and the Jaguar will sustain the full 352800 bytes/sec. rate. This high performance level is achievable because of Jaguar's very large bus bandwidth.

All data on the disc is accessed directly, not via a file system with a directory structure. The data is arranged in a "raw" format compliant with Red Book except that Jaguar discs may be multi-session (defined by the Orange Book standard). There is a table of contents on the disc which may have up to 99 entries each referencing a single track (for more information about CDs, see the section below titled **A Bit About CD-ROMs**).

Data on the disc is referenced via the time stamp of the data. Time stamps assume single speed play and start at the beginning of the disc. The minimum addressable data unit on the disc is a frame. Each frame is 588 longs (2352 bytes). There are 75 frames per second at single speed. Any position on the disc is accessible via a time stamp of the format *mm*:*ss*:*ff* (*mm* = minutes; *ss* = seconds; *ff* = frames).

Reading data from a CD is an inexact process. When a command is sent to the CD to request data starting at a particular time code, the mechanism cannot guarantee that the data being sent is coming from the exact location requested. It is important to recognize that the data that is written into memory *will not* start at the exact beginning of the requested frame. In order to guarantee that the data you want will be contained in the data read we suggest that you start reading six frames before the first block you actually want and search for your partition marker[1] in memory for 31 frames (72,912 bytes) from this point. Please note that while this amount is sufficient for most 'gold' discs, we have found that some writer software induces additional skew which may need to be compensated for by additional pre-seeking. Manufactured discs are guaranteed to be well within the tolerances given.

It should be noted that the data from the CD maintains long alignment only. This means that graphics data cannot be guaranteed to have a particular phrase alignment. This phrase alignment must be accounted for in your code, or else the data needs to be moved.

In order to allow for changes in CD vendors and changes in data transfer mechanism, it is essential that *all* access to the CD and its associated controls be via the CD BIOS. The BIOS is meant to be as

[1] A partition marker is a 64 byte block of data consisting of 16 repetitions of the same longword. Partition markers are covered in more detail in the section **Jaguar CD-ROM Programming Procedures and Guidelines.**

unobtrusive as possible. A detailed description of the BIOS can be found in the section **The Jaguar CD-BIOS**.

A Bit About CD-ROMs

Fundamentally, CDs are a constant linear velocity (CLV), single-data-track optical media with one data surface. The single data track is in the form of a spiral about a mile long. Absolute position information is contained in a time code recorded within the data. The time code can be resolved to a single sector of 2352 bytes, of which, all may be data, or 2048 data bytes and the remainder for an additional layer of error correction. Atari Jaguar CDs are recorded in CD-DA "raw data" format, with Motorola byte-ordering, so there are 2352 bytes per sector, or block. The total capacity of a Jaguar CD is 746.9 megabytes.

The logical organization of a standard CD divides the disc into four types of regions: lead-in, tracks, pauses, and lead-out. The lead-in area is about 10000 sectors long, near the inner diameter of the CD. The Table of Contents (TOC) is repeated endlessly within the Q subcode of this region. Following the lead-in is the first pause region, which must be 150 or 225 sectors long. After the first pause comes the first track, which is a data region. If the CD has more than one track, every track must be separated by a pause region of 2 or 3 seconds. After the last track comes the lead-out region which contains primary data all set to zeros and an alternating P subcode channel bit.

Multi-session CDs appear logically as a set of up to 40 standard CDs arranged as sequential annular rings on the disc. Independent of the number of sessions on the CD, the total number of tracks must always be 99 or less for the entire disc. In theory, each session could have up to 99 tracks, for a total of up to 3960 tracks, but this structure is not yet officially supported by Philips and Sony. The track number limitation is usually overcome with a "logical block-logical file" structure that is built in software on top of the physical track structure.

Some Definitions

Absolute Time — The time code information in the Q Subcode that ranges continuously from 00:00:00 to a maximum of 73:59:75, beginning at the start of the first pause region on the disc.

Area or Region — A physical portion of the CD's data carrying surface that is 2D ring-shaped like a flattened doughnut.

Channel Frame — The fundamental packet size of 588 bits that is transmitted on the high-frequency signal sent by the laser playback head's output amplifier. The packet contains 24 bytes of primary data and 1 byte of secondary data (1 bit each, P through W subcodes) as well as all of the overhead data bits required to form the packet.

Finalize — The process of making a recordable CD readable by standard CD players involves writing the lead-in that includes the main TOC at the inner diameter. An unfinalized CD will generally be unplayable, except on CD ROM players specifically designed for this situation, such as Jaguar and Photo CD players.

 © 1995 Atari Corp.

Index — A pointer in the track that is currently playing. This sometimes used for accessing specific parts of tracks, independently of time code.
Lead-in — The region of the CD near the inner diameter that contains the table of contents, usually abbreviated as "TOC".

Lead-out — The region of a CD that marks the end of a disc, or end of a session on multi-session discs. Within this region, the primary data is set to zeros, while the P Subcode in the secondary data channel alternates the P bit at a 2 Hz rate.

Mode — The type of track (audio, ROM, CD+G, Karaoke, CDI, etc.) that is presently being read.

Open/Closed Session — The process of making a session valid after recording data in it on a recordable CD involves writing a lead-in and lead-out for it, called "closing" it. While the session is open, data can be appended to the session. An open session can not be accessed by Jaguar's CD Module.

Pause — A region of the disc that must contain only digital zeros of primary data while the P Subcode in the secondary data channel is set to all ones. Some software refers to this as "Track Lead-in."

Program — The main data region, or regions of a CD.

Relative Time — The time code information in the Q Subcode that ranges continuously from 00:00:00 to the end-time of a specific track, while playing (reading) that track.

Sector or Block — The smallest addressable unit of primary data storage, 2352 bytes, that can be read from the disc without post-processing of the data.

Session — A session is an area of a CD that has at least one complete set of region types. i.e. lead-in, pause, track (the data), and lead-out. A standard audio CD has a single session, usually with multiple tracks and pauses between the lead-in and lead-out. There can be as many as 99 sessions on a single multi-session CD (in fact only about 40 sessions will fit on a disc).

Subcode Data Channel — The serial secondary data read from the disc at 1/192 of the rate of the primary data, both of which are combined within the main channel. There are 8 subcodes within the secondary channel, identified as P through W. The Q Subcode contains the position information of the primary data channel sectors. The position information is in a time-based format of :

minutes:seconds:frames

Subcode Frame — The subcode channel information extracted from one sector of the CD. The subcode frame rate is 75 per second at 1X speed playback and 150 per second at 2X speed playback.

Table of Contents — The directory of the CD read from the Q subcode channel. Each program on the disc is listed according to its position on the disc. There can be as many as 99 items in the TOC. Special information items about the disc and its manufacturer can also be found here.

Track Number — The number of a program (audio selection for example) on the CD.

2. Jaguar CD-ROM BIOS

The Jaguar CD BIOS provides hardware transparent access to the Jaguar CD subsystem. **IT IS REQUIRED THAT ALL ACCESS TO THE CD BE THROUGH THE BIOS.** The BIOS gives you control over all major aspects of the CD system. The BIOS allows single or double speed operation, a choice of data paths into the system, a data transfer function and other features. For more information on the CD subsystem, see section 1 and the sample source code CD_SAMP.S and CD_ASAMP.S.

2.1 Calling the CD-ROM BIOS

To call the CD-ROM BIOS, you load the proper values into the appropriate registers, then do a 68000 **jsr *CD_routine*** call for the CD-ROM BIOS routine you want to call. The addresses of the routines are defined in the CD.INC include file. Each CD BIOS call may require up to 64 bytes of stack space so A7 should be configured properly prior to calling any CD BIOS routine.

The CD-ROM BIOS is installed automatically in a retail Jaguar CD-ROM system. In a development CD-ROM system, however, you must manually load the CD-ROM BIOS into DRAM. A debugger script (CDBIOS??.DB)[2] is provided for this purpose.

The following is a list of the CD BIOS calls. Each block gives:

1. The name of the call (and what version it is available in).
2. The call's use.
3. What registers are used for input.
4. What registers are used for output.
5. What registers are changed by the call.

2.2 CD BIOS Errors

The CD.INC file defines an error variable named *err_flag*, which will receive an error code from certain CD BIOS routines. A value of zero indicates no error; non-zero values indicate an error. The contents of *err_flag* are valid *only* after a CD BIOS function which is documented as setting it. However, it may be changed by other CD BIOS functions.

Proper error checking is mandatory when using the Jaguar CD-ROM. Failure to properly check for and handle error conditions may prevent your product from obtaining final production approval. You should always check *err_flag* after those CD BIOS calls that set it. Additionally, your program should have some kind of timeout mechanism to prevent the situation where it endlessly waits for a CD BIOS call to return (which could happen if other errors have not been properly handled).

[2] Different versions of the CD BIOS may be distinquished by the last two digits of the filename. For example, CDBIOS43.DB would be a DB script that would load version 4.3 of the CD BIOS.

2.3 Debugging with the CD-ROM BIOS

Two versions, revisions 2.x and 4.x, of the CD-ROM BIOS are currently distributed by Atari Jaguar Developer Support. If you have revision 1.0, you should download the two newer versions from Compuserve or the Atari Software Development BBS. Developer CD systems with the Butch 1 chip can only use revision two of the BIOS. Butch 2 systems can support either (you have a Butch 2 system if your CD system is in a modified production-level case).

When debugging a CD title you should format your data on a CD-R disc or the emulator as specified in section 6. The CD-BIOS must be soft-loaded prior to making any CD-BIOS call using the command '`load cdbiosxx.db`' where '*xx*' is the version number of the BIOS you want to load[3].

To debug, you will need a copy of the disc's table of contents. To create a copy, load the CD-BIOS and execute a short 68000 program such as the following:

```
.include    "jaguar.inc"
.include    "cd.inc"

.68000
.text

move.l      #$70007,D_END

jsr         CD_setup
move.w      #0,d0
jsr         CD_mode

lea         $2C00,A0
jsr         CD_getoc

illegal

.end
```

This program sets up the CD hardware, calls ***CD_getoc*** to read the table of contents at $2C00 and then ends on an illegal instruction. Now you can use the debugger command '`write toc.dat 2C00[400]`' to store the TOC to disc. This step needs to be performed each time the data on the disc changes.

Now, you can create a simple debugger script such as:

```
load cdbios40.db
read toc.dat 2c00
aread bootcode.cof
```

This will load the CD BIOS rev 4.0 , the Table of Contents, and your bootcode to the correct location so you can begin debugging. Your bootcode program should be the same (and at the same location) as you

[3] Depending on your system setup, it may be necessary to switch to the directory containing the CD-ROM BIOS files, typically \JAGUAR\CDROM, prior to loading the debugger and issuing this command.

 © 1995 Atari Corp.

will have the CD Boot ROM load your code. This bootcode must be <64k and is responsible for the loading of other code/data segments.

You should *never* place a ***CD_getoc*** call in your main code as the CD Boot ROM will load the table of contents on a booting disc at $2C00 automatically.

2.4 Reading Data with the CD-ROM BIOS

Data is normally read from a CD by calling one of three forms of ***CD_init*** (***CD_init***, ***CD_initf***, and ***CD_initm***) followed by any number of ***CD_read*** calls. With the current hardware, each form of ***CD_init*** loads a piece of GPU interrupt code which handles interrupts redirected from Jerry's I^2S interrupt. This may change as new versions of the CD hardware are produced.



Warning! The CD-BIOS GPU code does not distinguish between which interrupts actually came from Jerry and which came from other sources. For this reason, you should never enable other interrupts in the **JINTCTRL** register when a handler from any version of ***CD_init*** is active, otherwise they will be mistaken for interrupts from the CD interface.

Following is a brief description of the variants of ***CD_init***:

CD_init – Average speed, does not automatically locate data[4], uses no (non-interrupt) registers.
CD_initf – Fastest read, does not automatically locate data, uses more registers.
CD_initm – Slowest read, locates data, supports circular buffers, uses no (non-interrupt) registers.

When reading data at double-speed these interrupts occur approximately every 90 µsecs. Due to interrupt overhead the required maximum latency is reduced to ≈ 54 µsecs. If the Object Processor is used extensively, this number may be reduced. This means that no processor that has priority over the GPU must take control of the bus for longer than this period of time. Specifically, 68000 vertical-blank handlers are a likely cause of problems. Preferably, use the GPU for object-list update, etc... or, if you must, use only a tiny handler in the 68k.

If you do not wish to use the GPU for CD reading you can also use the DSP. To do this, you must install a DSP I^2S interrupt handler, call ***CD_jeri*** appropriately, and set **SMODE** to $14 (**SMODE** is set to the default of $15 by the Boot ROM and should be restored when done). This method eliminates the need for any form of ***CD_init***. When a ***CD_read*** call is executed your handler can now extract data from the CD. CD data transfers using the DSP are, however, subject to infrequent unreported data errors. Data whose integrity is required to be perfect should be checksummed.

To play Red Book audio you need a very simple interrupt handler that reads the incoming data from the CD and outputs it to the DACs (see the file INOUT.DAS in \JAGUAR\CDROM) for an example. You

[4] The ***CD_init*** and ***CD_initf*** routines do not guarantee that a data read will begin exactly at a specified time code. We recommend that CD reading begin six blocks ahead of where data is needed and that your buffer is searched for 31 blocks worth of memory. The ***CD_initm*** routine does, however, automatically search for data tagged by partition markers and locates the data in memory automatically.

can then call ***CD_read*** with the "Just Seek" bit set and the timecode of your track. Audio will be played by your interrupt handler but no data will be stored by any installed version of ***CD_init***.

2.5 Command Acknowledge

Several CD BIOS functions give you the option of waiting for an acknowlege that the command completed or returning immediately. The only restriction to the "return immediately" mode is that a ***CD_ack*** *must* be used prior to any subsequent CD BIOS command. With the ***CD_read*** command in seek mode, this delayed acknowledge is implied by the command so you must also do a ***CD_ack*** prior to any CD BIOS command that follows. This structure gives you the flexibilty to perform other calculations or do other processing while a command takes place.

2.6 Error Recovery Procedure for CD Read Operations

To retry a CD read operation that fails (i.e. ***CD_ptr*** returns an error result) while running in double-speed mode, the following steps should be performed:

1. Switch to Single-Speed using CD_mode.
2. Switch to Double-Speed using CD_mode.
3. Reexecute the CD_read.

This should make error recovery reliable under all circumstances where it is actually possible (i.e. the disk isn't actually damaged or defective).

2.7 CD BIOS Functions

2.7.1 CD_ack

Input	none
Register Usage	d1
Returns	Nothing in registers. Error code in global *err_flag*: 0 indicates no error, non-zero indicates error
Purpose	If any call uses the "return immediately" option, ***CD_ack*** may be used to wait for the requested action to complete. **Note:** Any call that does not "return immediately" uses this call to wait for completion. This means that *err_flag* is set.

2.7.2 CD_getoc

Note: This call should never be used by a bootable CD-ROM. It is for debugging purposes only.

Input	A0.L The address of 1024 byte buffer for returned multi-session TOC
Register Usage	none

 © 1995 Atari Corp.

Returns	TOC data in buffer located in DRAM at the location pointed to by A0.L. Format for the first record: +0 - Unused, reserved (0). +1 - Unused, reserved (0). +2 - Minimum track number. +3 - Maximum track number. +4 - Total number of sessions. +5 - Start of last lead-out time, absolute minutes. +6 - Start of last lead-out time, absolute seconds. +7 - Start of last lead-out time, absolute frames. Format for the track records that follow: +0 - Track # (must be non-zero). +1 - Absolute minutes (0..99), start of track. +2 - Absolute seconds (0..59), start of track. +3 - Absolute frames, (0..74), start of track. +4 - Session # (0..99). +5 - Track duration minutes. +6 - Track duration seconds. +7 - Track duration frames.
Purpose	The returned buffer will contain 8-byte records, one for each track found on the CD in track/time order. The very first record (corresponding to the "0th" track) has overall disc information.

2.7.3 CD_init

Input	A0.L The address of a long aligned block of GPU RAM 224 bytes long.
Register Usage	A1,D0
Returns	NADA
Purpose	This call loads support code into the GPU to support **CD_read**. At the present time this uses the DSP interrupt in the GPU. The ISR requires 16 longs of stack space, and uses only registers R28 to R31 in Bank #0 (which are the same as those normally reserved for interrupts). *Note that there must be a primary GPU process running in order for GPU interrupts to be processed and this primary process must define the interrupt stack in R31.*
See Also	***CD_initf, CD_initm***

2.7.4 CD_initf

Input	A0.L The address of a long aligned block of GPU RAM 216 bytes long
Register Usage	A1,D0
Returns	NADA
Purpose	This call is a version of ***CD_init*** that is about 30% faster but uses more registers. This call loads support code into the GPU to support ***CD_read***. At the present time this uses the DSP interrupt in the GPU. The ISR requires 16 longs of stack space, and uses registers R18 to R31 in Bank #0. *Note that there must be a primary GPU process running in order for GPU interrupts to be processed and this primary process must define the interrupt stack in R31.*
See Also	***CD_init, CD_initm***

2.7.5 CD_initm (CD BIOS Rev 3.0-up)

Input	A0.L The address of a long aligned block of GPU RAM 336 bytes long.
Register Usage	A1,D0
Returns	NADA
Purpose	This call is a version of ***CD_init*** that is slower, but supports automatic data positioning and circular buffers. At the present time this uses the DSP interrupt in the GPU. The ISR requires 16 longs of stack space, and uses registers R28 to R31 in Bank #0 (which are the same as those normally reserved for interrupts). *Note that there must be a primary GPU process running in order for GPU interrupts to be processed and this primary process must define the interrupt stack in R31.*
See Also	***CD_init, CD_initf***

2.7.6 CD_jeri

Input	D0.W 0 ⇒ No data to Jerry. !0 ⇒ Send data to Jerry.
Register Usage	D1
Returns	NADA
Purpose	This call allows CD data to flow to the I^2S port on Jerry. This allows audio data to go into the system without taking main system bandwidth.

2.7.7 CD_mode

Input	D0.W Speed/mode desired: Bit 0 ⇒ Speed: 0 = Single, 1 = Double Bit 1 ⇒ Mode: 0 = Audio, 1 = Data
Register Usage	D1,D2
Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call sets the speed of the CD to either single or double-speed and the data mode to either audio or data. **Note:** When in audio mode, the CD mechanism may alter data or mute it entirely to correct for 'spikes' in the data.

2.7.8 CD_mute

Input	D0.W 0 ⇒ Return immediately. 1 ⇒ Wait for completion.
Register Usage	D1
Returns	Error code in *err_flag*. No return value in any registers
Purpose	This call mutes the CD. It functions only in audio mode.
See Also	***CD_umute***

 © 1995 Atari Corp.

2.7.9 CD_osamp

Input	D0.W Oversample by 2^(D0). 0 ⇒ No oversample. 1 ⇒ 2x oversample. 2 ⇒ 4x oversample. 3 ⇒ 8x oversample.
Register Usage	NADA
Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call sets the amount of oversampling in audio mode. **Note:** This call will only perform the functions that the mechanism can actually do. If the mechanism cannot perform the oversampling requested it will do the next best that it can. **Important!!** This call will cause the data rate to Jerry's I2S to be multiplied by the oversample factor. Whatever software is handling Jerry had better be able to handle it.

2.7.10 CD_paus

Input	D0.W 0 ⇒ Return immediately. 1 ⇒ Wait for completion
Register Usage	D1
Returns	Error code in *err_flag*. No return value in any registers
Purpose	This call pauses the CD. When in data mode, data will still be sent but it will not be sensible. When in pause mode, the CD *will not* advance along the disc. This means that, when in pause mode, a ***CD_read*** call will fill the buffer with nonsense.
See Also	***CD_upaus***

2.7.11 CD_ptr

Input	NADA
Register Usage	NADA
Returns	A0.L Address of last written data. A1.L Approximate address of most recent error.
Purpose	This call returns the address of the last longword of memory that was written to. If no data has been read, this value will be one longword prior to the start of the read buffer (often a symptom of the GPU interrupt code not running). This address will change in jumps. The size of these jumps may change with any new version of the CD. This call also returns the position of the last detected read error since the start of the last ***CD_read*** command.
See Also	Section 2.6, **Error Recovery Procedure for CD Read Operations**

2.7.12 CD_read

Input	A0.L Beginning of destination data buffer. A1.L End of destination data buffer (as many as 64 bytes of data may be written past the end address specified in this register). D0.L Time code to start reading at: Bit 31 set ⇒ Just seek. The remaining bits are: *mm:ss:ff* (*mm* = minutes, *ss* = seconds, *ff* = frames). D1.L Partition marker longword to look for (***CD_initm*** variant *only*) D2.L Contains 'N' where 2^N defines the circular buffer size in bytes. The buffer must be aligned on a 2^{N+1} boundary. The minimum functional size for 'N' is 3. If the circular buffer feature is not needed, set D2 to 0. **Note:** The read will stop if the buffer pointer exceeds the value in A1 even if a circular buffer is defined. (***CD_initm*** variant *only*)
Register Usage	A2, D1
Returns	Error code in *err_flag*. No return value in any registers
Purpose	This call transfers data from the CD, starting at a given time code. The manner in which data is read depends upon the ***CD_init*** variant called (or if you are reading using the DSP) as shown in the table below. This call always returns immediately. Once this call is made, use the call ***CD_ptr*** to find out the position of the next address to be written to. If the "Just Seek" bit is set, no data is transferred, but the CD will continue to advance at the current speed. A ***CD_ack*** may follow *only* if the "Just Seek" bit is set.
See Also	***CD_uread*** Section 2.6, **Error Recovery Procedure for CD Read Operations**

CD_init Type	Description
CD_init	Data is read into the specified buffer until the end of the buffer is reached. The timecode specified to read from should be 6 frames prior to that actually needed and the partition marker indicating the start of the data may be anywhere within the first 31 frames (72,912 bytes) of the buffer.
CD_initf	Same as ***CD_init*** but faster.
CD_initm	Incoming data from the CD is scanned for a partition marker consisting of the longword specified. Once the partition marker is identified, data immediately past the partition marker will be read into the buffer. Though data is automatically located correctly in memory by this call, more system resources are used. **Note:** If the partition marker is not found, this call will look 'forever.' This call also supports circular buffers. When enabled, data will be read into the circular buffer indefinitely or until ***CD_uread*** is called.
None	If ***CD_jeri*** has been called and **SMODE** has been set to $14 to allow data to flow to the I^2S port, you may install a custom interrupt handler that will read data from the CD and use it as necessary.

 © 1995 Atari Corp.

2.7.13 CD_setup

Input	NADA
Register Usage	NADA
Returns	NADA
Purpose	This call *must* be used to initialize the CD system before *any* other calls can be made.

2.7.14 CD_spin

Input	D0.W 0 ⇒ Return immediately. 1 ⇒ Wait for completion. D1.W Session to spin up on.
Register Usage	NADA
Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call sets the CD drive to a specific session. **Note:** This call is not actually required for reading data in another session.

2.7.15 CD_stop

Input	D0.W 0 ⇒ Return immediately. 1 ⇒ Wait for completion.
Register Usage	D1
Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call stops the CD

2.7.16 CD_switch (CD BIOS rev 4.0-up)

Input	NADA
Register Usage	D0, D1, A0
Returns	No return value in any registers.
Purpose	This call allows a different disc to be inserted into the Jaguar CD without a reset occurring. This call should only be made after a ***CD_stop*** with the "wait for completion" flag set, followed by the display of a graphic asking that the user insert a new disc. When the a new CD is inserted, its Table of Contents will be read at $2C00 and control will be returned to the program. Do not assume anything about the state of the CD after this call. This means, for example, that ***CD_mode*** should be reissued to place the CD in the state you require. See the section ***Jaguar CD-ROM Programming Procedures & Guidelines*** for more information.

2.7.17 CD_umute

Input	D0.W 0 => Return immediately. 1 => Wait for completion.
Register Usage	D1

Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call unmutes the CD. It functions only in audio mode.
See Also	***CD_mute***

2.7.18 CD_upaus

Input	D0.W 0 ⇒ Return immediately. 1 ⇒ Wait for completion.
Register Usage	D1
Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call undoes the actions of ***CD_paus***.
See Also	***CD_pause***

2.7.19 CD_uread

Input	NADA
Register Usage	D0
Returns	Error code in *err_flag*. No return value in any registers.
Purpose	This call stops data recording started with a ***CD_read*** call. The disc will not be stopped by this call, only the data transfer. This call is used to cause early termination of a data transfer in case of an error, or to disable the CD data transfer when it is no longer needed and the resources it uses are required for other purposes.
See Also	***CD_read***

 © 1995 Atari Corp.

3. Jaguar CD-ROM Emulator Setup

3.1 Introduction — Version 2.0

This document provides the information you will need to connect your Jaguar CD-ROM Emulator to your Jaguar Development System. Before proceeding with the setup of your Emulator, verify that you have the following items ready to use:

1. An Atari Falcon030 Computer with mouse and AC power cord.
2. A Jaguar Development System.
3. A Jaguar Developer CD.
4. Three-header connector.
5. A Falcon030 to Jaguar adapter card with ribbon cable.
6. A Falcon030 Monitor Port to VGA connector adapter.
7. A SCSI hard disk drive (not supplied by Atari).
8. A SCSI cable with a high-density SCSI connector.
9. A VGA monitor with VGA cable (not supplied by Atari).

Note that the SCSI hard disk drive must be supplied by you. Not all SCSI drives will work in this application, due to variations in the speed, hard drive buffer size and caching strategies among different drives. Atari strongly recommends the Connor Peripherals CFP1060S or CFP1080S, both of which are 3.5" one-third height drives with a storage capacity of approximately 1 gigabyte. Use of drives other than these may give unusable results.

3.2 Step By Step Setup

1. **Connect the AC power, video monitor and mouse.** Attach the AC power cable to the connector marked "Power" on the back panel of the Falcon030. Plug the AC cable into a properly grounded electrical outlet. Plug the Falcon030 Monitor Port VGA connector adapter into the Falcon030 back panel connector marked "Monitor". Connect your VGA monitor cable to this adapter. Plug in the Falcon030's mouse to the connector with the mouse symbol, which is located underneath the Falcon030, near the right front edge of the unit. There is also a joystick connector in the same area – do not plug the mouse into that.

2. **Power-up the Falcon030 and check software installation.** Turn on the Falcon030 using the power switch on the back panel, near the AC power cord. On you VGA monitor, you should see a black and white low-resolution display of the boot-up sequence in which the Falcon030 checks itself. At the end of the boot sequence the screen resolution will increase and the desktop will be displayed. The open window will have the CD-ROM Authoring and Emulator software "CDROM.PRG" as the last item in the list of files displayed. You are now finished with this part of setup, so turn off the Falcon030.

3. **Connect your SCSI hard drive and verify accessibility.** Attach a SCSI cable to the port on the back panel of Falcon030 marked "SCSI". Since this is a high-density SCSI connector, you may require the adapter cable to connect to your SCSI drive. After you have attached your drive, turn on the

Falcon030, and watch for your SCSI drive to show up in the list of devices displayed on the VGA monitor during boot-up. Turn off the Falcon030.

4. Ensure that the ribbon cable is attached to the Falcon030 to Jaguar connector. Connect the ribbon cable to the Falcon030 to Jaguar Interface connector. The red stripe should be on the *opposite* side of pin #1 of the connector. If you had an older system, this is the reverse of the old setup. Attach the Falcon030 to Jaguar Emulator adapter card to the Falcon030 back panel connector marked "DSP".

6. Connect the CD-ROM and Falcon. The CD development system contains a simple PCB with three ribbon cable connectors as shown in *Figure 3-A*. All three connectors are keyed to prevent plugging them in incorrectly. The cable attached to the Falcon030 to Jaguar connector should always be plugged into the grey connector oriented differently from the two black connectors. The ribbon cable protruding from the CD-ROM unit should be connected to the black connector on the outside to use the CD-ROM unit normally and disable emulation. Connect the cable from the CD-ROM to the inside connector to emulate and disable the onboard mechanism.

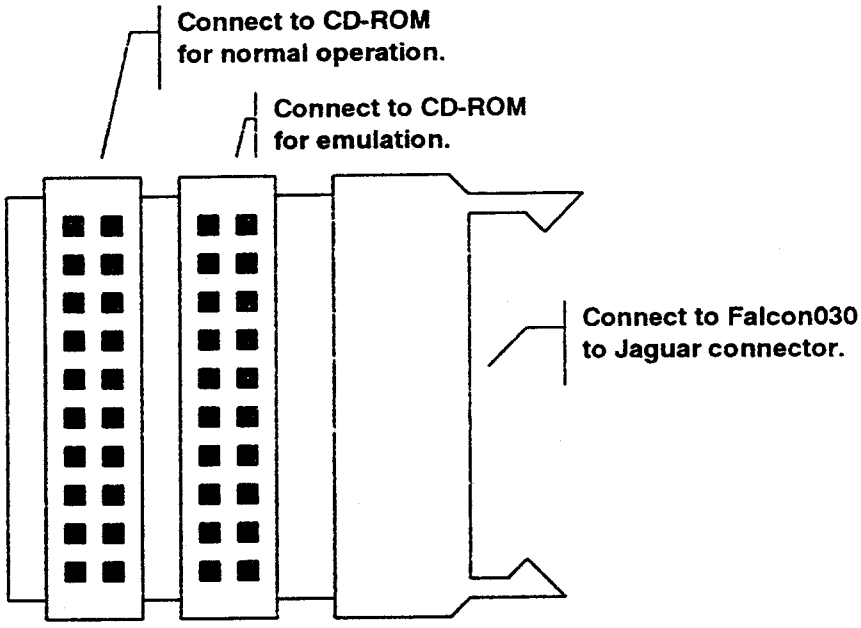


Figure 3-A – Three-Header Connector

That's it. The setup is done. If any of the above steps could not be accomplished, despite having all the bits and pieces and following the instructions, please contact Jaguar Developer Support.

To start using the Authoring Tool, turn on the Falcon030, wait for the desktop to appear and press the F1 key (or double-click on the file "CDROM.PRG"). Follow the **Jaguar CD-ROM Authoring Tool With Emulator Users Guide** to create a CD-ROM Table of Contents based on your SCSI drive's files.

 © 1995 Atari Corp.

4. Jaguar CD-ROM Authoring Tool With Emulator

The Jaguar CD-ROM Authoring Tool with Emulator provides a simple yet comprehensive user interface for creating sessions and tracks for a CD-ROM, and emulating the real hardware. To create tracks, the user specifies the files constituting a track. The data files can be audio/video data or executable code.

This software emulates CD-ROM by reading data from a large MS-DOS formatted SCSI hard disk drive. The SCSI identifier for the drive must be specified to the emulator. Failure to do so may result in the emulator refusing to initialize. Please refer to the section **How to set the SCSI identifier**.

4.1 Creating A New Document

To create a new document, choose "New" from the File Menu. The Authoring Tool will create a new document and will ask for a Title for the document. The window will show only one row saying "End of CD-ROM...", since you have not specified any files yet, as shown in *Figure 4-A*.

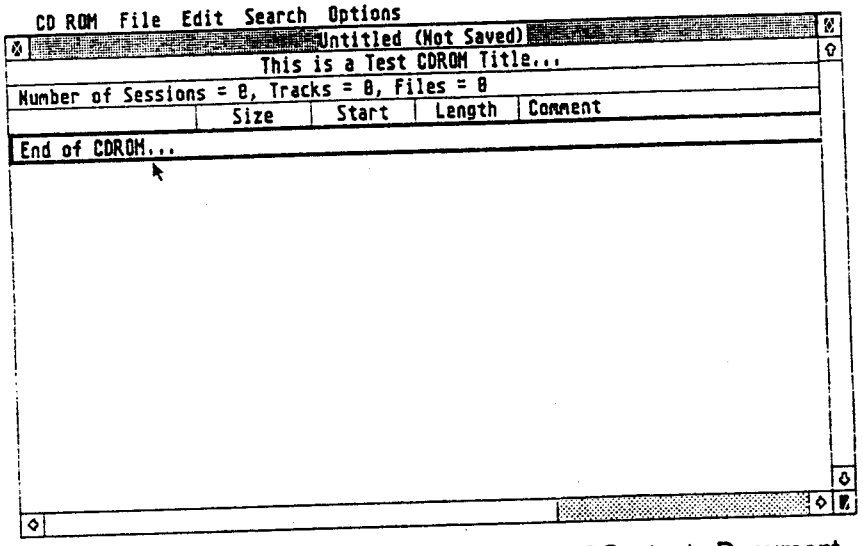


Figure 4-A – Creating a new CD-ROM Table of Contents Document

4.2 Opening An Existing Document

To open an existing document, choose "Open" from the File Menu. A file selector box will be presented in which you can select the document you want to open. Clicking on "OK" will open the document you just selected. The Authoring Tool will check for the validity of the files constituting the tracks in the document and update the position/length for each of them.

4.3 Description Of The Authoring Window

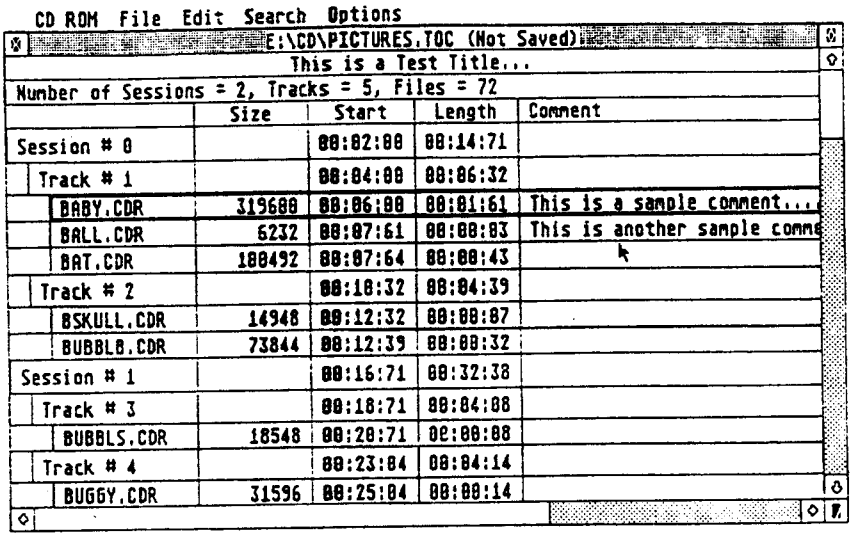


Figure 4-B – A CD-ROM Table of Contents Document

The Authoring Window is divided into various parts, as shown in *Figure 4-B*. The top row of the window contains the "Title" (user specified) for the document. The second row contains the total number of sessions, tracks and files used in this document. The next row contains the column headings, arranged as follows :

- The first column contains the current session number, current track number or filenames used to create the track. The tracks in a session are indented two characters inside the session to which they belong, and the files are indented further by two characters inside the track to which they belong.
- The second column contains the length of the files in bytes. The entries for session or track in this column are empty.
- The third column contains the start of the item on the CD-ROM in terms of it's time code position, also referred to as it's "time-stamp".
- The fourth column contains the length of the item in terms of time code.
- The fifth and last column contains the user specified comments for each item.

4.4 Current Item In The Window

The CD-ROM document opens in a window and presents itself in a hierarchical structure of Sessions/Tracks/Files. The "cursor" is a row-window, indicated by a thick border around the current row, as shown in *Figure 4-B*. Most of the editing operations work on the current row, depending upon whether it is a session or track or file.

 © 1995 Atari Corp.

4.5 Saving A Document

In order to save a document choose "Save" or "Save as" from the File Menu. For "Save As" a file selector dialog will appear and prompt you for the output path and filename.

4.6 Editing A CD-ROM Document

In the CD-ROM document, each session should have at least one track and each track should have at least one file. While editing a CD-ROM document, if the Authoring Tool finds that there are no files in a track or there are no tracks in a particular session, it will enter the required items automatically. If a new track is entered, then the subsequent tracks are renumbered. The default filename entered is "Untitled". This is true for all editing operations.

4.7 Inserting A Session

In order to insert a new session in the document at any position, choose "Insert Session" from the Edit Menu, as shown in *Figure 4-C*. This command inserts a new session before the current item. This function is disabled if it is not possible to insert a new session at the current row. A session should contain at least a track and each track should contain at least a file.

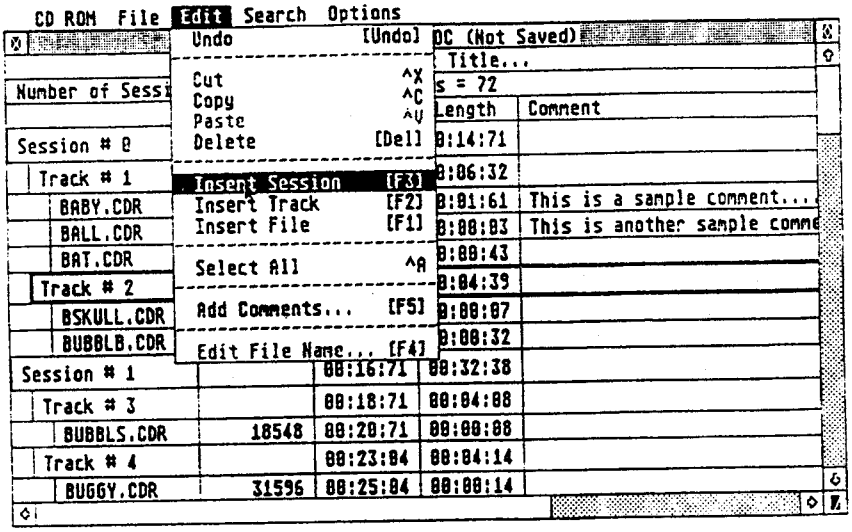


Figure 4-C – Inserting a New Session

4.8 Inserting A Track

In order to insert a new track in the document at any position, choose "Insert Track" from the Edit Menu. This command inserts a new track before the current item. This function is disabled if you can not enter a new track at the current row. A track should contain at least one file.

4.9 Inserting A File

In order to insert a new file at any position, choose "Insert File" from the Edit Menu. This command inserts an "Untitled" file before the current row. This function is disabled if you can not enter a file at the current row.

4.10 Editing A Filename

The Authoring Tool always enters an "Untitled" file of length zero when you create a new file. In order to edit this filename, use the cursor keys to make it the current item. Moving the mouse pointer over to the filename and clicking on it will also make it the current item. Now, choose "Edit Filename" from the Edit Menu to select a new filename. A file selector box will appear showing you the disk structure of the current SCSI drive being used. You can traverse through sub-directories and files on the disk and select the filename you want for the current item. The Authoring Tool will update the time code stamps for each item in the window.

4.11 Adding Comments

In order to provide some description for each item constituting the CD-ROM, the user can specify a description up to 64 characters long. To enter the description for a particular item, make that item the current item and choose "Add Comments" from the Edit Menu, as shown in *Figure 4-D*. A dialog box will appear where you can type the description you want for the item. This dialog box will also appear if you double click the mouse over the "comments" area for any item.

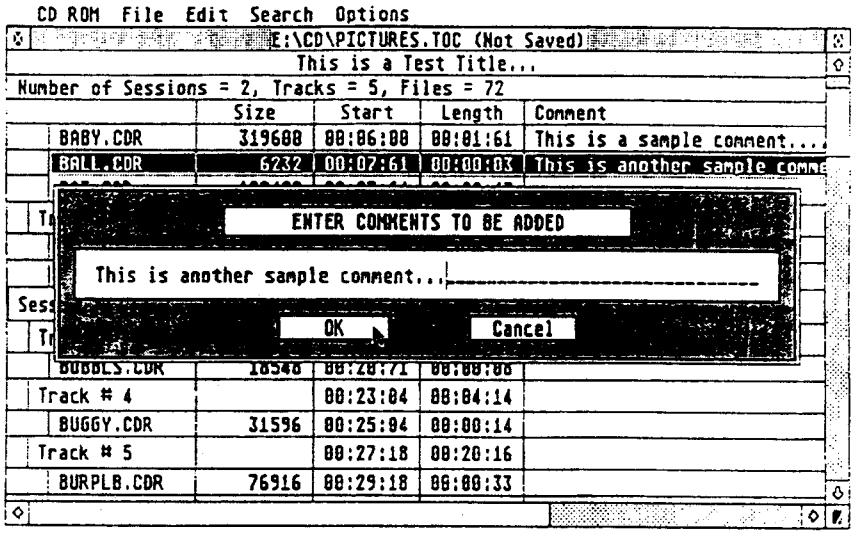


Figure 4-D — Entering Comments

 © 1995 Atari Corp.

4.12 Cut/Copy/Paste/Delete

The Authoring Tool provides common editing operations like Cut/Copy/Paste/Delete to make editing a CD-ROM document easy. In order to cut, copy or delete items from the document, first select the items and then choose "Cut", "Copy" or "Delete" from the Edit Menu. "Cut" will copy the items to the clipboard and delete them from the document, "Copy" just copies the items to the clipboard and "Delete" deletes the items from the document. If the clipboard contains CD-ROM document information already, you can paste that information to the document. The information added from the clipboard will go immediately before the current item. During these operations, if the Authoring Tool finds that any of the sessions are empty, it will enter a track for you. If any of the tracks are empty, it will enter an untitled file in those places for you. The Authoring Tool always updates the time code stamps for each item after each editing operation.

4.13 Undo

In order to undo the last editing operation, choose "*Undo*" from the Edit Menu.

4.14 Goto Session

In order to move to a specific session, click on "*Goto Session*" from the Search Menu.

4.15 Goto Track

In order to move to a specific track, click on "*Goto Track*" from the Search Menu.

4.16 Find/Find Next

You can also find a particular item by using this option.

4.17 Preferences – Specifying Lead-in/Lead-out for Sessions & Tracks

The Jaguar CD is a multi-session "Orange Book Standard" CD. The whole CD and each session within it contains certain specific regions. In order to specify length of such regions to the emulator, choose "Preferences" from the Options Menu. These regions may be lead-in/lead-out for sessions or the pause regions around the tracks, etc...

4.18 Preferences – Specifying SCSI ID

CD-ROM hardware emulation is performed by reading data from a large SCSI drive. Before this can be done, the SCSI identifier for the drive must be specified to the emulator by means of the Preferences dialog box. Failure to do so may result in the emulator refusing to initialize.

4.19 Preferences – How to set the SCSI identifier

The identifier of a SCSI device defines the number of the device set on its jumpers. The emulator expects this identifier to be specified through the preferences dialog box, and the emulator will use this identifier to access the data on that device. Sometimes, for an encased SCSI device, this identifier can be set by means of a rotary dial on the back of the case. In other SCSI modules, the ID can be set with dip switches. Consult the owner's manual of the SCSI sub-system or drive you are using.

4.20 Preferences – CD-ROM Latency

Different latency periods can be specified to the emulator by choosing "CD-ROM Latency" from the Options Menu, as shown in *Figure 4-E*. In our experience, these values are very 'worst-case'. You should probably set all of these values to zero since the existing defaults do not properly represent a production CD. If you are doing timing critical stuff you should burn a real disc to conduct your tests on.

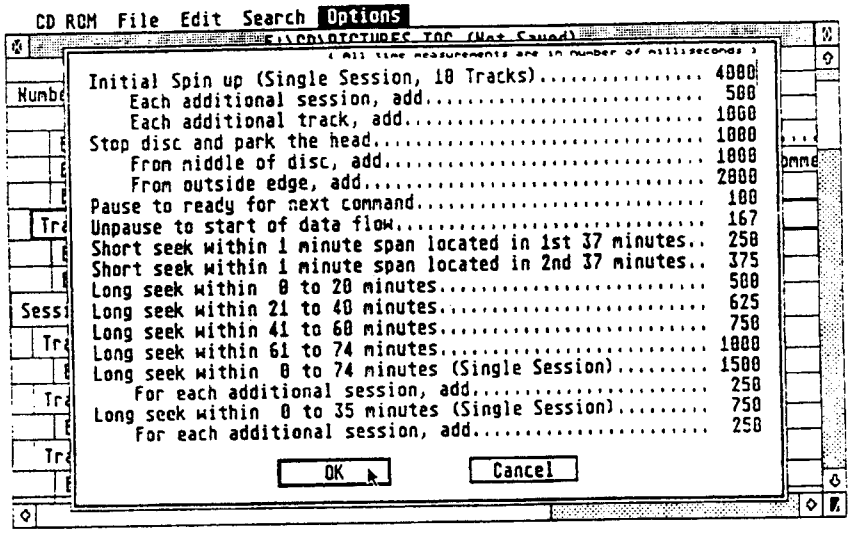


Figure 4-E – Editing the Latency Table

The default Jaguar CD-ROM Latency table is as follows:

Operation	Latency Time	Recommended
Initial Spin up - single session, 10 tracks	~~4 secs.~~	0 secs.
a. Each additional session, add	~~1 secs.~~	0 secs.
b. Each additional track, add	~~1/2 secs.~~	0 secs.

 © 1995 Atari Corp.

Operation	Latency Time	Recommended
Stop disc and park the head	~~1 secs.~~	0 secs.
a. From middle of disc, add	~~1 secs.~~	0 secs.
b. From outside the edge, add	~~2 secs.~~	0 secs.
Pause to ready for next command	~~1/10 secs.~~	0 secs.
Unpause to start of data flow	~~1/6 secs.~~	0 secs.
Short seek within 1 minute span, located in 1st 37 minutes	~~1/4 secs.~~	0 secs.
Short seek within 1 minute span, located in 2nd 37 minutes	~~3/8 secs.~~	0 secs.
Long seek within 0 to 20 minutes	~~1/2 secs.~~	0 secs.
Long seek within 21 to 40 minutes	~~5/8 secs.~~	0 secs.
Long seek within 41 to 60 minutes	~~3/4 secs.~~	0 secs.
Long seek within 61 to 75 minutes	~~1 secs.~~	0 secs.
Long seek from 0 to 74 minutes	~~3/2 secs.~~	0 secs.
a. For each additional session crossed, add	~~1/4 secs.~~	0 secs.
Long seek from 0 to 35 minutes	~~3/4 secs.~~	0 secs.
a. For each additional session crossed, add	~~1/4 secs.~~	0 secs.

4.21 Emulating The CD-ROM

After your various sessions and tracks of the CD-ROM have been specified, this function will emulate the Jaguar CD-ROM. To start, choose "Emulate CD-ROM" from the File Menu. The emulator will install various drivers and start emulating the CD-ROM by monitoring the Jaguar Console data requests and respond by sending data to the Jaguar Console, as if the Falcon030 were a Jaguar CD-ROM drive.

4.22 Stopping The Emulation

To stop emulation, press the "Esc" key.

4.23 Restrictions On The Emulation

- Data rate is always 95% of doubl ond vs. 352800).
- No CD errors will occur.
-
- t adeq ne real hardware in all cases, so the latency by you for your own disc structure's performance profile.

4.2 On Using The CD-ROM Emulator

Although the emulator allows you to specify multiple files per track, we suggest that you use one file per track. This way the emulator will give the best performance, when you compare it to an actual CD-ROM drive. The reasons for this are as follows:

The CD-ROM emulator allows you to add multiple files on each track on the CD. In order to do this, the emulator adds zeroes at the end of each file, so that the files are a multiple of 2352 bytes. This is done internally, and it does not effect the files on your SCSI drive.

In order to get the best performance from the emulator on the Atari Falcon030, version 2.0 of the emulator does this padding process differently. First the emulator adds zeroes at the end of each file so that the length of file is a multiple of 16K, and then it adds zeroes further so that the files are now a multiple of 2352 bytes in length. Again, this is done on the fly by the emulator as the files are sent to the Jaguar and it does not effect the files themselves on the SCSI drive.

Note that a lot of emulated space on CD-ROM is wasted in order to get the best performance from the emulator. This does not mean that your file layout on tracks should waste this kind of space. This is the reason you whould use only one file per track in practice. Therefore, the user should layout different files on a track and create one big file and specify it as one track to the authoring and emulation system. The version 2.0 accepts the old '.TOC' files from version 1.0. This '.TOC' format is a native format for the emulator.

4.25 Log File Name Preload Buffers (*File* Menu) (*Options* Menu)

These two menu items are not functional yet. The file name entered in ***Log File Name*** and the values entered in ***Preload Buffer Size*** dialog boxes are ignored.

 © 1995 Atari Corp.

5. CD-ROM Emulator Q & A

There are some common questions that arise even after reading the installation instructions delivered with the CD-ROM Emulation system. We want to address these in this document.

Question	What external hard drive are we supposed to buy?
Answer	The SCSI hard drives we have tested and know work are the Conner Peripherals CFP1060S and CFP1080S. Using other drives not tested by Atari may not give acceptable results. An external drive with its own case and power supply is most convenient, which is why we include a SCSI-II cable with the emulation system.

Question	How do I prepare and connect the drive for the Emulation System?
Answer	You must format the CD data drive on an Adaptec 1542 SCSI Controller in an MS-DOS based computer. Set the disk up with a single partition using the Adaptec tools. Now format it under MS-DOS (you need MS-DOS 5.0 or later to deal with partitions of this size). **Do NOT use *DoubleSpace* or other real-time disk compression utilities!!** Other SCSI cards may work, provided they create and use the exact same partition setup as the Adeptec. However, other cards have not been tested by Atari, so proceed at your own risk. After formatting, copy some of the files that you want to access as CD data to the drive. Switch your PC and CD data drive off. Disconnect the CD data drive from the PC. Connect it to the Falcon030 emulation computer. Now proceed as detailed in section 3.

Question	It looks like I can access the PC formatted drive even from the Falcon030 desktop (I can read and copy file from and to it) - is something wrong?
Answer	Don't even try to access the PC drive from anywhere except the File Selector dialog within the CD Emulator (see section 4.10). The hard disk partition scheme used by the Falcon030 is very close to that used by MSDOS, but it is not identical. Reading from the PC formatted drive on the Falcon030 will corrupt the internal memory structures of the Falcon030's operating system, which will in turn cause other errors and system crashes. It may even allocate all of the system's memory in a desperate attempt to make sense out of the PC drive's directory structure. This can lead to the failure to allocate memory as you start CDROM.PRG so that when you try to access the directory of the external drive you will see: "Error in Fileselektor Box! (Internal Error Number -3000)" Do not install a desktop icon on the Falcon030's desktop for accessing the emulation data drive. In the event you do accidentally read the PC drive (even just the directory), you should reboot the Falcon030 immediately to avoid problems. Likewise, attempting to write to the PC formatted drive from the Falcon030 will result in a corrupted disk structure, which will require that you repartition and reformat the drive, and then recopy all of your data files to it.

Question	Why may I get the message "Internal Error Number -4000"?
Answer	You may have set the wrong SCSI ID in the Emulator Software.

Question	I read data from the emulator but when I do a memory dump I see a region full of the hex value $DEAD.
Answer	This is the emulator's default return value for areas of the virtual disc that don't have any data associated with them such as the lead-in, lead-out regions and prior to and after valid tracks on the disc.

Question	What is the current distribution of the CD-oriented tools?
Answer	As Atari is constantly improving and updating the Jaguar Developer Tools, you should periodically check for new revisions on the Atari Software Development BBS or the Jaguar Developer's area on Compuserve (See **Online Support** in the **Getting Started** section).

Question	I have problems getting the technical information for the Conner drive, such as Jumper settings and so on. Where can I get these?
Answer	Call the Automated Conner HelpFax for all your possible drive information requirements. From any touchtone phone in the world you can call this number. The machine asks for the number of a FAX machine you want to get the information faxed to and directly faxes to that machine. The number of this Automated Conner Fax Service is: (408) 456-4903. Adaptec also has an info faxback service for their SCSI controller cards at (408) 945-6776, and a BBS for software updates at (408) 945-7727, Parameters 9600bps, 8N1.

Question	I can access the code, but the ***CD_read*** routine just stops working after transferring 5-20 kilobytes of information. What might be wrong?
Answer	It is likely that you are using a 68000 based vertical blank interrupt handler that consumes too much time. The time you have within the 68000 Vertical Blank Interrupt (VBL) must be significantly shorter than the time between two interrupts coming from the CD. Make it short. Do *not* build object lists within the 68000 VBL (this is generally true, not only for CD).

Question	The Falcon030 crashes everytime I do a ***CD_read***. What's wrong?
Answer	Versions of the CD emulator through v2.02 seem to have a bug where if you try to read data from before the start of the first track or after the data in the last track, the emulator will crash. We are working on this problem to remedy it. For now, add padding tracks as necessary to access your data.

 © 1995 Atari Corp.

6. The Jaguar CD-ROM: Programming, Procedures, and Guidelines

This is a "living document." Many of the details are subject to change but there are no expected changes that will cause overall structural changes or require changes in game code.

The Jaguar CD format is raw data and multi-session. Session #0 of a Jaguar CD is an audio-only session. It must contain only standard "Red Book" audio. This may be used to store future product information, sound track music, etc... ***No CD title that contains anything other than "Red Book" audio in Session #0 will be compatibility encoded.*** Atari will probably take the first track(s) for our own information. If you have no Red Book audio for your CD title, then you should test and submit your CD with at least one "dummy" track in Session #0.

All developer code starts with the next session, Session #1. The first track located in this session will be the boot track. The last track of the last session will contain data used by the Atari authentication code. The size of this track will be quite small, about 300k or less.

6.1 Track Headers and Tailers

The beginning of *each* data track (i.e. Session #1 and above) you provide must contain a specific Atari format data header and tailer. The track header must consist of 16 long-aligned repetitions of the ASCII block 'ATRI' (64 bytes) followed by the string:

```
ATARI APPROVED DATA HEADER ATRIx
```

This string is exactly 32 bytes in length. The last character is a special byte that increments for each track. Your first data track (i.e. the boot track) should have an ASCII space character in that position (0x20). In your second data track, this byte should be an ASCII exclamation point (0x21). In your third data track, this bytes should be an ASCII quote character (0x22), etc...

Each track must also end with a specific track tailer. The track tailer must consist of the string shown below *followed* by 16 long-aligned repetitions of the ASCII block 'ATRI' (64 bytes).

```
ATARI APPROVED DATA TAILER ATRIx
```

The last byte of the track tailer string should be the same as the last byte of the track header string for the same track.

No data should precede a track header or follow a track tailer.

6.1 The Boot Track

The boot track has two additional Motorola (MSB-LSB) style longwords that must follow immediately after the track header. The first is a long word that indicates the target address of your startup code and the second should indicate the length of your startup code in bytes. Your startup code should follow immediately after these two longwords. The CD Boot ROM will load a ***maximum*** of 64k of code at the location you specify in DRAM and transfer control to the 68000 at the start of this code. Your boot track may contain data beyond this 68k which will be your responsibility to load, however, the system will only load up to 64k.

When control is passed to your code, the results of a ***CD_getoc*** call will be in memory at 0x2C00. ***Your code must not call CD_getoc again.*** Use the table of contents to determine the first track in Session #1. The track number and timecode of all subsequent tracks should then be calculated as an offset to this. Do not reference absolute track numbers in your code because the layout of your CD is certain to change after compatibility encoding.

6.2 CD Track and Session Layout

Atari will master your CD using a two second lead-in period at the beginning of each track. The track start times found in the table-of-contents will account for this and point to the beginning of your data. The start times of every track, however, will change as a result of this process so you should not rely on absolute timings to find your data.

You should add a dummy track to your last session to simulate where the compatibilty encoding data will be placed. This track should be 156,192 bytes in length and may contain any dummy data. Please note that the final size of compatibility encoding data may vary due to the layout of your CD.
The first session will have at least as many tracks as you asked for (Atari will probably add one), your tracks will be at the end of the session. For example, if you give us a CD for compatibility encoding in the following format:

Session	Track	Contents
#0	#1	Developer Audio #1
	#2	Developer Audio #2
#1	#3	Developer Boot Code
	#4	Developer Game Data #1
	#5	Developer Game Data #2
#2	#6	Developer Game Data #3
	#7	Developer Game Data #4
	#8	Dummy End Track (required)

Atari will master a CD and return it to you in the following encoded format:

Session	Track	Contents
#0	#1	Atari Audio
	—	maybe more Atari audio tracks...
	#2	Developer Audio #1
	#3	Developer Audio #2

 © 1995 Atari Corp.

Session	Track	Contents
#1	#4	Developer Boot Code
	#5	Developer Game Data #1
	#6	Developer Game Data #2
#2	#7	Developer Game Data #3
	#8	Developer Game Data #4
	#9	Atari Compatibility Encoding Data

6.3 Minimizing Startup Delay

One goal of the Jaguar CD is to remove the "slow" stigma from CD-ROM. Using a small number of sessions will minimize startup time.

At startup, disc authentication takes place. During authentication your code will be scanned for partition markers that separate your data into blocks of a managable size. Partition markers are sixteen consecutive and identical longwords that are long-aligned relative to the beginning of the track. Each track header and tailer, for instance, contains a marker using 16 longwords of 'ATRI'. **Do *not* use a sequence of 'ATRI', 0x00000000, or 0xFFFFFFFF for a partition marker.**

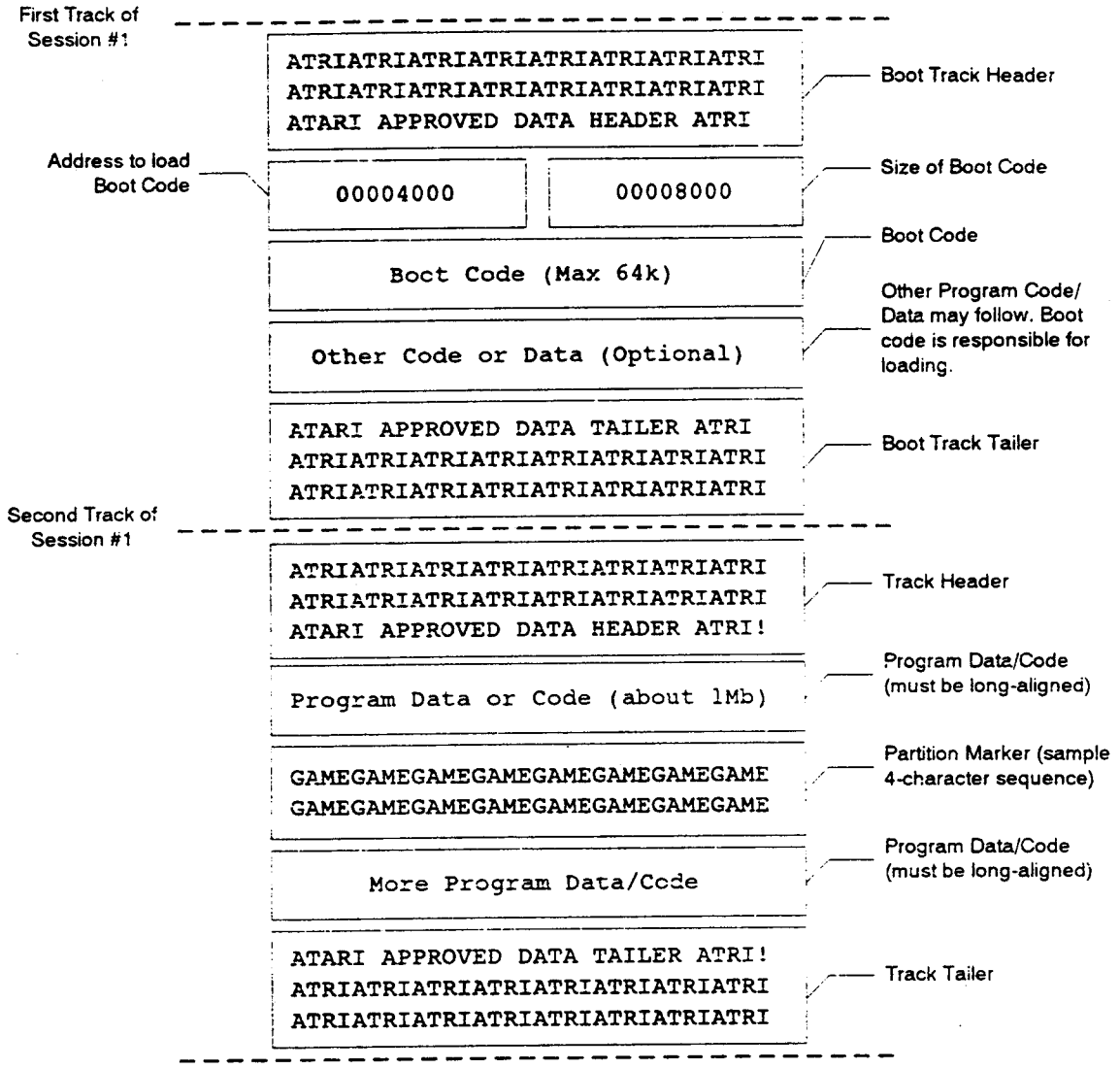
We recommend that you break-up any tracks containing more than 1Mb of data with partition markers so that a partition marker occurs approximately between every chunk of data between 128k and 1Mb in size. This will ensure that the authentication process is reasonably quick. The worst-case authentication delay will be no shorter than the time it takes to read the data between the two headers with the longest gap.

6.4 Minimizing Loading Delay

The best way to minimize loading delay is to plan ahead. Design your software so that there is enough time to load new data in the background. This technique, used in the Cinepak demos, allows continuous streaming of data many times larger than DRAM with *no loading delays*. The latest release of the CD BIOS contains a new ***CD_initm*** call that enables a special version of ***CD_read*** that reads continuously into a circular buffer with no extra programming. Designing both the game play and the programming to avoid loading delays will be a significant effort but it will be well worth it.

6.5 Track Layout Diagram

The following diagram is a sample of how a boot track and subsequent code/data tracks should be laid out:



6.6 Using Red Book Audio

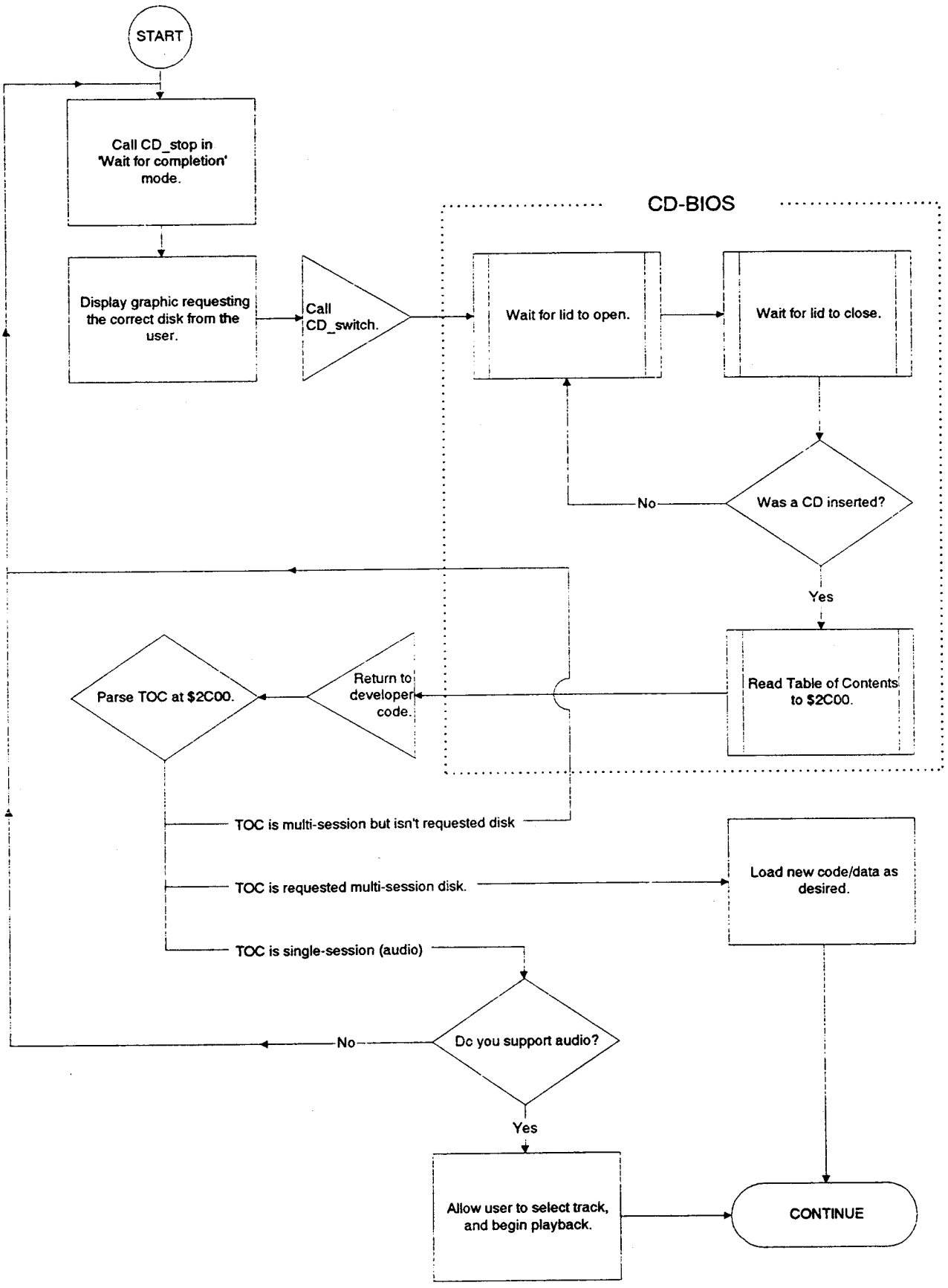
Titles designed for use with Jaguar CD may optionally use Red Book audio as in-game music. Normally this music should be placed on Session #0 so that the user may listen to it in a normal CD player. Optionally, 'secret' game audio may be placed on later sessions so that the game can restrict access to this track until game/level completion etc... Placing Red Book audio on sessions after Session #0 will prevent playback on an audio CD player.

If your title requires little or no CD access once your code is loaded, you may also, optionally, provide the user with an option to insert another Red Book audio disk for playback during gameplay.The procedure for using multiple discs within a game is contained in the following section.

 © 1995 Atari Corp.

6.7 Accessing Additional CD-ROM Discs

Despite the large amount of data capable of being stored on CD-ROMs, some titles are beginning to appear which require multiple discs. In addition, some games with minimal data requirements may offer the user the choice of inserting Red Book audio discs which can be used to replace in-game audio. The Jaguar CD-ROM BIOS contains a call (***CD_switch***) which automates the process of accepting a new CD and re-reading the disc's Table of Contents. Please examine the flowchart below which demonstrates the disc switching process.



 © 1995 Atari Corp.

7. Jaguar CD Mastering

Now that you've read all about the format of a track on a Jaguar CD, you are ready to master your first disc. The first thing you need is a computer system with a CD-Recordable Writer. Next, you need a CD Mastering software package and some idea of how to use it. Finally, you need data to put on your CD.

7.1 CD Recorders & Mastering Software

There are many CD-Recorder/Players and CD mastering software packages to choose from today. At Atari we use a ***Phillips CDD-522*** CD Recorder connected to a 486-based PC machine using an Adaptec SCSI host adapter. We have not tested other recorders or platforms; they may work just fine for creating Jaguar CDs, but require different configurations. Note that some developers have reported problems using some of the new generation of less-expensive CD recorders to create multi-session discs (a Jaguar CD requirement).

The CD mastering software used most often at Atari is ***CeQuadrat's WinOnCD Pro*** and ***Easy CD Pro v3.0*** from ***InCat Systems***. These packages both run under Microsoft Windows[5] and allow you to make discs in different formats such as CD-DA (Digital Audio), ISO 9660 CD-ROM, and CD-XA.

Atari has not had any success creating Jaguar discs with the current version of ***Corel CD Creator***. It requires Windows WAV sound files as input for creating tracks on an CD-DA disc and won't work with raw binary files. See section 7.1.1 for more information on this situation.

7.1.1 If Your CD Mastering Software Won't Work With Binary Files...

A Jaguar CD looks very much like a standard audio CD, except that it is multisession. In most CD Mastering software programs, you specify "Audio" or "Raw" as the track type. Unfortunately, some CD mastering software packages, such as ***Corel CD Creator***, do not have the ability to create a "Raw" track, and do not allow you to create an audio track from a raw binary data file. They require that the file must look like an AIFF or WAV audio file, even though the AIFF or WAVE file wrapper is removed prior to the data being written to the disc.

Atari supplies a tool known as the ***Jaguar CD Track Creator*** that is used to create a track file for CD mastering from the Jaguar program and data files you specify (see section 7.2 for more information). However, the current version of this tool has no option to add an AIFF or WAV wrapper to the files it creates; this must be done as an additional step afterwards. The ***MKAIFF*** tool included in the Jaguar Developer's Kit as part of the Jaguar Sound & Music tools can be used for this purpose right now, but this feature will be added to future versions of the ***Jaguar CD Track Creator*** program.

An early approach to this problem was the *FilmToAIFF* option of the ***Jaguar Cinepak Utilities*** program. However, this only works with Jaguar Cinepak Film files, which isn't the only thing you'll need to put onto a Jaguar CD disc. There are other problems as well, and we recommend that this option no longer be used. For more information see the **Cinepak For Jaguar** chapter.

[5] In fact, we are currently running them under the beta release of Windows 95 (build 4.00.347).

The best solution is to select a CD Mastering package that doesn't have any restrictions regarding what type of files can be used as source data. See section 7.1 for information about the CD mastering package used by Atari.

7.1.2 Other Considerations

Note that some CD-ROM mastering software automatically inserts two seconds worth of silence (150 blocks at 2352 bytes each = 352800 bytes) at the start of each audio track it creates. If your CD-ROM mastering software does this, you should turn this feature off if possible. If you can't turn it off, you should consider getting a new CD-ROM mastering software package. Until you do that, you will have to account for this extra data whenever reading data from the CD.

7.2 Jaguar CD Track Creator

In order to put your data into the proper format for creating a CD track, Atari supplies the ***Jaguar CD Track Creator*** program. This program runs under Microsoft Windows[6] and allows you to create track files suitable for mastering a Jaguar CD disc. *Figure 7-A* shows what the program looks like on screen when you run it.

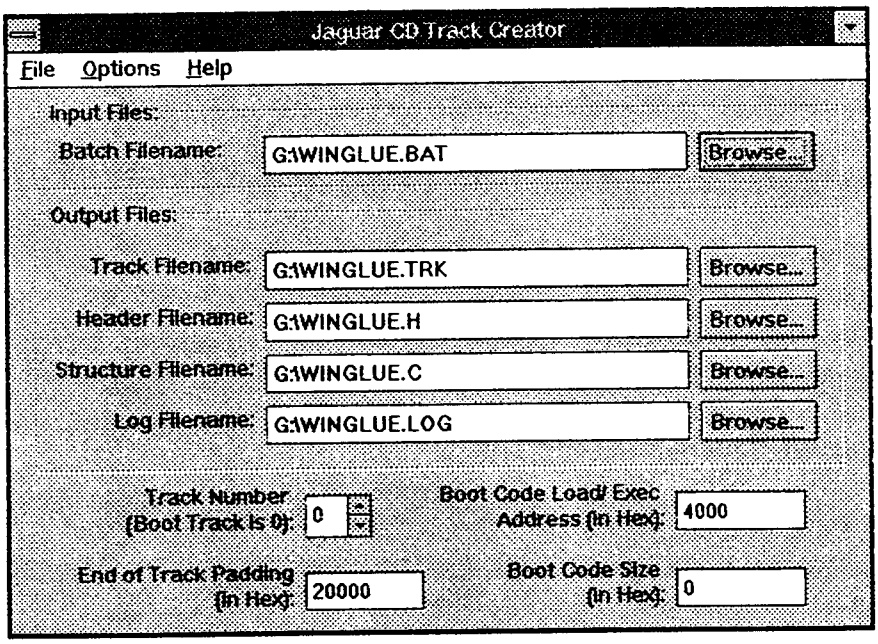


Figure 7-A — *Jaguar CD Track Creator*

The ***Jaguar CD Track Creator*** takes care of all the dirty work of merging all of your data files together and creating a track file with the proper header and tailer (as described in section 6.1). You provide it a list of files, and it combines them into a single large file, separated by a 64-byte partition sync marker of your choosing, complete with the proper track header and tailer information. If you specify track #0, it also inserts the fields for the load address and size of your boot code.

6 It has been tested with Windows 3.1, Windows For Workgroups v3.11, and Windows 95 beta 4.00.347.

7.2.1 Input Files

The ***Jaguar CD Track Creator*** takes two different categories of files as input. The first category consists of the files that contain your Jaguar program code, graphics, music, sound effects, and so forth. The second category is a batch file that lists all of the files from the first category that must be merged together into a CD track file.

7.2.1.1 Batch Filename

Clicking on the *Browse* button next to the *Batch Filename* edit box at the top of the window will bring up a standard Windows file selector dialog and allow you to select the name of your batch file that contains the list of files that will be used to construct your track, along with the partition sync marker codes that will be used for each file. Optionally, you may simply type in the filename.

The batch file is an ASCII file that has one or more lines of information (separated by CR/LF) with the name of your data file, a Tab character (ASCII 9), and a 4 letter code that will be repeated 16 times to create a 64-byte partition sync marker that will delineate the beginning of this particular file within the track (see *Figure 7-C* and *Figure 7-D*). At runtime, your code will search for this 64-byte block and know that the desired data comes immediately afterwards.

Section 7.2.1.2 shows a sample batch file. In this example, we are creating a file for our boot track. Our boot code is contained in the file G:\JAGUAR\PROJECT\BOOTCODE.BIN, so the first line of the batch file contains this filename followed by a <TAB> and then the 4 letter partition sync marker "CODE". This is followed on the second line of the batch file by the file name for our title screen data, G:\JAGUAR\PROJECT\TITLESCR.RGB, which is followed by a <TAB> and the four letter partition sync marker "SCRN". Finally, the last line of the batch file specifies the last file of the track, our music score which is contained in G:\JAGUAR\PROJECT\MUSIC.DAT, and a partition sync marker of "MUSC".

7.2.1.2 Sample Batch File

```
G:\JAGUAR\PROJECT\BOOTCODE.BIN      CODE
G:\JAGUAR\PROJECT\TITLESCR.RGB      SCRN
G:\JAGUAR\PROJECT\MUSIC.DAT         MUSC
```

Figure 7-B — *Contents of sample batch file*

7.2.2 Output Files

The *Track Filename*, *Header Filename*, *Structure Filename*, and *Log Filename* fields specify the filenames that will be used to create your output files. These fields are filled in automatically when you *Browse* your input *Batch Filename*, using derivatives of the batch filename. You can also type in the filenames or use the file selector by selecting the *Browse* button next to the desired field.

7.2.2.1 Track Filename

The *Track Filename* field specifies the name of the raw track file that will be created. This file is ready to pass to your CD mastering software to create a CD track.[7] The file created will have the structure shown in *Figure 7-C* if you have specified track #0.

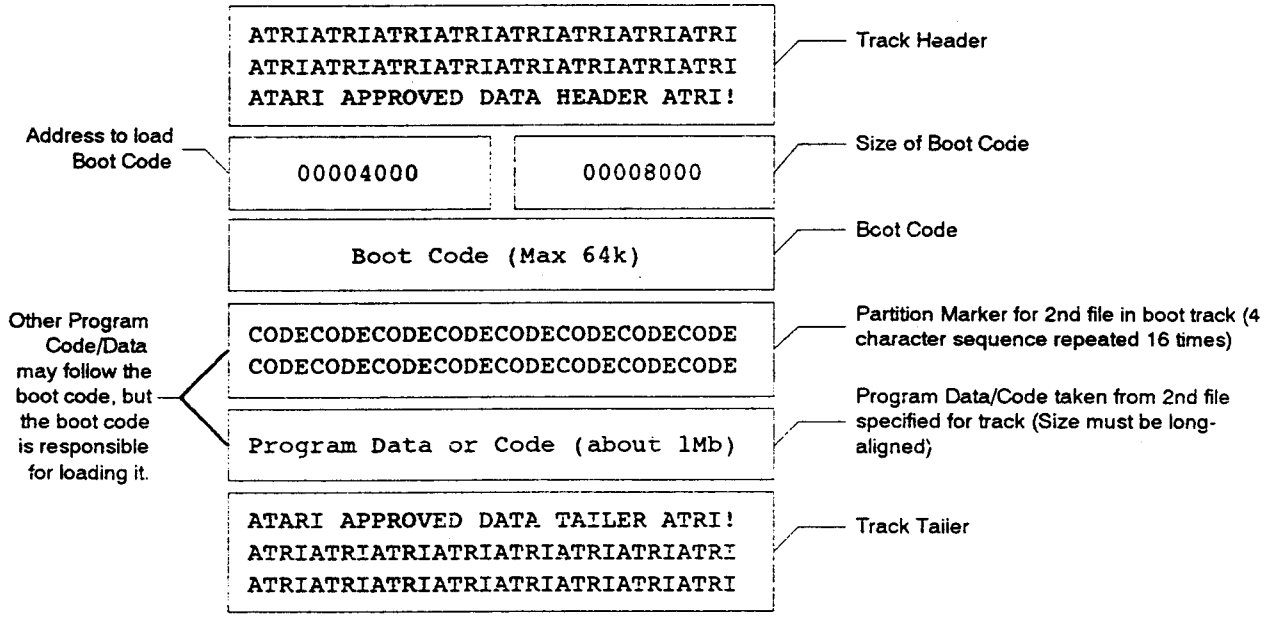


Figure 7-C — CD Boot Track Structure

Track #0 is handled specially because of the requirements of the boot code. First note that the boot code block does not have a partition sync marker in front of it (such as the "CODE" marker preceeding the next program data/code block). This is because the boot code is loaded for you automatically by the system, and must always be at a specific offset from the track header anyway, so there's really no need for your program to have a specific partition marker for this particular data.

If you have specified a track other than #0, the track file structure will be as shown in *Figure 7-D*. The main difference is that there are no fields for the load address and code size of your boot code and that the first file is not treated specially, so it gets the partition sync marker specified in your batch file.

While it's true that the partition sync marker is not absolutely required for the first chunk of data in a track, because you could use the track header instead, it is included because it makes it easier for your program to deal with all of your code and data files in the same way, regardless of their position within a track.

[7] See section 7.1.1 for additional information which may be relevant.

 © 1995 Atari Corp.

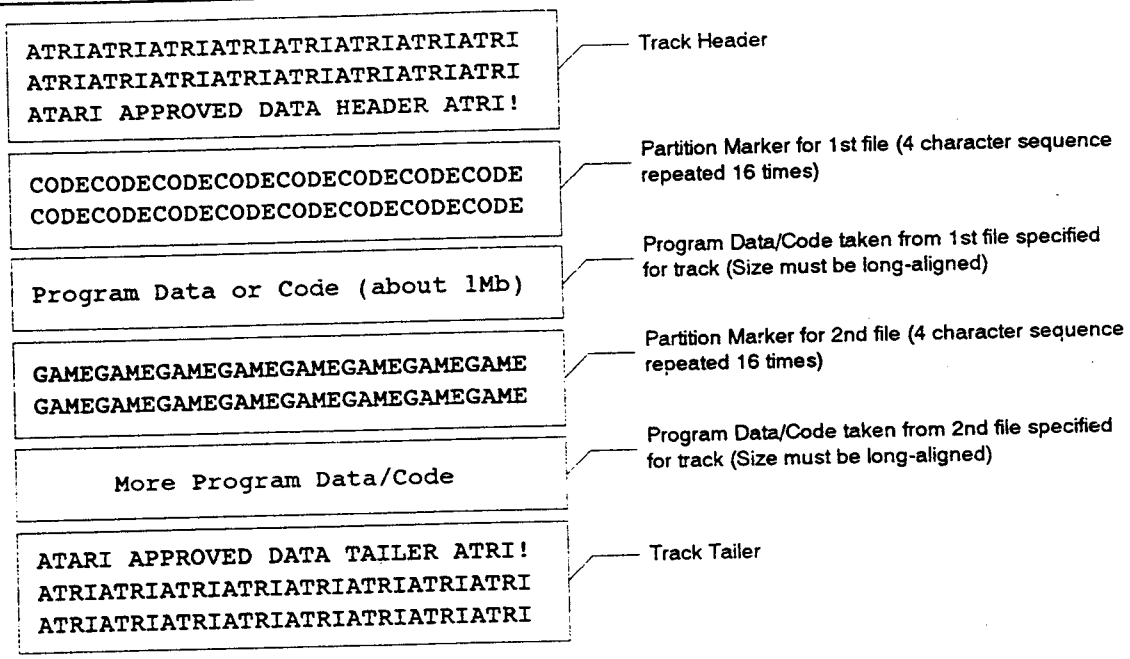


Figure 7-D — CD Track File Structure

7.2.2.2 Header Filename (*.H or *.INC)

The *Header Filename* field defines the name of a file that will be created by the ***Jaguar CD Track Creator*** with definitions corresponding to the order of the files within the track.

If the *C Language Output* option of the *Options* menu is selected, the file created will be a C language header file. See *Figure 7-E* for a sample C language header file created from the sample batch file in section 7.2.1.2.

```
#define   FILE_ G:\JAGUAR\PROJECT\BOOTCODE    0
#define   FILE_ G:\JAGUAR\PROJECT\TITLESCR    1
#define   FILE_ G:\JAGUAR\PROJECT\MUSIC       2
```

Figure 7-E — Sample C Language Header File

If the *Assembly Output* option of the *Options* menu is selected instead, the file created will be a Madmac assembly language include file. See *Figure 7-E* for a sample Madmac include file created from the sample batch file in section 7.2.1.2.

```
FILE_  G:\JAGUAR\PROJECT\BOOTCODE    equ    0
FILE_  G:\JAGUAR\PROJECT\TITLESCR    equ    1
FILE_  G:\JAGUAR\PROJECT\MUSIC       equ    2
```

Figure 7-F — Sample Madmac Assembly Language Include File

7.2.2.3 Structure Filename (*.C or *.S)

The *Structure Filename* field defines the name of a source code file that will be created by the ***Jaguar CD Track Creator*** with an array of structures containing information about the files placed into the track file. There will be one element in the array for each file placed into the track file. In "C", the structure is defined as:

```
typedef struct
{
      int       track;
      long      block_offset;
      long      length;
      long      marker;
} FILEDATA;
```

The *track* field indicates the track number where the file is located. The *block_offset* field indicates the offset, in CD blocks, from the beginning of the track to where the file data is located. The *length* field specifies the length of the file data in bytes. The *marker* field specifies the 4 byte partition sync marker used for this file.

If the *C Language Output* option of the *Options* menu is selected, the file created will be a C language source file containing an array of FILEDATA structures. See *Figure 7-G* for a sample C language source file created from the sample batch file in section 7.2.1.2.

```
FILEDATA fd[] = {
{ 0x01, 0x00000000, 0x0004BA04, 0x57494E47 }, /* FILE_ G:\JAGUAR\PROJECT\BOOTCODE CODE */
{ 0x01, 0x00000083, 0x0000EA04, 0x46494C32 }, /* FILE_ G:\JAGUAR\PROJECT\TITLESCR SCRN */
{ 0x01, 0x0000009D, 0x0000009C, 0x3344534F }  /* FILE_ G:\JAGUAR\PROJECT\MUSIC MUSC */
};
```

Figure 7-G — Sample C Language Structure File

If the *Assembly Output* option of the *Options* menu is selected instead, the file created will be a Madmac assembly language source file. See *Figure 7-E* for a sample Madmac include file created from the sample batch file in section 7.2.1.2.

```
fd::    dc.w $01
        dc.l $00000000,$0004BA04,$57494E47  ; FILE_ G:\JAGUAR\PROJECT\BOOTCODE CODE
        dc.w $01
        dc.l $00000083,$0000EA04,$46494C32  ; FILE_ G:\JAGUAR\PROJECT\TITLESCR SCRN
        dc.w $01
        dc.l $0000009D,$0000009C,$3344534F  ; FILE_ G:\JAGUAR\PROJECT\MUSIC MUSC
```

Figure 7-H — Sample Madmac Assembly Language Structure File

7.2.2.4 Log Filename (*.LOG)

The *Log Filename* field specifies the filename of a file that will be created as a log of the entire track creation process. This file contains basically the same information about each file used to create the track as what is shown in *Figure 7-G*, except in a more human-readable text format.

 © 1995 Atari Corp.

7.2.3 Track Options

The bottom of the main screen shows a number of options for the track being created.

7.2.3.1 Track Number

This field specifies the track number of the track being created. The track number is placed into the track header and tailer information (see section 6.1).

If you specify track #0, the program recognizes the first file in your batch list as being your program's boot code. The track file created follows the format shown in *Figure 7-C*. Also, the *Boot Code Load/Exec Address* and *Boot Code Size* fields become visible.

For any track number other than zero, the track file created follows the format shown in *Figure 7-D*. The *Boot Code Load/Exec Address* and *Boot Code Size* fields are removed from the screen.

7.2.3.2 Boot Code Load/Exec Address

When you have specified track #0, this field allows you to specify the desired load address for the code in the first file listed in your batch.

When a track other than #0 is specified, this field is not available.

7.2.3.3 End of Track Padding

This field allows you to specify the desired amount of extra padding information that will be added at the end of the track.

7.2.3.4 Boot Code Size

When you have specified track #0, this field allows you to specify the length of the code and data in the boot code contained in the first file listed in your batch. This value is placed into the boot track header that the program creates for the file (see section 6.1).

When a track other than #0 is specified, this field is not available.

7.2.4 The Menu Bar

The menu bar of the ***Jaguar CD Track Creator*** allows you to set options that control how the program operates, begin processing a track file, or quit from the program.

7.2.4.1 File Menu

Figure 7-I shows the program's *File* menu.

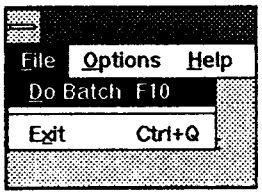


Figure 7-I — *The* File *Menu*

The *Do Batch* item in the *File* menu causes the ***Jaguar CD Track Creator*** to start processing the specified batch file (see section 7.2.1.1) and create the desired output files. A status window will be shown to display the ongoing status of the track creation and information about any errors that may occur.

The *Exit* item in the *File* menu causes the ***Jaguar CD Track Creator*** program to quit.

7.2.4.2 Options Menu

Figure 7-J shows the program's *Options* menu.

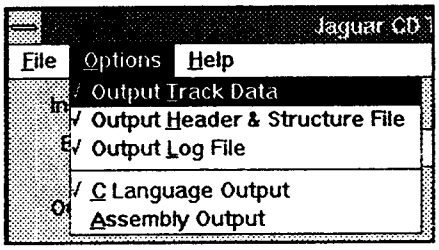


Figure 7-J — *The* Options *Menu*

When checked, the *Output Track Data* item in the *Options* menu causes the ***Jaguar CD Track Creator*** to merge the source files specified by your batch file into a new track data file suitable for CD mastering, as described in section 7.2.2.1. When unchecked, the track data file is not created.

When checked, the *Output Header & Structure Files* item in the *Options* menu causes the ***Jaguar CD Track Creator*** to create the files described in sections 7.2.2.2 and 7.2.2.3. When the menu item is unchecked, these files are not created.

When checked, the *Output Log File* item in the *Options* menu causes the ***Jaguar CD Track Creator*** to create the log file described in section 7.2.2.4. When the menu item is unchecked, this file is not created.

The status of the *C Language Output* and *Assembly Output* menu items determine what file format is used to create the files described in sections 7.2.2.2 and 7.2.2.3. Only one item can be checked at a time.

 © 1995 Atari Corp.