

trackdisk

COLLABORATORS

	<i>TITLE :</i> trackdisk		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	trackdisk	1
1.1	trackdisk.doc	1
1.2	trackdisk.device/CMD_CLEAR	1
1.3	trackdisk.device/CMD_READ	2
1.4	trackdisk.device/CMD_UPDATE	2
1.5	trackdisk.device/CMD_WRITE	3
1.6	trackdisk.device/TD_ADDCHANGEINT	4
1.7	trackdisk.device/TD_CHANGENUM	5
1.8	trackdisk.device/TD_CHANGESTATE	5
1.9	trackdisk.device/TD_EJECT	5
1.10	trackdisk.device/TD_FORMAT	6
1.11	trackdisk.device/TD_GETDRIVETYPE	7
1.12	trackdisk.device/TD_GETGEOMETRY	7
1.13	trackdisk.device/TD_GETNUMTRACKS	8
1.14	trackdisk.device/TD_MOTOR	8
1.15	trackdisk.device/TD_PROTSTATUS	9
1.16	trackdisk.device/TD_RAWREAD	9
1.17	trackdisk.device/TD_RAWWRITE	11
1.18	trackdisk.device/TD_REMCHANGEINT	12
1.19	trackdisk.device/TD_SEEK	13

Chapter 1

trackdisk

1.1 trackdisk.doc

CMD_CLEAR	TD_CHANGESTATE	TD_MOTOR
CMD_READ	TD_EJECT	TD_PROTSTATUS
CMD_UPDATE	TD_FORMAT	TD_RAWREAD
CMD_WRITE	TD_GETDRIVETYPE	TD_RAWWRITE
TD_ADDCHANGEINT	TD_GETGEOMETRY	TD_REMCHANGEINT
TD_CHANGENUM	TD_GETNUMTRACKS	TD_SEEK

1.2 trackdisk.device/CMD_CLEAR

NAME

CMD_CLEAR/ETD_CLEAR -- mark the track buffer as containing invalid data.

FUNCTION

These commands mark the track buffer as invalid, forcing a reread of the disk on the next operation. ETD_UPDATE or CMD_UPDATE would be used to force data out to the disk before turning the motor off. ETD_CLEAR or CMD_CLEAR are usually used after having locked out the trackdisk.device via the use of the disk resource, when you wish to prevent the track from being updated, or when you wish to force the track to be re-read. ETD_CLEAR or CMD_CLEAR will not do an update, nor will an update command do a clear.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CMD_CLEAR or ETD_CLEAR
io_Flags	0 or IOF_QUICK
iotd_Count	(ETD_CLEAR only) maximum allowable change counter value.

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in <devices/trackdisk.h>

SEE ALSO
 CMD_WRITE, CMD_UPDATE

1.3 trackdisk.device/CMD_READ

NAME

CMD_READ/ETD_READ -- read sectors of data from a disk.

FUNCTION

These commands transfer data from the track buffer to a supplied buffer. If the desired sector is already in the track buffer, no disk activity is initiated. If the desired sector is not in the buffer, the track containing that sector is automatically read in. If the data in the current track buffer has been modified, it is written out to the disk before a new track is read. ETD_READ will read the sector label area if the `iotd_SecLabel` is non-NULL.

IO REQUEST INPUT

<code>io_Device</code>	preset by the call to <code>OpenDevice()</code>
<code>io_Unit</code>	preset by the call to <code>OpenDevice()</code>
<code>io_Command</code>	CMD_READ or ETD_READ
<code>io_Flags</code>	0 or IOF_QUICK
<code>io_Data</code>	pointer to the buffer where the data should be put
<code>io_Length</code>	number of bytes to read, must be a multiple of TD_SECTOR.
<code>io_Offset</code>	byte offset from the start of the disk describing where to read data from, must be a multiple of TD_SECTOR.
<code>iotd_Count</code>	(ETD_READ only) maximum allowable change counter value.
<code>iotd_SecLabel</code>	(ETD_READ only) NULL or sector label buffer pointer. If provided, the buffer must be a multiple of TD_LABELSIZE.

IO REQUEST RESULT

`io_Error` - 0 for success, or an error code as defined in `<devices/trackdisk.h>`

NOTES

Under versions of Kickstart earlier than V36, the `io_Data` had to point to a buffer in chip memory. This restriction is no longer present as of Kickstart V36 and beyond.

SEE ALSO
 CMD_WRITE

1.4 trackdisk.device/CMD_UPDATE

NAME

CMD_UPDATE/ETD_UPDATE -- write out the track buffer if it is dirty.

FUNCTION

The trackdisk device does not write data sectors unless it is

necessary (you request that a different track be used) or until the user requests that an update be performed. This improves system speed by caching disk operations. These commands ensure that any buffered data is flushed out to the disk. If the track buffer has not been changed since the track was read in, these commands do nothing. ETD_UPDATE command checks for diskchange.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CMD_UPDATE or ETD_UPDATE
io_Flags	0 or IOF_QUICK
iotd_Count	(ETD_UPDATE only) maximum allowable change counter value.

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
----------	-----------------------------------------------------------------------

SEE ALSO

CMD_WRITE

1.5 trackdisk.device/CMD_WRITE

NAME

CMD_WRITE/ETD_WRITE -- write sectors of data to a disk.

FUNCTION

These commands transfer data from a supplied buffer to the track buffer. If the track that contains this sector is already in the track buffer, no disk activity is initiated. If the desired sector is not in the buffer, the track containing that sector is automatically read in. If the data in the current track buffer has been modified, it is written out to the disk before the new track is read in for modification. ETD_WRITE will write the sector label area if iotd_SecLabel is non-NULL.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	CMD_WRITE or ETD_WRITE
io_Flags	0 or IOF_QUICK
io_Data	pointer to the buffer where the data should be put
io_Length	number of bytes to write, must be a multiple of TD_SECTOR.
io_Offset	byte offset from the start of the disk describing where to write data to, must be a multiple of TD_SECTOR.
iotd_Count	(ETD_WRITE only) maximum allowable change counter value.
iotd_SecLabel	(ETD_WRITE only) NULL or sector label buffer pointer. If provided, the buffer must be a multiple of TD_LABELSIZE.

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in
<devices/trackdisk.h>

NOTES

Under versions of Kickstart earlier than V36, the io_Data had to point to a buffer in chip memory. This restriction is no longer present as of Kickstart V36 and beyond.

SEE ALSO

CMD_READ, TD_FORMAT

1.6 trackdisk.device/TD_ADDCHANGEINT

NAME

TD_ADDCHANGEINT -- add a disk change software interrupt handler.

FUNCTION

This command lets you add a software interrupt handler to the disk device that gets invoked whenever a disk insertion or removal occurs.

You must pass in a properly initialized Exec Interrupt structure and be prepared to deal with disk insertions/removals immediately. From within the interrupt handler, you may only call the status commands that can use IOF_QUICK.

To set up the handler, an Interrupt structure must be initialized. This structure is supplied as the io_Data to the TD_ADDCHANGEINT command. The handler then gets linked into the handler chain and gets invoked whenever a disk change happens. You must eventually remove the handler before you exit.

This command only returns when the handler is removed. That is, the device holds onto the IO request until the TD_REMCHANGEINT command is executed with that same IO request. Hence, you must use SendIO() with this command.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_ADDCHANGEINT
io_Flags	0
io_Length	sizeof(struct Interrupt)
io_Data	pointer to Interrupt structure

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in
<devices/trackdisk.h>

SEE ALSO

TD_REMCHANGEINT, <devices/trackdisk.h>, <exec/interrupts.h>, exec.library/Cause()

1.7 trackdisk.device/TD_CHANGENUM

NAME

TD_CHANGENUM -- return the current value of the disk-change counter.

FUNCTION

This command returns the current value of the disk-change counter (as used by the enhanced commands). The disk change counter is incremented each time a disk is inserted or removed from the trackdisk unit.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_CHANGENUM
io_Flags	0 or IOF_QUICK

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
io_Actual	- if io_Error is 0, this contains the current value of the disk-change counter.

1.8 trackdisk.device/TD_CHANGESTATE

NAME

TD_CHANGESTATE -- check if a disk is currently in a drive.

FUNCTION

This command checks to see if there is currently a disk in a drive.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_CHANGESTATE
io_Flags	0 or IOF_QUICK

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
io_Actual	- if io_Error is 0, this tells you whether a disk is in the drive. 0 means there is a disk, while anything else indicates there is no disk.

1.9 trackdisk.device/TD_EJECT

NAME

TD_EJECT -- eject the disk in the drive, if possible.

FUNCTION

This command causes the drive to attempt to eject the disk in it, if any. Note that the current trackdisk.device does not implement this command, but it might in the future, and other

trackdisk-compatible drivers may implement this command.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_EJECT
io_Flags	0 or IOF_QUICK

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
----------	--------------------------------------------------------------------------

1.10 trackdisk.device/TD_FORMAT

NAME

TD_FORMAT/ETD_FORMAT -- format a track on a disk.

FUNCTION

These commands are used to write data to a track that either has not yet been formatted or has had a hard error on a standard write command. TD_FORMAT completely ignores all data currently on a track and does not check for disk change before performing the command. The io_Data field must point to at least one track worth of data. The io_Offset field must be track aligned, and the io_Length field must be in units of track length (that is, NUMSEC*TD_SECTOR).

The device will format the requested tracks, filling each sector with the contents of the buffer pointed to by io_Data. You should do a read pass to verify the data.

If you have a hard write error during a normal write, you may find it possible to use the TD_FORMAT command to reformat the track as part of your error recovery process. ETD_FORMAT will write the sector label area if iotd_SecLabel is non-NULL.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_FORMAT or ETD_FORMAT
io_Flags	0 or IOF_QUICK
io_Data	points to a buffer containing the data to write to the track, must be at least as large as io_Length.
io_Length	number of bytes to format, must be a multiple of (TD_SECTORS * NUMSEC).
io_Offset	byte offset from the start of the disk for the track to format, must be a multiple of (TD_SECTORS * NUMSEC).
iotd_Count	(ETD_FORMAT only) maximum allowable change counter value.
iotd_SecLabel	(ETD_FORMAT only) NULL or sector label buffer pointer. If provided, the buffer must be a multiple of (TD_LABELSIZE * NUMSEC).

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
----------	--------------------------------------------------------------------------

NOTES

Under versions of Kickstart earlier than V36, the `io_Data` had to point to a buffer in chip memory. This restriction is no longer present as of Kickstart V36 and beyond.

SEE ALSO

`CMD_WRITE`, `TD_RAWWRITE`

1.11 trackdisk.device/TD_GETDRIVETYPE

NAME

`TD_GETDRIVETYPE` -- return the type of disk drive for the unit that was opened.

FUNCTION

This command returns the type of the disk drive to the user. This number will be a small integer and will come from the set of `DRIVEXXX` constants defined in `<devices/trackdisk.h>`.

The only way you can actually use this command is if the trackdisk device understands the drive type of the hardware that is plugged in. This is because the `OpenDevice()` call will fail if the trackdisk device does not understand the drive type. To find raw drive identifiers see the `disk.resource`'s `DR_GETUNITID` entry point.

IO REQUEST INPUT

<code>io_Device</code>	preset by the call to <code>OpenDevice()</code>
<code>io_Unit</code>	preset by the call to <code>OpenDevice()</code>
<code>io_Command</code>	<code>TD_GETDRIVETYPE</code>
<code>io_Flags</code>	0 or <code>IOF_QUICK</code>

IO REQUEST RESULT

<code>io_Error</code>	- 0 for success, or an error code as defined in <code><devices/trackdisk.h></code>
<code>io_Actual</code>	- if <code>io_Error</code> is 0 this contains the drive type connected to this unit.

SEE ALSO

`TD_GETNUMTRACKS`, `<devices/trackdisk.h>`

1.12 trackdisk.device/TD_GETGEOMETRY

NAME

`TD_GETGEOMETRY` -- return the geometry of the drive.

FUNCTION

This command returns a full set of information about the layout of the drive. The information is returned in the `DriveGeometry` structure pointed to by `io_Data`.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_GETGEOMETRY
io_Flags	0 or IOF_QUICK
io_Data	Pointer to a DriveGeometry structure
io_Length	sizeof(struct DriveGeometry)

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in
<devices/trackdisk.h>

NOTE

This information may change when a disk is inserted when certain hardware is present.

SEE ALSO

TD_GETDRIVETYPE, TD_GETNUMTRACKS

1.13 trackdisk.device/TD_GETNUMTRACKS

NAME

TD_GETNUMTRACKS -- return the number of tracks for the type of disk drive for the unit that was opened.

FUNCTION

This command returns the number of tracks that are available on the disk unit.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_GETNUMTRACKS
io_Flags	0 or IOF_QUICK

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in
<devices/trackdisk.h>
io_Actual - if io_Error is 0 this contains the drive type connected to this unit.

SEE ALSO

TD_GETDRIVETYPE

1.14 trackdisk.device/TD_MOTOR

NAME

TD_MOTOR/ETD_MOTOR -- control the on/off state of a drive motor.

FUNCTION

This command gives control over the disk motor. The motor may be turned on or off. When it is on, the drive light automatically turns on as well.

If the motor is just being turned on, the device will delay the proper amount of time to allow the drive to come up to speed. Normally, turning the drive on is not necessary, the device does this automatically if it receives a request when the motor is off. However, turning the motor off is the programmer's responsibility.

In addition, the standard instructions to the user are that it is safe to remove a disk from a drive if and only if the motor is off (that is, if the disk light is off).

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_MOTOR or ETD_MOTOR
io_Flags	0 or IOF_QUICK
io_Length	the requested state of the motor, 0 to turn the motor off, and 1 to turn the motor on.
iotd_Count	(ETD_MOTOR only) maximum allowable change counter value.

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
io_Actual	- if io_Error is 0 this contains the previous state of the drive motor.

1.15 trackdisk.device/TD_PROTSTATUS

NAME

TD_PROTSTATUS -- return whether the current disk is write-protected.

FUNCTION

This command is used to determine whether the current disk is write-protected.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_PROTSTATUS
io_Flags	0 or IOF_QUICK

IO REQUEST RESULT

io_Error	- 0 for success, or an error code as defined in <devices/trackdisk.h>
io_Actual	- if io_Error is 0, this tells you whether the disk in the drive is write-protected. 0 means the disk is NOT write-protected, while any other value indicates it is.

1.16 trackdisk.device/TD_RAWREAD

NAME

TD_RAWREAD/ETD_RAWREAD -- read raw data from the disk.

FUNCTION

These commands read a track of raw data from disk and deposits it in the provided buffer. The data is taken straight from the disk with no processing done on it. It will appear exactly as the bits come out off the disk, hopefully in some legal MFM format.

This interface is intended for sophisticated programmers only. Commodore-Amiga reserves the right to make enhancements to the disk format in the future. We will provide compatibility via the CMD_READ and ETD_READ commands, anyone using TD_RAWREAD is bypassing this upwards compatibility, and may thus stop working.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_RAWREAD or ETD_RAWREAD.
io_Flags	if the IOTDB_INDEXSYNC bit is set then the driver will make a best effort attempt to start reading from the index mark. Note that there will be at least some delay, and perhaps a great deal of delay (for example if interrupts have been disabled).
io_Length	Length of buffer in bytes, with a maximum of 32768 bytes.
io_Data	Pointer to CHIP memory buffer where raw track data is to be deposited.
io_Offset	The number of the track to read in.
iotd_Count	(ETD_RAWREAD only) maximum allowable change counter value.

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in
<devices/trackdisk.h>

NOTES

The track buffer provided MUST be in CHIP memory

There is a delay between the index pulse and the start of bits coming in from the drive (e.g. dma started). This delay is in the range of 135-200 microseconds. This delay breaks down as follows: 55 microsecs is software interrupt overhead (this is the time from interrupt to the write of the DSKLEN register). 66 microsecs is one horizontal line delay (remember that disk IO is synchronized with agnus' display fetches). The last variable (0-65 microsecs) is an additional scan line since DSKLEN is poked anywhere in the horizontal line. This leaves 15 microsecs unaccounted for... Sigh.

In short, You will almost never get bits within the first 135 microseconds of the index pulse, and may not get it until 200 microseconds. At 4 microsecs/bit, this works out to be between 4 and 7 bytes of user data of delay.

BUGS

This command does not work reliably under versions of Kickstart earlier than V36, especially on systems with 1 floppy drive.

SEE ALSO

TD_RAWWRITE

1.17 trackdisk.device/TD_RAWWRITE

NAME

TD_RAWWRITE/ETD_RAWWRITE -- write raw data to the disk.

FUNCTION

This command writes a track of raw data from the provided buffer to the specified track on disk. The data is copied straight to the disk with no processing done on it. It will appear exactly on the disk as it is in the memory buffer, hopefully in a legal MFM format.

This interface is intended for sophisticated programmers only. Commodore-Amiga reserves the right to make enhancements to the disk format in the future. We will provide compatibility via the CMD_WRITE and ETD_WRITE commands, anyone using TD_RAWWRITE is bypassing this upwards compatibility, and may thus stop working.

IO REQUEST INPUT

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_RAWWRITE or ETD_RAWWRITE.
io_Flags	if the IOTDB_INDEXSYNC bit is set then the driver will make a best effort attempt to start writing from the index mark. Note that there will be at least some delay, and perhaps a great deal of delay (for example if interrupts have been disabled).
io_Length	Length of buffer in bytes, with a maximum of 32768 bytes.
io_Data	Pointer to CHIP memory buffer where raw track data is to be taken.
io_Offset	The number of the track to write to.
iotd_Count	(ETD_RAWWRITE only) maximum allowable change counter value.

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in <devices/trackdisk.h>

NOTES

The track buffer provided MUST be in CHIP memory

There is a delay between the index pulse and the start of bits going out to the driver (e.g. write gate enabled). This delay is in the range of 135–200 microseconds. This delay breaks down as follows: 55 microseconds is software interrupt overhead (this is the time from interrupt to the write of the DSKLEN register). 66 microseconds is one horizontal line delay (remember that disk IO is synchronized with agnus' display fetches). The last variable (0–65 microseconds) is an additional scan line

since DSKLEN is poked anywhere in the horizontal line. This leaves 15 microseconds unaccounted for... Sigh.

In short, You will almost never get bits within the first 135 microseconds of the index pulse, and may not get it until 200 microseconds. At 4 microseconds/bit, this works out to be between 4 and 7 bytes of user data of delay.

BUGS

This command does not work reliably under versions of Kickstart earlier than V36, especially on systems with 1 floppy drive.

SEE ALSO

TD_RAWREAD

1.18 trackdisk.device/TD_REMCHANGEINT

NAME

TD_REMCHANGEINT -- remove a disk change software interrupt handler.

FUNCTION

This command removes a disk change software interrupt added by a previous use of TD_ADDCHANGEINT.

IO REQUEST INPUT

The same IO request used for TD_ADDCHANGEINT.

io_Device	preset by the call to OpenDevice()
io_Unit	preset by the call to OpenDevice()
io_Command	TD_REMCHANGEINT
io_Flags	0
io_Length	sizeof(struct Interrupt)
io_Data	pointer to Interrupt structure

IO REQUEST RESULT

io_Error - 0 for success, or an error code as defined in <devices/trackdisk.h>

BUGS

This command did not function properly under versions of Kickstart earlier than V36. A valid workaround under these older versions of Kickstart is:

```
Forbid();
Remove(ioRequest);
Permit();
```

Do not use this workaround in versions of Kickstart >= V36, use TD_REMCHANGEINT instead (for future compatibility with V38+).

SEE ALSO

TD_ADDCHANGEINT, <devices/trackdisk.h>

1.19 trackdisk.device/TD_SEEK

NAME

TD_SEEK/ETD_SEEK -- control positioning of the drive heads.

FUNCTION

These commands are currently provided for internal diagnostics, disk repair, and head cleaning only.

TD_SEEK and ETD_SEEK move the drive heads to the track specified. The `io_Offset` field should be set to the (byte) offset to which the seek is to occur. TD_SEEK and ETD_SEEK do not verify their position until the next read. That is, they only move the heads; they do not actually read any data.

IO REQUEST INPUT

<code>io_Device</code>	preset by the call to <code>OpenDevice()</code>
<code>io_Unit</code>	preset by the call to <code>OpenDevice()</code>
<code>io_Command</code>	TD_SEEK or ETD_SEEK
<code>io_Flags</code>	0 or <code>IOF_QUICK</code>
<code>io_Offset</code>	byte offset from the start of the disk describing where to move the head to.
<code>iotd_Count</code>	(ETD_SEEK only) maximum allowable change counter value.

IO REQUEST RESULT

`io_Error` - 0 for success, or an error code as defined in `<devices/trackdisk.h>`
