trackdisk

COLLABORATORS

	TITLE : trackdisk		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 14, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

trackdisk

1.1 trackdisk.doc

```
TD_ADDCHANGEINT

TD_GETDRIVETYPE

TD_RAWREAD

TD_CHANGENUM

TD_GETNUMTRACKS

TD_RAWWRITE

TD_CHANGESTATE

TD_MOTOR

TD_REMCHANGEINT

TD_FORMAT

TD_PROTSTATUS

TD_SEEK
```

1.2 trackdisk.device/TD_ADDCHANGEINT

```
NAME
    TD_ADDCHANGEINT - add a new change software int

SYNOPSIS
    TDUAddChangeInt( IORequest ), UnitPtr
                     A1           A3

FUNCTION
```

```
    Alas, the old TDURemove call was not robust enough.  This routine
    supports an extensible list of software interrupts for use by many
    different supporting drivers.

    The call does not "complete" (e.g. TermIO).  The request
    is stashed until TDURemChangeInt is called, when it is
    finally replied.

INPUTS
    IORequest - a standard IO Request block (IO_DATA-> soft int struct).

RESULTS

EXCEPTIONS

SEE ALSO

BUGS
```

1.3 trackdisk.device/TD_CHANGENUM

```
NAME
    TD_CHANGENUM - return the current disc change number

SYNOPSIS
    TDUChangeNum( IORequest ), UnitPtr
                  A1           A3

FUNCTION
    This routine checks to see if there is a disc in the drive
    of the specified unit.

INPUTS
    IORequest - a standard IO Request block

RESULTS

EXCEPTIONS

SEE ALSO

BUGS
```

1.4 trackdisk.device/TD_CHANGESTATE

```
NAME
    TD_CHANGESTATE - Return the current state of the disc

SYNOPSIS
    TDUChangeState( IORequest ), UnitPtr
                    A1           A3
```

```
FUNCTION
    This routine checks to see if there is a disc in the drive
    one the specified unit.

INPUTS
    IORequest - a standard IO Request block

RESULTS
    IO_ACTUAL -- nonzero if there is no diskette in the drive

EXCEPTIONS

SEE ALSO

BUGS
```

1.5 trackdisk.device/TD_FORMAT

```
NAME
    TD_FORMAT -- format the entire disc

SYNOPSIS
    TDUFormat( iOBlock ), DevNode
    D0         A1         A6

FUNCTION
    The function formats the entire disc, destroying all data.
    It fills all the sectors with the contents of the iOBlock.
    The iOBlock must point to (at least) one sector worth of
    information.  Any info greater than one sector is ignored.
    NO ERROR CHECKING is done

INPUTS

RESULTS

SEE ALSO
```

1.6 trackdisk.device/TD_GETDRIVETYPE

```
                NAME
    TD_GETDRIVETYPE - return the type of the disk drive to the user

FUNCTION
    This routine returns the type of the disk to the user.
    This number will be a small integer.  It will come from
    the set of DRIVE... defines in trackdisk.h
    or trackdisk.i.

    The only way you can get to this call 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
    io_Command      TD_GETDRIVETYPE

RESULTS
    io_Actual       the drive type connected to this unit.

SEE ALSO

                TD_GETNUMTRACKS
```

1.7 trackdisk.device/TD_GETNUMTRACKS

```
                NAME
    TD_GETNUMTRACKS - return the number of tracks on this type of disk

FUNCTION
    This routine returns the number of tracks that are available
    on this disk unit.  This call obsoletes the older NUMTRACKS
    hard coded constant.

IO REQUEST
    io_Command      TD_GETNUMTRACKS

RESULTS
    io_Actual       number of tracks accessible on this unit

SEE ALSO

                TD_GETDRIVETYPE
```

1.8 trackdisk.device/TD_MOTOR

```
NAME
    TD_MOTOR - user visible control for motor

SYNOPSIS
    TDUMotor( IOBlock ), UnitPtr, DevPtr
              A1         A3       A6

FUNCTION
    This routine allows the user to control the disc motor. He
    may turn it either on or off.  Note that the motor will be
    automatically turned on during an I/O request, but is never
    turned of except by this command.

INPUTS
```

```
    IOBlock - the command block for this IO operation.
        IO_ACTUAL -- returns the previous state of the motor
        IO_LENGTH -- the requested state of the motor
            0 ==> turn motor off
            1 ==> turn motor on

EXCEPTIONS

SEE ALSO

BUGS
```

1.9 trackdisk.device/TD_PROTSTATUS

```
NAME
    TD_PROTSTATUS -- return whether the current disk is write protected

SYNOPSIS
    TDUProtstatus( IOBlock ), UnitPtr, DevPtr
                   A1          A3       A6

FUNCTION
    This routine tells whether the current disk is write protected.

INPUTS
    IOBlock - the command block for this IO operation.
            IO_ACTUAL - nonzero if the disk is protected, 0 otherwise
            If there is no disk in the drive, then IO_ERROR is set
            to TDERR_DiskChanged

EXCEPTIONS

SEE ALSO

BUGS
```

1.10 trackdisk.device/TD_RAWREAD

```
                NAME
    TD_RAWREAD - read a raw sector from the disk

FUNCTION
    This routine performs a raw read for the track disk.
    It seeks to the specified track and reads it in to the
    user's buffer.  This buffer MUST be in chip memory.

    NO PROCESSING OF THE TRACK IS DONE.  It will appear exactly
    as the bits come out off the disk -- hopefully in some legal MFM
    format (if you don't know what MFM is, you shouldn't be using
    this call...).  Caveat Programmer.

    This interface is intended for sophisticated programmers
    only.  Commodore-Amiga may make enhancements to the disk
    format in the future.  We will provide compatibility
    within the trackdisk device.  Anyone who uses this routine
    is bypassing this upwards compatibility.  If your application
    breaks, TOUGH!

    If this warning is not enough, then add suitable additional
    harrassment of your choice.

IO REQUEST
    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 (if, for example, interrupts have
                      been Disabled()...).
    io_Command      TD_RAWREAD or ETD_RAWREAD.
    io_Length       Length of buffer (in bytes).  The maximum allowable
                      length is 32K bytes.
    io_Data         Pointer to buffer in chip memory where raw track
                      will be read into.
    io_Offset       The track number to read in (not this is different
                      from a normal trackdisk io call which is given
                      in terms of logical bytes from the beginning of
                      the disk.  This is because the trackdisk driver
                      has no idea what the format of the disk is).
    iotd_Count      (ETD_RAWREAD only) maximum allowable change counter
                      value

RESULTS
    io_Error        non-zero if there was an error

LIMITATIONS for synced reads and writes
    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 micro seconds.  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 withing 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.

SEE ALSO

                TD_RAWWRITE
```

1.11 trackdisk.device/TD_RAWWRITE

```
                NAME
    TD_RAWWRITE - write a raw sector to the disk

FUNCTION
    NO PROCESSING OF THE TRACK IS DONE.  The disk will appear exactly
    as the bits come out of memory -- hopefully in some legal MFM
    format (if you don't know what MFM is, you shouldn't be using
    this call...).  Caveat Programmer.

    NO PROCESSING OF THE TRACK IS DONE.  It will exactly
    as the bits come out off the disk.  Caveat Programmer.

    This interface is intended for sophisticated programmers
    only.  Commodore-Amiga may make enhancements to the disk
    format in the future.  We will provide compatibility
    within the trackdisk device.  Anyone who uses this routine
    is bypassing this upwards compatibility.  If your application
    breaks, TOUGH!

    If this warning is not enough, then add suitable additional
    harrassment of your choice.

IO REQUEST
    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 (if, for example, interrupts have
                      been Disabled()...).
    io_Command      TD_RAWWRITE or ETD_RAWWRITE.
    io_Length       Length of buffer (in bytes).  The maximum allowable
                      length is 32K bytes.
    io_Data         Pointer to buffer in chip memory where raw track
                      will be read into.
    io_Offset       The track number to read in (not this is different
                      from a normal trackdisk io call which is given
                      in terms of logical bytes from the beginning of
                      the disk.  This is because the trackdisk driver
                      has no idea what the format of the disk is).
    iotd_Count      (ETD_RAWWRITE only) maximum allowable change counter
                      value

RESULTS
    io_Error        non-zero if there was an error

LIMITATIONS for synced reads and writes
    There is a delay between the index pulse and the start of bits
    going out to the drive (e.g. write gate enabled).  This delay
    is in the range of 135-200 micro seconds.  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 withing 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.

SEE ALSO

                TD_RAWREAD
```

1.12 trackdisk.device/TD_REMCHANGEINT

```
NAME
    TD_REMCHANGEINT - remove a change software int

SYNOPSIS
    TDURemChangeInt( IORequest ), UnitPtr
                     A1           A3

FUNCTION
    This function unlinks the IOReqest stashed by AddChangeInt.
    It also replies it to the user.

INPUTS
    IORequest - a standard IO Request block

RESULTS

EXCEPTIONS

SEE ALSO

BUGS
```

1.13 trackdisk.device/TD_SEEK

```
NAME
    TD_SEEK - user visible control for the heads

SYNOPSIS
    TDUSeek( IOBlock ), TDLib
              A1         A6

FUNCTION
    This routine allows the user to control the seek position.
    Note that the heads will be automatically seeked during an
    I/O request; this command allows the heads to be preseeked
    if the next position is known prior to the I/O being ready.
```

```
INPUTS
    IOBlock - the command block for this IO operation.
        IO_OFFSET -- the location to seek to

EXCEPTIONS

SEE ALSO

BUGS
```