USB TC-08 C API Reference Manual

Generated by Doxygen 1.4.7

Thu Aug 2 10:19:42 2007

Contents

1 USB TC-08 C API File Index **1**

1.1 USB TC-08 C API File List 1

2 USB TC-08 C API File Documentation **3**

2.1 TC08Api.cpp File Reference 3

Chapter 1

USB TC-08 C API File Index

1.1 USB TC-08 C API File List

Here is a list of all documented files with brief descriptions:

TC08Api.cpp (C API to the USB TC-08 driver) 3

Chapter 2

USB TC-08 C API File Documentation

2.1 TC08Api.cpp File Reference

C API to the USB TC-08 driver.

Functions

- short usb_tc08_open_unit (void)

  Open a USB TC-08 unit for access by the driver.

- short usb_tc08_open_unit_async (void)

  Open a USB TC-08 unit without locking the calling thread while firmware is downloaded.

- short usb_tc08_open_unit_progress (short *handle, short *percent_progress)

  Determine the progress of an async opening operation.

- short usb_tc08_close_unit (short handle)

  Close the TC08 with the specified handle.

- short usb_tc08_stop (short handle)

  Stop the TC08 running in streaming mode.

- short usb_tc08_set_mains (short handle, short sixty_hertz)

  Set up mains interference rejection.

- long usb_tc08_get_minimum_interval_ms (short handle)

  Find the fastest possible sampling rate in streaming mode with the currently enabled set of channels.

- short usb_tc08_get_unit_info (short handle, USBTC08_INFO *info)

  Get some information about the USB TC-08 unit in the form of a USBTC08_INFO structure.

- short usb_tc08_get_unit_info2 (short handle, char *string, short string_length, short line)

  Get some information about the USB TC-08 unit in the form of a string representing a single line of the USBTC08_INFO structure.

- short usb_tc08_get_formatted_info (short handle, char *unit_info, short string_length)

  Get some information about the USB TC-08 unit in the form of a string representing a the whole USBTC08_INFO structure.

- short usb_tc08_get_last_error (short handle)

  Get a more detailed return code describing the status of the last function call.

- short usb_tc08_set_channel (short handle, short channel, char tc_type)

  Set up one channel of the USB TC-08 for thermocouple (or voltage) measurements.

- long usb_tc08_run (short handle, long interval_ms)

  Set a TC-08 unit running in streaming mode.

- short usb_tc08_get_single (short handle, float *temp, short *overflow_flags, short units)

  Get a single reading on all enabled channels of a USB TC-08.

- long usb_tc08_get_temp (short handle, float *temp_buffer, long *times_ms_buffer, long buffer_length, short *overflow, short channel, short units, short fill_missing)

  Retrieve buffered readings acquired in streaming mode from the driver.

- long usb_tc08_get_temp_deskew (short handle, float *temp_buffer, long *times_ms_buffer, long buffer_length, short *overflow, short channel, short units, short fill_missing)

  Retrieve buffered readings acquired in streaming mode from the driver.

2.1.1 Detailed Description

C API to the USB TC-08 driver.

2.1.2 Function Documentation

2.1.2.1 short usb_tc08_open_unit (void)

Open a USB TC-08 unit for access by the driver.

If multiple units are connected, the function may open any unit which is not already in use: call it repeatedly to open all available units. The function blocks while firmware is downloaded to the unit - for a non-blocking alternative, see usb_tc08_open_unit_async.

Returns:

A positive short integer handle to the TC-08 if it has been successfully opened. Zero if there are no more TC-08s available for opening, and -1 if an error occurs when attempting to open the unit.

2.1.2.2 short usb_tc08_open_unit_async (void)

Open a USB TC-08 unit without locking the calling thread while firmware is downloaded.

This function is used instead of usb_tc08_open_unit() if the calling application needs to perform operations while waiting for a unit to open. The TC-08 unit is opened in the background and the function returns immediately. Use usb_tc08_open_unit_progress to find out when opening has completed and get the handle to the device

Returns:

1 if the opening operation started successfully, or 0 if it could not be started. Call usb_tc08_get_last_-error in this case to get more information

2.1.2.3 short usb_tc08_open_unit_progress (short * *handle*, short * *percent_progress*)

Determine the progress of an async opening operation.

Call this function after usb_tc08_open_unit_async to determine whether that operation has completed.

Parameters:

handle Pass in a pointer to a location in which the handle of the opened unit can be stored

percent_progress Pointer to a location in which the opening progress from 0 to 100% can be stored. Note that on Linux due to the nature of the download process the only progress values that can be retuned are 0% (not complete) and 100% (complete).

Returns:

USBTC08_PROGRESS_PENDING if a unit is still being opened, USBTC08_PROGRESS_-COMPLETE if the opening attempt has completed successfully or USBTC08_PROGRESS_FAIL if the opening failed or an error was made calling this function (handle NULL, or no async open operation in progress). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.4 short usb_tc08_close_unit (short *handle*)

Close the TC08 with the specified handle.

The handle becomes invalid and may be reused when subsequent TC-08 units are opened.

Parameters:

handle Handle identifying the unit to be closed

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.5 short usb_tc08_stop (short *handle*)

Stop the TC08 running in streaming mode.

Parameters:

handle Handle identifying the unit to be stopped

Returns:

1 for success, 0 for error (handle not valid, or operation failed, e.g. unit was not in streaming mode). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.6 short usb_tc08_set_mains (short *handle*, short *sixty_hertz*)

Set up mains interference rejection.

This changes the sampling time of the ADC in the TC08 to be a whole number of mains cycles, ensuring that mains-related interference is cancelled out.

Parameters:

handle Handle identifying the unit to be set

sixty_hertz USBTC08_MAINS_FREQUENCY value to set the mains frequency for rejection. This will vary by country, see USBTC08_MAINS_FREQUENCY

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.7 long usb_tc08_get_minimum_interval_ms (short *handle*)

Find the fastest possible sampling rate in streaming mode with the currently enabled set of channels.

Set up all channels before calling this. The allowable sampling interval will be 100ms $*$ number of channels enabled, including the CJC if needed.

Parameters:

handle Handle identifying the unit to be set

Returns:

The fastest allowable sampling rate in streaming mode with the currently enabled set of channels, in milliseconds. Returns 0 if an error occurs (no channels enabled), use usb_tc08_get_last_error

2.1.2.8 short usb_tc08_get_unit_info (short *handle*, USBTC08_INFO $*$ *info*)

Get some information about the USB TC-08 unit in the form of a USBTC08_INFO structure.

Parameters:

handle Handle identifying the unit to get info from. If you pass zero, only the driver version will be filled in and the function will succeed

info Pointer to USBTC08_INFO structure in which the information must be written. Your program must fill in the size of the USBTC08_INFO structure in the field size like this: info.size = sizeof(USBTC08_INFO) as a check against buffer overruns

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.9 short usb_tc08_get_unit_info2 (short *handle*, char * *string*, short *string_length*, short *line*)

Get some information about the USB TC-08 unit in the form of a string representing a single line of the USBTC08_INFO structure.

Parameters:

handle Handle identifying the unit to get info from. This can be zero only if asking for the driver version

string Pointer to char array in which the information must be written. Your program must fill in the size of the array in the string_length parameter

string_length The size of the array pointed to by string

line Line number of the information from USBTC08_INFO to be returned

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.10 short usb_tc08_get_formatted_info (short *handle*, char * *unit_info*, short *string_length*)

Get some information about the USB TC-08 unit in the form of a string representing a the whole USBTC08_INFO structure.

The string consists of multiple lines, each line represents a fild of USBTC08_INFO.

Parameters:

handle Handle identifying the unit to get info from. This can be zeroin which case only the driver version is valid

unit_info Pointer to char array in which the information must be written. Your program must fill in the size of the array in the string_length parameter

string_length The size of the array pointed to by unit_info

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.11 short usb_tc08_get_last_error (short *handle*)

Get a more detailed return code describing the status of the last function call.

Pass 0 as the unit handle to get the last reason that usb_tc08_open_unit() failed.

Parameters:

handle Handle identifying the unit to be queried

Returns:

USBTC08_ERROR value indicating the result of the last function call to the specified unit

2.1.2.12 short usb_tc08_set_channel (short *handle*, short *channel*, char *tc_type*)

Set up one channel of the USB TC-08 for thermocouple (or voltage) measurements.

Call this function once for each channel that you want to use. The cold jucntion compensation (CJC) channel is always enabled is any channels are set as thermocouples, or can be manually enabled.

Parameters:

handle Handle identifying the unit to be set

channel Character value identifying the channel to be set up, from 0 (CJC) to 8 (Channel 8)

tc_type Character value identifying the thermocouple type (B,E,J,K,N,R,S,T) for the selected channel, or X for voltage or a space to disable the channel.

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.13 long usb_tc08_run (short *handle*, long *interval_ms*)

Set a TC-08 unit running in streaming mode.

You must set up the channels to be converted using usb_tc08_set_channel() before calling this function.

Parameters:

handle Handle identifying the unit to be set running

interval_ms Sample interval in milliseconds to get one sample on all enabled channels. Must be between the values returned by GetMinimumInterval() and the maximum interval of 65535ms

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.14 short usb_tc08_get_single (short *handle*, float * *temp*, short * *overflow_flags*, short *units*)

Get a single reading on all enabled channels of a USB TC-08.

You must set up the channels to be converted using usb_tc08_set_channel() before calling this function, and the unit must not be in streaming mode

Parameters:

handle Handle identifying the unit from which to get the reading

temp Pointer to an array of 9 floats where the readings will be placed. Note that the array must always be of size 9 regardless of the number of channels enabled. Each reading is placed in the array location corresponding to its channel number, and disabled channels are filled with NaN values

overflow_flags Pointer to an unsigned short which the driver sets as a bit-field indicating whether each of the channels was over-range. The LSB represents the CJC and the next 8 bits represent the 8 thermocouple channels.

units USBTC08_UNITS value indicating the temperature unit in which to return the results

Returns:

1 for success, 0 for error (handle not valid, or operation failed). Call usb_tc08_get_last_error in this case to get more information.

2.1.2.15 long usb_tc08_get_temp (short *handle*, float * *temp_buffer*, long * *times_ms_buffer*, long *buffer_length*, short * *overflow*, short *channel*, short *units*, short *fill_missing*)

Retrieve buffered readings acquired in streaming mode from the driver.

Must be called at least once per minute to ensure no data is lost.

Parameters:

handle Handle identifying the unit from which to get the readings

temp_buffer Address of an array of floats to store the sampled temperature values

times_ms_buffer Address of an array of longs to store the sample times

buffer_length Length of the temp and times bufferss

overflow Pointer to a short which is set non-zero by the driver if a buffer overflow (data loss) has occurred

channel The number of the channel from which to get the readings. 0 = CJC, 1 = Channel 1, etc.

units USBTC08_UNITS value indicating the temperature scale to use when returning readings

fill_missing If true, any missin greadings are assigned the last known good value. Otherwise, they are assigned NaN

Returns:

The number of readings copied. Zero if no readings were available or -1 if an error occurs

2.1.2.16 long usb_tc08_get_temp_deskew (short *handle*, float * *temp_buffer*, long * *times_ms_buffer*, long *buffer_length*, short * *overflow*, short *channel*, short *units*, short *fill_missing*)

Retrieve buffered readings acquired in streaming mode from the driver.

Must be called at least once per minute to ensure no data is lost.

Parameters:

handle Handle identifying the unit from which to get the readings

temp_buffer Address of an array of floats to store the sampled temperature values

times_ms_buffer Address of an array of longs to store the sample times

buffer_length Length of the temp and times bufferss

overflow Pointer to a short which is set non-zero by the driver if a buffer overflow (data loss) has occurred

channel The number of the channel from which to get the readings. 0 = CJC, 1 = Channel 1, etc.

units USBTC08_UNITS value indicating the temperature scale to use when returning readings

fill_missing If true, any missin greadings are assigned the last known good value. Otherwise, they are assigned NaN

Returns:

The number of readings copied. Zero if no readings were available or -1 if an error occurs

Index

TC08Api.cpp, 3
    usb_tc08_close_unit, 5
    usb_tc08_get_formatted_info, 7
    usb_tc08_get_last_error, 7
    usb_tc08_get_minimum_interval_ms, 6
    usb_tc08_get_single, 8
    usb_tc08_get_temp, 9
    usb_tc08_get_temp_deskew, 9
    usb_tc08_get_unit_info, 6
    usb_tc08_get_unit_info2, 6
    usb_tc08_open_unit, 4
    usb_tc08_open_unit_async, 4
    usb_tc08_open_unit_progress, 5
    usb_tc08_run, 8
    usb_tc08_set_channel, 7
    usb_tc08_set_mains, 5
    usb_tc08_stop, 5

usb_tc08_close_unit
    TC08Api.cpp, 5
usb_tc08_get_formatted_info
    TC08Api.cpp, 7
usb_tc08_get_last_error
    TC08Api.cpp, 7
usb_tc08_get_minimum_interval_ms
    TC08Api.cpp, 6
usb_tc08_get_single
    TC08Api.cpp, 8
usb_tc08_get_temp
    TC08Api.cpp, 9
usb_tc08_get_temp_deskew
    TC08Api.cpp, 9
usb_tc08_get_unit_info
    TC08Api.cpp, 6
usb_tc08_get_unit_info2
    TC08Api.cpp, 6
usb_tc08_open_unit
    TC08Api.cpp, 4
usb_tc08_open_unit_async
    TC08Api.cpp, 4
usb_tc08_open_unit_progress
    TC08Api.cpp, 5
usb_tc08_run
    TC08Api.cpp, 8
usb_tc08_set_channel
    TC08Api.cpp, 7
usb_tc08_set_mains
    TC08Api.cpp, 5
usb_tc08_stop
    TC08Api.cpp, 5