Terasense software. C API reference

Terasense Group, Inc
21143 Hawthorne Blvd., #459, Torrance, CA 90503, USA

July 18, 2016

Contents

1 Directory content **2**

2 Introduction **2**

3 Reference **3**
3.1 Types . . . 3
3.2 Variables . . . 3
3.3 Constants . . . 3
3.4 Return codes . . . 4
3.5 Functions . . . 4

4 Samples **6**

1 Directory content

dlls\	
libteraview.dll	DLL files for the TeraView library and driver
okFrontPanel.dll	support software. Any software will depend on
driver\	these DLLs.
FrontPanelUSB-DriverOnly-4.4.0.exe	Device driver. Needs to be installed on the PC which would work with the device.
inc\	
libteraview.h	Header file for the TeraView library.
lib\	
libteraview.lib	Import library file for the TeraView library.
sample\	
sample.c	Source file for the simplest sample program.
sample.vcxproj	Visual Studio project for the simplest sample program.
simple App\	
simple App.c	Source file for a simple application.
simple App.vcxproj	Visual Studio project for a simple application.
Terasense Samples.sln	Visual Studio solution for the sample projects.

2 Introduction

The TeraView library provides C interface for developing programs for Terasense imaging cameras models Tera-1024, Tera-256, and their custom modifications. It is compiled for Win32 platform and should work for Windows 7 and above. In order to use library you need to include header file found in inc\ folder, link to import library file found in lib\ folder and put *both* DLL files found in dll\ folder within the search path (for example, in the same folder as your executable). You also need to install device driver found in driver\ folder (if you use Terasense Software on this computer, the driver is already installed).

The operation is started by initialization of the device (tsInit()). During this process a connection with the device is established, configuration data is read, and parameter variables are set. Next data acquisition is started (tsStart()). This step creates internal thread that constantly reads data from the device, process them, and stores result in internal buffer. Read operation tsRead() reads data from the internal buffer into preallocated buffer provided by you (use (TS_DATA) malloc(dataLength*sizeof(TS_ELEMENT)). It is guaranteed than the same data are never read twice and if there is no new data yet, the function blocks until they are available.

Data processing includes three steps: substracting backgound data, normalizing data, and "stitching" – interpolating over non-performing pixels. At the initialization device uses factory calibration (background and normalization), but you can record a new one using tsRecordBackground() and tsRecordNormalization() functions or load previously saved one using tsLoadConfig(). Background data are stored for each exposure separately and selected automatically. The normalization always contains two sets of data — default and recorded. Default never changes tsRecordNormalization() replaces recorded set; if you never used this function, the recorded data coincide with the default.

The "stitching" of isolated non-performing pixels is performed automatically and if there was extended regions of non-performing pixels during normalization recording (i.e., not all the surface have been illuminated), you may switch on interpolations over those regions by turning on "greedy" mode (tsSetGreedy()). It is possible

to influence which pixels are considered non-performing by setting threshold parameter with tsSetThreshold(). The meaning of the parameter roughly corresponds to signal-to-noise ratio during calibration. Pixels which have this value below the set threshold are marked as non-performing.

You can change exposure setting using tsSetExposure(). Change in exposure changes signal amplification and readout rate. Generally, higher number of the exposure corresponds to higher amplification and lower readout. Available range of exposures can be found using tsGetExposureRange(). (Important notice: if tsSetExposure() is called when acquisition is running, there is no guarantee that first frame to be read afterwards would be obtained with new exposure settings. You may want to dismiss this frame.)

To free resources, call tsStop() when you do no need data, and tsClose() to free the device.

3 Reference

3.1 Types

typedef signed __int16 TS_ELEMENT
Type for a single data point.

typedef TS_ELEMENT * TS_DATA
Pointer to a buffer containing data points of a frame.

typedef int TS_RES
Type for return codes for functions in the library.

typedef int (*ts_ticker_t)(double completion)
Type for a ticker callback function being used by tsRecordBackground() and tsRecordNormalization(). Completion parameter varies from 0 (just started) to 1 (finished). The function should return FALSE by default and TRUE to request abort of the operation.

3.2 Variables

Variables are undefined before the initialization (see tsInit()). All variables are read-only, attempts to assign a new value will lead to crash!

int sizeX, sizeY Dimensions of the sensor.

int dataLength Total number of data points ($dataLength = sizeX * sizeY$).

char deviceIDstring Identification string for the device.

3.3 Constants

TS_MAX_VALUE 1023
Maximum value for a data point.

TS_RAW_LIMIT 700
The top limit for raw data. This limit is "soft", it varies for different pixels – and actual value is within ±1% of TS_RAW_LIMIT.

DEFAULT_EXPOSURE 3
Initial setting for the exposure. See tsSetExposure().

DEFAULT_THRESHOLD 20
Initial setteing for the threshold. See tsSetThreshold().

3.4 Return codes

Name	Value	Description
TS_OK	0	Success.
TS_FAILED	-9999	General failure.
TS_CANCELLED	-9998	Cancelled by user.
TS_ERR_WRONG_PARAMETERS	-9997	Invalid parameters (out of range, too long or too short, etc.).
TS_NOTRUNNING	-9996	Data acquisition have not been started. (Use tsStart()).
TS_ERR_SIZE_MISMATCH	-9995	Imported data does not fit current configuration.
TS_OUTPUT_ERROR	-9994	File write operation failed.
TS_INPUT_ERROR	-9993	File read operation failed.
TS_NOTCONFIGURED	-9992	Initialization have not been performed or device have been closed. (Use tsInit()).
TF_PATHNOTFOUND	-9991	File opening error - file or path cannot be found or permissions prevent it fom being opened or file is in use.

3.5 Functions

TS_RES tsInit(void);
Initialize the device. This function should be called prior to any operation with the device, which have to be connected to the computer. It performs initialization and sets variables *sizeX, sizeY, dataLength*, and *deviceIDstring*.

TS_RES tsClose(void);
Close the device. Device can be initialized again by calling tsInit().

TS_RES tsStart(void);
Starts data acquisition and processing. The acquisition and processing are performed continuously in separate threads and results are stored in internal cyclic buffer.

TS_RES tsStop(void);
Stops data acquisition and processing, joining corresponding threads. It can take as long as one frame period at current exposure to return.

int tsIsRunning(void);
Checks whether the acquisition is running. Returns 0 (false) or -1 (true).

TS_RES tsRead(TS_DATA buffer);

Reads processed data into the buffer (the buffer of the size $dataLength * sizeof(TS_ELEMENT)$ must be allocated beforehand!). Returned data are between 0 and **TS_MAX_VALUE**.
If there are no new data, the function will block until they become available. If acquisition have not been started, the function will return **TS_NOTRUNNING**.

TS_RES tsReadRaw(TS_DATA buffer);

//Reads raw (unprocessed) data into the buffer (the buffer of the size $dataLength*sizeof(TS_ELEMENT)$ must be allocated beforehand!). No background compensation and calibration is applied. Returned data are approximately between **TS_RAW_LIMIT**−**TS_MAX_VALUE** and **TS_RAW_LIMIT**.
If there are no new data, the function will block until they become available. If acquisition have not been started, the function will return **TS_NOTRUNNING**.

TS_RES tsSetExposure(int exposure);

Sets current exposure. Available range can be obtained tsGetExposureRange(). If the aquisition is running when you use this function, the next frame read with tsRead() or tsReadRaw() may be obtained either with old, or with new exposure setting. You may wish to dismiss this frame.

int tsGetExposure(void);

Returns current exposure setting.

TS_RES tsGetExposureRange(int *pMin, int *pMax);

Gets admissible range of exposures.

double tsGetIntTime(int exposure);

Gets length of the integration time for the given exposure in milliseconds; if parameter is out of range (i.e., negative), the value for the current exposure is returned. The integration time is proportional to the amplification factor and is about 1/32 of the frame duration.

TS_RES tsSetThreshold(double threshold);

Sets threshold value. The threshold is used to define non-performing pixels for current normalization; it rougly corresponds to the signal-to-noise ratio during the calibration. Higher threshold means more non-performing pixels, lower means more noise in resulting data (because more noisy pixels are used).

TS_RES tsSetDifference(int on);

Turns on/off difference mode. Non-zero value of parameter will turn it on, zero will turn it off.

TS_RES tsSetGreedy(int on);

Turns on/off greedy stitch mode. Non-zero value of parameter will turn it on, zero will turn it off.

TS_RES tsRecordBackground(ts_ticker_t callback);

Records background compensation data. The radiation source should be off. The data are recorded for all exposures and the procedure will take several minutes. The parameter is a pointer to callback function which is called periodically to indicate progress. Return value of this function can be used to cancel the process. Set the parameter to **NULL** to prevent callbacks.
The recorded data replace default during the session, to use them in subsequent sessions you need to save and then load config.

TS_RES tsRecordNormalization(ts_ticker_t callback);

Records normalization data. The radiation source should be on and it is advisable to record background data beforehand. The parameter is a pointer to callback function which is called periodically to indicate progress. Return value of this function can be used to cancel the process. Set the parameter to **NULL** to prevent callbacks.

The recorded data replace default during the session, to use them in subsequent sessions you need to save and then load config.

TS_RES tsSelectNorm(int i);

Selects nomaliztion (0 - default, 1 - recorded).

TS_RES tsSaveConfig(FILE *stream);

Saves calibration data (i.e. background and normalization) to file. The parameter should be a binary stream opened for writing.

TS_RES tsSaveConfigAs(const char *filename);
TS_RES tsSaveConfigAs_w(const _wchar_t *filename);

Helper functions that save configuration to the file indicated by filename. If the file already exists, it is overwritten silently; if it cannot be open for any reason, functions return **TS_PATHNOTFOUND**. The first function takes ANSI string as argument, while the second takes wide-character string.

TS_RES tsLoadConfig(FILE *stream);

Reads calibration data (i.e. background and normalization) from file. The parameter should be a binary stream opened for writing. The data read replace default during the session.

TS_RES tsLoadConfigFrom(const char *filename);
TS_RES tsLoadConfigFrom_w(const _wchar_t *filename);

Helper functions that load configuration from the file indicated by filename. If the file cannot be open for any reason, functions return **TS_PATHNOTFOUND**. The first function takes ANSI string as argument, while the second takes wide-character string.

4 Samples

For the expample of a simplest program see included 'sample' project; the 'simple App' project contains a little more elaborate example, which demonstrates repeated read out, changing exposure, and recording background and normalization.