Multichannel data-acquisition systems

LTR210

Programmer manual

Revision 1.0.5
September 2014

L-CARD

http://en.lcard.ru
en@lcard.ru

DAQ SYSTEMS DESIGN, MANUFACTURING & DISTRIBUTION

Author of the manual:
Alexey Borisov

L-Card LLC
117105, Moscow, Varshavskoye shosse, 5, block 4, bld. 2

tel.: +7 (495) 785-95-19
fax: +7 (495) 785-95-14

Internet contacts:
http://en.lcard.ru/

E-Mail:
Sales department: en@lcard.ru
Customer care: en@lcard.ru

Table 1: Current document revisions

Revision	Date	Description
1.0.0	23.04.2013	Full revision of this document (preliminary data)
1.0.1	14.06.2013	Description of new version of synchronization threshold, additional bit mode set-up, outputting-to-interface rate are added. All field names are finalized to the library version
1.0.2	25.06.2013	AFC adjustment procedure description is added.
1.0.3	19.08.2013	Description of measurement and null offset adjustment capability is added.
1.0.4	02.06.2014	Description of library connection for projects with 64-bit compilers MSVC, Embarcadero C++ Builder and MinGW is added.
1.0.5	10.09.2014	The problem to connect the library to the projects is discussed in the individual document.

Contents

Chapter 1

What this document is about

This document assumes that the user is familiar with the documents “Starting operating with the LTR Crate System. Software issues.“ and “Software for the LTR system”, where the main operating principles of the software for LTR crates are described.

This document is mainly intended for the programmers who are going to code for operation with LTR210 module using library ltr210api provided by the "L-Card" Company.

The problem of library connection to the user's project is under consideration in this document as well as interface functions provided by the library and types used and basic approaches to use these functions are described in detail herein.

The library itself is written in *C* language and all function and type declarations are provided in *C* language. However, all bindings to all other software languages are only envelopes over other *C* libraries and all functions, types and parameters save their values for other software languages. Therefore this document is useful for users that code in other software languages.

The problems related to module characteristics and signals connection are out of scope of this document and the operating principles of the module itself are come up in general only. The mentioned above problems are described in the relevant chapter of the document "LTR Crate System. User manual”, the user must be familiar with prior to start reading this document.

Chapter 2

Library installation and connection to the project

Application of the libraries for manipulating with the LTR crate system is described in the document “Starting operating with the LTR crate system. Software issues.”.

Chapter 3

General approach to working with the library

3.1 General algorithm to working with the module

This section describes the typical sequence of actions during operation of LTR210 module. Each step is described in details in the following chapters. The typical sequence of actions is as follows:

1. Create the instance of the structure TLTR210 representing the module handle. The module handle contains the information on the module and is used when accessing all other functions.
2. Initialize the handle fields using LTR210_Init()
3. Make connection with targeted module with the function LTR210_Open().
4. Check whether the module FPGA firmware is downloaded via the function LTR210_FPGAIsLoaded(). If the firmware is not downloaded, then download it calling the function LTR210_LoadFPGA().
5. Fill the necessary fields with module configurations of module handle substructure Cfg and call LTR210_SetADC() to record configurations in the module.
6. If required, own null may be measured for additional adjustment of escape via LTR210_MeasAdcZeroOffset().
1. Starting data acquisition using LTR210_Start().
2. Data acquisition and processing depending on the mode as described below
3. Upon completion of operation make stopping data acquisition with the LTR210_Stop().
4. Close connection with the module by calling the function LTR210_Close().

3.1.1 Module operation in the frame-based acquisition mode

Typical data acquisition and processing cycle in the frame-based acquisition mode is as follows:

1. Waiting for data arrival from the module via LTR210_WaitEvent().
2. Determine what kind of data has arrived via the event type (parameter event):

- If no data arrived within the specified period then we shall return to paragraph 1 to wait for further data. Upon initiated periodic status sending the last status acceptance time may be checked to identify facts of the module failure.
- If the periodic module status (live signal) is accepted, then analyze status (if necessary) and proceed to paragraph 1.
- If the beginning of recorded frame is accepted, then proceed to paragraph 3.

3. Acceptance of readings of the frame in quantity of State.RecvFrameSize with the LTR210_Recv().
4. Processing of received frame using LTR210_ProcessData().
5. Analysis of accepted frame status to define validity of accepted frame. User-specific processing and proceeding to paragraph 1.

3.1.2 Module operation during continuous thread input

The typical data acquisition and processing cycle is similar to most of ADC modules and is as follows:

1. Receiving of set number of readings using LTR210_Recv().
2. Processing of received readings using LTR210_ProcessData().

3.2 Downloading of the module FPGA firmware

LTR210 module is not capable of FPGA firmware storage in internal non-volatile memory, thus it is recommended that the firmware is previously downloaded to the firmware for module operation.

As a rule, firmware downloading to the module is the primary action subject to execution after module connection via LTR210_Open(). However, as FPGA firmware after downloading is saved in the module until power reset, it should be redownloaded, if it was downloaded during the previous module operation. The function LTR210_FPGAIsLoaded() is used for check whether the firmware is downloaded. If this function has returned LTR_OK then the firmware is already downloaded and you may proceed to the module setting. Otherwise the firmware shall be downloaded via the function LTR210_LoadFPGA().

Function LTR210_LoadFPGA() takes the firmware file name as one of parameters. File **ltr210_fpga.rbf** containing the firmware is supplied complete with the library ltr210api. Besides upon the library creation the last version of the firmware in OC Windows is integrated into the library as a resource which allows to omit the separate firmware file storage. Downloading of the firmware integrated into the library is executed upon transfer of the blank string or null pointer as the file name. As for OC Linux the firmware file is downloaded to the **/usr/share/ltrapi/ltr210** directory during installation from supplied packages, and upon indication of the blank file name the file located in the same route is used.

It should be noted that in LTR210_LoadFPGA() a pointer to the callback function (callback) that shall be called each time the data block from the firmware file was successfully written to FPGA may be transferred. It allows for downloading indicator realization in the program, if required. Full size firmware file in bites, quantity of successfully written bites and a pointer to

user-specific data which may be transferred to LTR210_LoadFPGA(), are regularly transferred to the callback function. Initially the callback function shall be called immediately after successful opening of file (or resource downloading) of written zero-size bites, and finally - after successful downloading of files of size in written bites equal to the firmware size. In case of failure LTR210_LoadFPGA() immediately returns failure code as the result without calling of the callback function. If the downloading indicator is not required then the null pointer may be transferred as the function pointer.

3.3 Module setting

If the FPGA firmware was downloaded then module setting should be set at the next stage.

Module setting is performed in the same manner for the most of other LTR modules: initially values of all module parameters are written in corresponding field of the module handle structure, then the function LTR210_SetADC() which transfers values of these fields to the module is called.

It should be noted that all fields relating to module setting are integrated in structure of TLTR210_CONFIG type (module handle Cfg). These fields only can be changed by the user manually in module handle during normal operation and only these fields affect parameters recorded by LTR210_SetADC(). The following parameters are defined during the module setting:

- ADC acquisition frequency (section Set-up of ADC acquisition frequency)
- ADC channels configurations (section ADC channels setting)
- Frame size and pre-history for the frame-based acquisition mode (section Setting of the received frame parameters)
- The synchronization event whereby the frame is acquired (section Synchronization event setting)
- Thread operating LTR210 set-up in a group (section Operation of few modules LTR210 in a group)

Upon completion of LTR210_SetADC() some parameters being derivative of configurations from TLTR210_CONFIG are calculated. These parameters are filled in corresponding structure fields of module state TLTR210_STATE type (module handle State field). For example, resultant ADC acquisition frequency (Hz) is calculated based on ADC frequency dividers and decimation factors, and filled in the AdcFreq field.

The module shall be set at least once per year prior to initiation of data recording.

Besides, a part of the module parameters may be modified upon acquisition run:

- ADC channel measurement ranges
- ADC channel measuring mode
- Analogue synchronization level
- Frame frequence at periodic acquisition

To modify these parameters "just-in-time" you should equally modify required fields of the structure configurations and call LTR210_SetADC().

3.3.1 Set-up of ADC acquisition frequency

Both channels of the module ADC always operate in parallel at the same conversion frequency that may reach up to 10 MHz (for each channel). ADC acquisition frequency is set by two parameters:

- ADC conversion frequency divider. Frequency 10 MHz is divisible by a divider equal to the value of the field AdcFreqDiv+1. Value AdcFreqDiv may be from 0 to LTR210_ADC_FREQ_DIV_MAX-1.
- Data decimation factor. Data shall be decimated from ADC to ensure lower acquisition frequency in FPGA. This factor is set by the field AdcDcmCnt. Only one reading shall be recorded from readings AdcDcmCnt+1 to the module buffer. Value AdcDcmCnt may be modified in a range from 0 to LTR210_ADC_DCM_CNT_MAX-1.

Hence, resulting acquisition frequency for each channel ADC shall be equal to

$$f_{acq} = \frac{10000000}{(AdcFreqDiv + 1) * (AdcDcmCnt + 1)} \text{ Hz}$$

Installation of essential values of fields AdcFreqDiv and AdcDcmCnt may require application of the function LTR210_FillAdcFreq() which select these values such that the ADC acquisition frequency is the closest to the specified one indicated in corresponding function parameter.

After invocation of LTR210_SetADC() the value corresponding to the actual ADC acquisition frequency is filled in the field AdcFreq of the module structure.

Set acquisition frequency does not depend on the number of enabled ADC channels.

3.3.2 ADC channels setting

Each LTR210 module has two channels handling the conversion in parallel. Recording may be enabled either by any channel, or by both channels simultaneously. Each channel configurations are integrated in structure TLTR210_CHANNEL_CONFIG. The array of structures consisting of 2 elements which each element corresponds to the required channel is the field of Ch structure with module configurations.

The following can be configurated for each channel independently:

- Whether the record is enabled for this channel. Defined by the field Enabled.
- Range for this field (field Range).
- Channel operating mode (field Mode): open input (continuous steady component), closed input (interrupted steady component), or measurement of eigen null.
- Upper and lower synchronization levels in corresponding synchronization mode (section Synchronization event setting)
- Operating mode of special bit integrated into the reading thread corresponding to the data channel (section Setting of additional bit mode in the data thread from ADC).

3.3.3 Setting of the received frame parameters

In a frame-by-frame mode of data acquisition which is the main mode for the LTR210 module, reading output is executed in blocks herein referred to as frames. Frame output is executed only in case of synchronization event.

Number of points per channel in the frame is set by the field FrameSize. Herewith a part of the frame readings corresponds to measurements immediately prior to the synchronization event, and another part - to ADC readings immediately after this event. Frame readings corresponding to the pre-event period resulting in output of this frame are called event pre-history. The pre-history size at a rate of points per channel is set via parameter HistSize (may be equal to 0).

Frame size constraints depending on record mode are described in section Frame record and output concept.

The Frame state, which contains information about the frame validity and other different events occurred during recording and frame output, is always sent after the last frame word. The frame state should always be verified to ensure conclusion of received data validity.

Therefore, the frame consists of $N_{ch} * FrameSize + 1$ words, where N_{ch} is a quantity of enabled channels (1 or 2). After recording of the module configuration via LTR210_SetADC() calculated frame size is saved in the field RecvFrameSize of the module state structure. Value of this field may be used further upon data acquisition from the module.

3.3.4 Synchronization event setting

In a frame-by-frame mode upon invocation of the function LTR210_Start() the module is switched to the specified condition standby mode, and only in this condition the data frame would be transferred to the module interface. This condition called the synchronization event is set via the field SyncMode. The condition itself should be set prior to invocation of LTR210_Start() at the stage of the module configuration.

The following synchronization event variations are possible:

- On command from PC. This command is sent via the function LTR210_FrameStart(). Upon command a data frame output shall be executed.
- Via rising or falling edge of analogue signal in one of two channels in respect of preset synchronization level. Synchronization level is set separately for each channel via two levels — upper (field SyncLevelH) and lower hysteresis level (field SyncLevelL). Two levels enable false positive avoidance even at noisy signal. Rising edge is a signal transition through the upper level assuming that it was lower than the lowest level. Falling edge is a transition lower than the lowest level in case the signal was higher that the upper level. Preset levels should be within the specified range for this channel. Undoubtedly, the upper level should be not lower than the lower one. Whereby the channel record used for analogue synchronization may be prohibited, if only other channel data is of interest, but acquisition range and mode should be set correctly.
- On rising/falling edge of digital signal at the SYNC input. It should be noted that the mode is intended for synchronization from external source of digital signal that differs from another module LTR210. For operation of a number of modules LTR210 in a group on the

principle "master-slaves" there is a special configuration for the module operation in a group.

- Periodic frame acquisition. In this mode the synchronization event is generated by the module hardware at specified frequency. This frequency is determined by the field FrameFreqDiv and is equal to $f_{frame} = \frac{10^6}{FrameFreqDiv+1}$ Hz. After invocation of LTR210_SetADC() calculated frequency in Herz is filled in the field FrameFreq of the module state structure. Function LTR210_FillFrameFreq() may be used to fill the field FrameFreqDiv.

It should be noted that in case of synchronization event following after occurrence of the synchronization event prior to record completion and further frame output to the interface, then this event shall be discarded. The previous frame shall be successfully transferred, but the missing event notification flag shall be set as its status flag LTR210_STATUS_FLAG_SYNC_SKIP.

It must be separately noted that SyncMode=LTR210_SYNC_MODE_CONTINUOUS is a specific case. Upon this value the module is set for continuous acquisition mode which is described in details in section Continuous data acquisition mode.

3.3.5 Operation of few modules LTR210 in a group

As for the module LTR210 it is possible to integrate some modules in a thread on the "master-slaves" principle. The field GroupMode is intended for the mode set-up. By default this field is equal to LTR210_GROUP_MODE_INDIVIDUAL which corresponds to the module operation apart from the rest.

In case of module operation in a group one module should be set as the master by setting GroupMode = LTR210_GROUP_MODE_MASTER. The master module traces synchronization events in accordance with specified value of the field SyncMode upon initiation of record. Detecting synchronization event the master module not only transfers the frame to the interface with crate but also generates a pulse in the synchronization line. Any value of SyncMode, apart from LTR210_SYNC_MODE_CONTINUOUS may be set for the master module.

Slave modules must be connected with the master module one by one via SYNC inputs (for detailed information refer to the document "LTR Crate System. User manual"). That means, if there are master module *M* and two slave modules *S1* and *S2*, then M may be connected with *S1*, and *S1* with *S2*. he field GroupMode = LTR210_GROUP_MODE_SLAVE shall be installed for all slave modules. Whereby the field SyncMode has no value — the slave module always runs frame acquisition by a signal from the master module.

Modules may be set randomly. However initiation of data record via LTR210_Start() should be initially executed for all slave modules, and then - for master module to ensure that the first synchronization event is not lost by any slave module.

3.3.6 Data output to interface rate setting

In case of synchronization event recorded data is read out of the circular buffer and transferred to the module → crate interface at specified rate. Maximum rate is determined by the crate type wherein the module is inserted. For LTR-U-1 it is equal to 200 Kwords/s, for the rest crates - 500 Kwords/s. This rate determines recorded frame output time and effects minimum interval of synchronization event sequence which shall not be missed, and maximum

size of recorded frame in case automatic record delay is not on (for more detailed information refer to section Frame record and output concept).

The rate may be explicitly set via the field IntfTransfRate. Rate 500 KWords/s is set by default. Whereby, if set value exceeds maximum permitted rate for crate (if the rate is 500 KWords/s and LTR-U-1), then the library automatically records correct value. I.e. maximum permitted rate is used for the applied crate.

Explicit indication of rate may be used in case of a large number of modules LTR210, for which the frame output time may intersect over time, to prevent excess maximum capacity crate → computer.

Example: Suppose LTR-EU-16 with 16 modules LTR210, connected to the Ethernet interface, is used. If all modules simultaneously transfer recorded frame at the rate 500 KWords/s, then total rate shall be 8 MWords/s (24 MBite/c) which exceeds maximum crate transfer rate via Ethernet (about 10 MBite/s or 2.5 MWords/s). I.e. crate can not transfer data to PC at the same rate as it acquires it; at specific size of frames the internal crate buffer may be overloaded. In such case output rate to interface for modules may be set as 100 KWords/s, which shall reduce overall flow rate up to 1.6 MWords/s.

Hence in case of large amount of modules one should always estimate total transfer rate of data from all crate modules (as well as some crates if they are connected to one network) taking into consideration other types of module (not LTR210) and, if required, explicitly reduce data output rate to interface to omit overloading of internal crate buffer.

3.3.7 Setting of additional bit mode in the data thread from ADC

Additional data bit is transferred with ADC reading in the module LTR210. You may set what exactly this bit would display specifically for each ADC channel via the field DigBitMode. For example, this bit may display level at the SYNC inlet that allows for using this inlet for synchronous input of one digital bit, if required. Other possibilities are described in e_LTR210_DIG_BIT_MODE specifying the mode for this bit.

In the processing of data via LTR210_ProcessData() value of this bit is saved in the structure array with additional information (if non-zero pointer was transferred as a parameter data_info). Each structure corresponds to its report generated from processed data which allows for execution of one-to-one correspondence.

3.4 Frame record and output concept

Compared to the majority of other "L-CARD" ADC the main operating mode of the module LTR210 does not assume continuous data acquisition. Instead of this, upon invocation of LTR210_Start() the module triggers data record to the internal circular buffer in SRAM of the module with capacity of 16 MWords (defined as constant LTR210_INTERNAL_BUFFER_SIZE) and waits for occurrence of specified synchronization event.

Only in case of synchronization event the module outputs data frame to the module → crate interface (hereinafter referred to as interface) in accordance with frame parameters described in section Setting of the received frame parameters.

Immediately after the frame transfer via the interface the module is ready for the next frame output upon the synchronization event. All synchronization events which occurred from the moment of the frame output to the interface (moment of occurrence of corresponding sync

event) to the moment of output completion shall be lost, whereby the frame status shall be indicated with the flag LTR210_STATUS_FLAG_SYNC_SKIP.

It should be noted that the rate of interface between the module and the crate is limited (200 KWords/s for LTR-U-1 and 500 KWords/s for other crates). Besides it may be additionally limited with the help of special configuration, if required, (section Data output to interface rate setting). Hence, frame-to-interface output rate may significantly exceed time of frame recording to the internal module buffer.

The module, by default, executes continuous recording to internal buffer. In this mode in case of specific frame size and ADC acquisition frequency, it may be that the process of reading recording to the circular buffer is in advance of the process of reading out of the buffer to the circle, and a part of transferred frame shall be filled with new data. Accordingly, in this case a part of the transferred frame data shall be invalid. This fact is displayed in the frame status by flag LTR210_STATUS_FLAG_OVERLAP.

Rate of data recording to the buffer is equal to $N_{ch}*f_{acq}$, where f_{acq} is the acquisition frequency (section Set-up of ADC acquisition frequency) and N_{ch} is the number of channels enabled for record. If recording rate is less than reading rate, then the size FrameSize may be equal to any value up to $\frac{LTR210\_FRAME\_SIZE\_MAX}{N_{ch}}$. Constant LTR210_FRAME_SIZE_MAX specifies maximum frame size which may be set in one enabled channel, corresponding to size of the internal buffer reduced by 512 words, or the size of blocked record in the module SDRAM.

If recording rate exceeds the reading rate, then time of frame transfer via interface shall be less than recording time of the entire circular buffer plus the frame size after deduction of pre-history, to prevent overwriting. I.e. inequality shall be performed

$$\frac{S + N_{ch} * (FrameSize - HistSize)}{f_{acq} * N_{ch}} \geq \frac{N_{ch} * FrameSize}{f_{intf}}$$

where S = LTR210_FRAME_SIZE_MAX, f_{intf} is the rate of data-to-interface output.

If the pre-history size is set in formula as a portion of the entire frame n size: $HistSize = n * FrameSize$, where n from 0 to 1, then the formula may be rewritten into more convenient form. Where appropriate to identify maximum frame size at specified ADC acquisition frequency, rate of data-to-interface output and pre-history percentage, then the formula shall be as follows:

$$FrameSize \leq \frac{S * f_{intf}}{N_{ch} * (f_{acq} * N_{ch} - f_{intf} * (1 - n))}$$

Table of results of this formula for some typical cases is set out in the document "Crate system LTR. User Manual".

In case set frame sizes require identification of maximum acquisition frequency, then the formula shall be as follows:

$$f_{acq} \leq \frac{f_{intf} * (S/N_{ch} + FrameSize * (1 - n))}{N_{ch} * FrameSize}$$

To enable setting of the full-size buffer at acquisition frequencies up to maximum, the automatic delay of recording-to-circular-buffer mode is set. This mode is set at the module configuration via the flag LTR210_CFG_FLAGS_WRITE_AUTO_SUSP in the field Flags of configuration structure. In this mode upon synchronization event data is recorded to the internal buffer till the end of the frame, afterwards recording process shall be automatically stopped. Whereby record shall be proceeded only after completion of output of the entire frame to interface, and rewriting is impossible.

On the plus side, in the automatic record delay mode a frame of size up to $LTR210\_FRAME_{Nch}\_SIZE\_MAX$ may be recorded despite of ADC reading frequency. On the negative side, if the-prehistory is of null size, then time required for pre-history recording shall pass after completion of the frame-to-interface output prior to occurrence of the next synchronization event. In continuous record mode in the event occurred immediately after completion of transfer of previous frame to the interface, the frame shall be always successfully recorded notwithstanding of the pre-history size.

Hence to omit no event in case of continuous record, an interval between the closest synchronization events shall be:

$$T_{sync\_cont} > \max(\frac{N_{ch} * FrameSize}{f_{intf}}, \frac{FrameSize - HistSize}{f_{acq}})$$

and in automatic record delay mode:

$$T_{sync\_susp} > T_{sync\_cont} + \frac{HistSize}{f_{acq}}$$

3.5 Data acquisition in the frame-based acquisition mode

As previously described, there is no continuous data thread from module in standard operating conditions, and the data is transferred asynchronously frame by frame. Besides apart from data frames, the module can periodically transfer its status word (for more detailed information refer to section Module operation control via periodic status (live signal)). As such data acquisition procedure slightly differs from other LTR modules. At first, it is advised to wait for the event of any data receipt from the module. The function LTR210_WaitEvent() waiting for data receipt from the module during timeout, analyzing this word and returning information of the type of received data in parameter *event,* enables this.

If this is the beginning of the frame (LTR210_RECV_EVENT_SOF), then it is recommended to read the data frame similar to the procedure regarding the LTR modules: initially read RecvFrameSize of words received from the module via LTR210_Recv(), then process them using LTR210_ProcessData(). If necessary, the frame may be acquired and/or processed not entirely, but in blocks of less size. However it should be noted that only after processing of the last block the frame status enabling conclusion of validity of received data.

It is essential to note that the first word of the frame upon transmission from module to crate/PC is specifically denoted and LTR210_WaitEvent() returns LTR210_RECV_EVENT_SOF just after the beginning of the frame is detected. If LTR210_WaitEvent() is called in the middle of half received frame, then words of incomplete frame shall be discarded.

Also LTR210_Recv(), as opposed to similar functions of the rest modules, upon detection of the frame end, shall immediately return data till the end of the frame, even if more data was required and timeout has not run out yet.

3.6 Analysis of the accepted frame status

As it was previously noted, apart from words with ADC readings the frame status which enables conclusion on validity of the frame data is transferred in the frame. Information of the frame status is returned by function LTR210_ProcessData() in the form of structure of type TLTR210_FRAME_STATUS.

Value of the field Result enables making general conclusion on whether the end of the frame is found and, if so, on data validity in received frame. If the word with the frame status was not found, then the field Result is equal to LTR210_FRAME_RESULT_PENDING (for example, if the user processes the frame partially). If the word of the frame status is found, then Result = LTR210_FRAME_RESULT_OK, if any data is valid, or LTR210_FRAME_RESULT_ERROR, if a part of data is damaged.

The field Flags contains full information of the frame status in the form of bit-by-bit combination of the flag set and may serve for more detailed analysis of the frame status. Particularly, these flags enable identification of reasons that caused data damage, if the value Result = LTR210_FRAME_RESULT_ERROR was returned.

The following flags refer directly to the received frame:

- LTR210_STATUS_FLAG_OVERLAP indicates that recording process is in advance of the reading process and a part of data in the frame was rewritten. It cannot be claimed which data in the frame are valid and which is not. The field Result is set in LTR210_FRAME_RESULT_ERROR. For more details of the event cause refer to section Setting of the received frame parameters.

- LTR210_STATUS_FLAG_INVALID_HIST indicates that the synchronization event has occurred earlier than HistSize of points for each channel was recorded, and the pre-history in the received frame is not valid. The field Result is set in LTR210_FRAME_RESULT_ERROR; however in the absence of other error flags it may be assumed that a part of the frame related to the period after the sync event is valid. In the continuous record to the buffer such situation may occur only in case of synchronizat6ion event immediately after the beginning of record via LTR210_Start() and HistSize/= 0 and only in the initial frame if record is enabled. If automatic record delay is triggered the flag may be set in case the following synchronization event has occurred earlier than HistSize of points for each channel was recorded upon and after completion of the previous frame output to the interface.

- LTR210_STATUS_FLAG_SYNC_SKIP indicates that during record and output of the frame to the interface one or few synchronization events which have been discarded occurred, as the previous frame output has not been finished yet. This flag does not affect data validity and, correspondingly, value of the field Result.

Besides flags contain information:

- Which channels enable record.

- PLL module capture state. Besides in absence of capture attribute PLL LTR210_ProcessData() returns error LTR210_ERR_PLL_NOT_LOCKED, as this fact indicates module failure.

All values of flags are listed in the description of the type e_LTR210_STATUS_FLAGS.

3.7 Module operation control via periodic status (live signal)

As in the frame-by-frame acquisition mode in absence of synchronization events the module records only to the internal circular buffer and does not transfer any data to the crate/PC, then

in such case there is no information upon which the module failure may be inferred. If additional check whether the module correctly executes record to internal buffer and synchronization conditions check is required, then periodic sending of the module word status may be enabled. This calls for setting of the flag LTR210_CFG_FLAGS_KEEPALIVE_EN in the field Flags of the configuration structure upon the module configuration prior to invocation of LTR210_SetADC(). In this case after record enable by invocation of LTR210_Start() the module shall transfer its status word once per 500 ms until record inhibit by invocation of LTR210_Stop(). Whereby status words are transferred only when the frame transmission is not running (i.e. they may be transferred only between frames, but not inside of the frame) thereby the word size in the frame is unchanged and acquisition procedure does not vary in case the periodic transfer of the module status is on/off.

Periodic status transfer may be used as the module live signal - absence of the module status words during the specified interval indicates the module failure. It should be kept in mind that this interval must exceed 500 ms (for example, few seconds) as it should consider any probable delay both of data transfer via interface between the crate and PC, and of possible software delay in data acquisition in the upper level server and program.

In the receipt of the module status, LTR210_WaitEvent() shall return the event LTR210_RECV_EVENT_KEEPALIVE. Besides, an additional information from received status shall be saved in the parameter *status* (if the null index was not transferred). It is similar to information in the frame status, except for the absence of flags that refer directly to the received frame. Thus the following information is available in the periodic status:

- Which channels enable record
- PLL module capture state. Besides in absence of capture attribute PLL LTR210_WaitEvent() returns error LTR210_ERR_PLL_NOT_LOCKED, as this fact indicates module failure.

Live signal (status) check during the specified interval may be executed by one of the following methods:

1. In the receipt cycle LTR210_WaitEvent() may be called with timeout equal to the interval during which it is deemed that the status must be acquired. If on expiration of timeout no event has occurred (LTR210_WaitEvent() returned the event LTR210_RECV_EVENT_TIMEOUT), then it indicates the module failure.
2. LTR210_WaitEvent() may be called with less timeout and, in case of no event, it should be checked how much time has passed since receipt of the last module data, by invocation of LTR210_GetLastWordInterval(). If this interval exceeds threshold, then the module may be considered defective. This method, particularly, allows interactive stop of data acquisition as it does not require expectation for longer timeouts in LTR210_WaitEvent().

3.8 Continuous data acquisition mode

For continuous data acquisition configuration SyncMode = LTR210_SYNC_MODE_CONTINUOUS shall be set. In this mode frame parameters (FrameSize and HistSize) have no value, and GroupMode should be equal to LTR210_GROUP_MODE_INDIVIDUAL (default value).

Besides in the continuous acquisition mode there is ADC acquistion frequency limit. This limit relates to the data transfer rate via interface between the module and the crate. As for LTR-U-1 inequation $N_{ch} * f_{acq} \leq 200$ KHz, and $N_{ch} * f_{acq} \leq 500$ KHz for the rest crates, shall be executed. These conditions should be checked in the invocation of LTR210_SetADC().

In the continuous data acquisition the live status transfer is not applied, i.e. the flag LTR210_CFG_FLAGS_KEEPALIVE_EN has no value.

In the continuous data acquisition by invocation of LTR210_Start() the data acquisition from ADC and transfer to the interface are initiated. Data is received similarly to other LTR modules from ADC via LTR210_Recv() and LTR210_ProcessData(). The module does not send any status words as the reading thread is not split into frames. Hence, as a result of frame processing, LTR210_ProcessData() always returns LTR210_FRAME_RESULT_PENDING and this data do not contain any additional information, so as a rule the null index may be transferred as *frame_status*.

3.9 The peculiarities of data calibration by the module

It should be noted that unlike most of the rest LTR modules the data calibration is performed by hardware inside the module but not by software. Thus, there are no instructions on calibration in LTR210_ProcessData(). The library itself performs reading of calibration coefficients from module Flash-memory and storing them in array CbrCoef fields in structure containing the information about the module and recording of these coefficients in FPGA.

FPGA performs calibration immediately by the formula $Y = (X + Offset) * Gain$, where X - ADC readings, Y — calibrated data, Offset — scale offset, and Gain — scale factor. In addition, before calibration, the ADC input value is expanded by 1 bit. Already calibrated 15-bit readings are transferred to the crate. In addition, the code LTR210_ADC_SCALE_CODE_MAX corresponds to voltage equal to maximum voltage for the given range.

3.10 AFC correction

Adjustment of the amplitude-frequency characteristic (AFC) of the module is enabled in the library.

For all ranges the AFC slope may be adjusted via the second stage feedback filter in accordance with the procedure described in the article
Delicate AFC response slope correction method with the use of ordinary digital filter. In Flash-memory of the module the measured amplitude ratio of the given frequency signal at maximum ADC acquisition frequency (10 MHz) to amplitude of the least frequency is saved for each channel as well as the signal frequency (Hz).

Besides, in ranges ± 10 V and ± 5 V the AFC knee in the low-frequency area may be adjusted via the infinite-impulse response filter in compliance with the procedure described in the article Ordinary infinite-impulse response filter for AFC knee adjustment in the low-frequency area of the bandwidth. Flash-memory contains R2 and C (R1 is always equal to 1) parameters of the equivalent filter circuit for AFC adjustment at maximum discretion frequency (10 MHz). The infinite-impulse response filter is applied prior to the feedback filter (whereby the feedback filter factor recorded in the Flash-memory considers AFC changes reflected by the infinite-impulse response filter).

All AFC adjustment parameters are read out from the module Flash-memory upon its opening along with calibration factors and saved in corresponding fields of the structure with information about the module.

When setting module via LTR210_SetADC() the library calculates filter factors for specified discretion frequency, if they were calculated in the previous start, using abovementioned parameters. In the processing of received data to the function LTR210_ProcessData() the flag LTR210_PROC_FLAG_AFC_COR, which indicates that calculated filter(s) shall be applied for the module AFC adjustment, may be set.

3.11 Null offset measuring and adjustment

LTR210 enables measurement of the own null. It allows, if necessary, for taking into account null offset which may be related either with its temporary escape, or with change in ambient conditions. This calls for own null measurement immediately prior to record initiation and upon further acquisition get adjusted measurements by deduction of measured null from received readings. Null offset depends on applied ranges and a channel, thus null should be measured for the same configurations, as the further measurements.

Though the user may program this algorithm by himself/herself, if necessary, the library provides functions to facilitate this operation. The function LTR210_MeasAdcZeroOffset() measures its own null for ranges specified in the module configuration. It should be called after ADC setting prior to enabling data recording. Calculated values are saved in the field AdcZeroOffset of the module status structure.

Hereafter, in the data processing, LTR210_ProcessData() may be used to transfer the flag LTR210_PROC_FLAG_ZERO_OFFS_COR, whereby LTR210_ProcessData() shall deduct saved measured values of null offset from processed readings.

It should be noted that upon and after null offset measurement via LTR210_MeasAdcZeroOffset() until data processing, channel ranges shall not change. In case of change in ranges, the null offset must be measured again.

3.12 Invocation of library functions from different treads

Functions of the library ltr210api, and libraries for operation with other LTR modules, are not thread-safe. That means that the user must ensure that function invocation for operation with the same module is sequential (certainly, parallel operation with different modules from different threads is possible).

However, to ease application, there are exceptions to the rules. Following data acquisition in a separate thread via LTR210_WaitEvent(), LTR210_Recv() and LTR210_ProcessData() the functions may be called from another thread:

- LTR210_FrameStart() for invocation of software synchronization event
- LTR210_SetADC() for immediate update of some configurations

It enables invocation of LTR210_FrameStart() and LTR210_SetADC() from the user interface thread even in case of acquisition in a separate thread.

Chapter 4

Constants, types of data and library functions

4.1 Constants and enumerations.

4.1.1 Constants and macros.

Constant	Value	Description
LTR210_NAME_SIZE	8	Size of a string with the module name in the structure TINFO_LTR210
LTR210_SERIAL_SIZE	16	Size of a string with the serial module number in the structure TINFO_LTR210
LTR210_CHANNEL_CNT	2	Number of ADC channels in one module
LTR210_RANGE_CNT	5	Number of ADC measurement ranges
LTR210_AFC_IIR_COR_RANGE_CNT	2	Number of ranges for which additional AFC adjustment shall be executed via IIR filter
LTR210_ADC_SCALE_CODE_MAX	13000	Code of received ADC reading corresponding to maximum voltage with given range
LTR210_ADC_FREQ_DIV_MAX	10	Maximum value of ADC frequency divider
LTR210_ADC_DCM_CNT_MAX	256	Maximum reduction factor for data received from ADC
LTR210_ADC_FREQ_HZ	10000000	Frequency (Hz) for which ADC reading frequency is set
LTR210_FRAME_FREQ_HZ	1000000	Frequency (Hz) for which frame sequence frequency is set in the mode LTR210_SYNC_MODE_PERIODIC

LTR210_INTERNAL_BUFFER_SIZE	(16777216)	Size of internal circular module buffer in ADC readings
LTR210_FRAME_SIZE_MAX	(16777216 - 512)	Maximum frame size which may be set in one-channel mode

4.1.2 Error codes specific to LTR210.

Type: e_LTR210_ERROR_CODES		
Description: Error codes defined and used in ltr210api only. Other error codes used by different modules are defined in ltrapi.h		
Constant	**Value**	**Description**
LTR210_ERR_INVALID_SYNC_MODE	-10500	Wrong code of the frame acquisition condition is preset
LTR210_ERR_INVALID_GROUP_MODE	-10501	Wrong code of the group module operating mode is preset
LTR210_ERR_INVALID_ADC_FREQ_DIV	-10502	Wrong value of ADC frequency divider is preset
LTR210_ERR_INVALID_CH_RANGE	-10503	Wrong code of ADC channel range is preset
LTR210_ERR_INVALID_CH_MODE	-10504	Wrong channel measuring mode is preset
LTR210_ERR_SYNC_LEVEL_EXCEED_ RANGE	-10505	Preset level of analogue synchronization is beyond the preset range
LTR210_ERR_NO_ENABLED_CHANNEL	-10506	No ADC channel is enabled
LTR210_ERR_PLL_NOT_LOCKED	-10507	PLL capture error
LTR210_ERR_INVALID_RECV_DATA_CNTR	-10508	Wrong counter value in received data
LTR210_ERR_RECV_UNEXPECTED_CMD	-10509	Acceptance of unsupported command in data thread
LTR210_ERR_FLASH_INFO_SIGN	-10510	Wrong attribute of the module information in the Flash-memory
LTR210_ERR_FLASH_INFO_SIZE	-10511	Wrong size of the module information read out of the Flash-memory
LTR210_ERR_FLASH_INFO_UNSUP_ FORMAT	-10512	Unsupported format of the module information from the Flash-memory

LTR210_ERR_FLASH_INFO_CRC	-10513	Failed to check CRC information about the module from the Flash-memory failed
LTR210_ERR_FLASH_INFO_VERIFY	-10514	Failed to check module information record from the Flash-memory
LTR210_ERR_CHANGE_PAR_ON_THE_FLY	-10515	Part of changed parameters can not be modified immediately
LTR210_ERR_INVALID_ADC_DCM_CNT	-10516	Wrong ADC data reduction factor is preset
LTR210_ERR_MODE_UNSUP_ADC_FREQ	-10517	Preset mode does not support preset frequency ADC
LTR210_ERR_INVALID_FRAME_SIZE	-10518	Wrong frame size is preset
LTR210_ERR_INVALID_HIST_SIZE	-10519	Wrong pre-history size is preset
LTR210_ERR_INVALID_INTF_TRANSF_ RATE	-10520	Wrong data-to-interface transfer rate is preset
LTR210_ERR_INVALID_DIG_BIT_MODE	-10521	Wrong operating mode of additional bit is preset
LTR210_ERR_SYNC_LEVEL_LOW_EXCEED_ HIGH	-10522	Lower threshold of analogue synchronization exceeds the upper one
LTR210_ERR_KEEPALIVE_TOUT_ EXCEEDED	-10523	No status from module has been received during the given time interval
LTR210_ERR_WAIT_FRAME_TIMEOUT	-10524	Failed to expect frame receipt during the given period
LTR210_ERR_FRAME_STATUS	-10525	Status word in received frame indicates the data error

4.1.3 ADC channel ranges

Type: e_LTR210_ADC_RANGE		
Description: ADC channel range		
Constant	**Value**	**Description**
LTR210_ADC_RANGE_10	0	Range ±10 V
LTR210_ADC_RANGE_5	1	Range ±5 V
LTR210_ADC_RANGE_2	2	Range ±2 V
LTR210_ADC_RANGE_1	3	Range ±1 V
LTR210_ADC_RANGE_0_5	4	Range ±0.5 V

4.1.4 ADC channel measuring mode

Type: e_LTR210_CH_MODE		
Description: ADC channel measuring mode		
Constant	**Value**	**Description**
LTR210_CH_MODE_ACDC	0	Measurement of variable and constant component (open input)
LTR210_CH_MODE_AC	1	Constant component split (closed input)
LTR210_CH_MODE_ZERO	2	Own zero measurement mode

4.1.5 Operating mode and synchronization events.

Type: e_LTR210_SYNC_MODE		
Description: Determined module operating mode and frame acquisition start condition (synchronization events)		
Constant	**Value**	**Description**
LTR210_SYNC_MODE_INTERNAL	0	Frame acquisition mode on the software command issued by invocation of LTR210_FrameStart()
LTR210_SYNC_MODE_CH1_RISE	1	Frame acquisition mode on the rising edge of signal relative to synchronization level in the initial analogue channel
LTR210_SYNC_MODE_CH1_FALL	2	Frame acquisition mode on the falling edge of signal relative to synchronization level in the initial analogue channel
LTR210_SYNC_MODE_CH2_RISE	3	Frame acquisition mode on the rising edge of signal relative to synchronization level in the secondary analogue channel
LTR210_SYNC_MODE_CH2_FALL	4	Frame acquisition mode on the falling edge of signal relative to synchronization level in the secondary analogue channel
LTR210_SYNC_MODE_SYNC_IN_RISE	5	Frame acquisition mode on the rising edge of signal at the SYNC inlet (not from another module!)
LTR210_SYNC_MODE_SYNC_IN_FALL	6	Frame acquisition mode on the falling edge of signal at the SYNC inlet (not from another module!)

LTR210_SYNC_MODE_PERIODIC	7	Periodic frame acquisition mode with preset frame sequence frequency
LTR210_SYNC_MODE_CONTINUOUS	8	Continuous data acquisition mode

4.1.6 Module operation mode in a group

Type: e_LTR210_GROUP_MODE		
Description: Determines the module operating mode within a group. Applied in the arrangement of LTR210 modules thread that record frames by one event.		
Constant	**Value**	**Description**
LTR210_GROUP_MODE_INDIVIDUAL	0	Module operates apart from other items
LTR210_GROUP_MODE_MASTER	1	Master mode - in case of preset synchronization event the module generates a signal to the SYNC inlet. This signal may be used by slave modules to initiate conversion simultaneously with the master module. May be used in association with any SyncMode value, except for LTR210_SYNC_MODE_CONTINUOUS
LTR210_GROUP_MODE_SLAVE	2	Slave module mode - the module initiates frame acquisition from a signal at the SYNC inlet which must generate another LTR210 set for a master mode. SyncMode value is not taken into account

4.1.7 Asynchronous event codes

Type: e_LTR210_RECV_EVENT		
Description: Codes that determine which asynchronous data was accepted from the module and which event it corresponds to. Returned by function LTR210_WaitEvent() in parameter event		
Constant	**Value**	**Description**
LTR210_RECV_EVENT_TIMEOUT	0	No event is received from the module within the given period of time
LTR210_RECV_EVENT_KEEPALIVE	1	Correct status signal is received from the module (live signal)
LTR210_RECV_EVENT_SOF	2	The beginning of recorded frame is received

4.1.8 Codes defining correctness of accepted frame

Type: e_LTR210_FRAME_RESULT		
Description:		
Constant	**Value**	**Description**
LTR210_FRAME_RESULT_OK	0	The frame is received without failure. Frame data is valid
LTR210_FRAME_RESULT_PENDING	1	No frame end attribute in processed data.
LTR210_FRAME_RESULT_ERROR	2	Accepted frame is defected. Frame data is invalid. The error reason may be recognized by status flags.

4.1.9 Status flags

Type: e_LTR210_STATUS_FLAGS		
Description: Flags indicating both current state of the LTR210 module, and information about the last accepted frame. Information from periodic words of the module status and the frame status words is presented as the combination of these flags.		
Constant	**Value**	**Description**
LTR210_STATUS_FLAG_PLL_LOCK	0x0001	PLL capture attribute upon status transfer. If zero, then the module is inoperable.
LTR210_STATUS_FLAG_PLL_LOCK_HOLD	0x0002	Attribute of the fact that PLL capture was maintained since the previous status transfer. Shall be set in any status except for the initial one
LTR210_STATUS_FLAG_OVERLAP	0x0004	Attribute of the fact that recording process is of advantage of the reading process. A part of the frame data may be invalid
LTR210_STATUS_FLAG_SYNC_SKIP	0x0008	Attribute of the fact that during the frame record at least one missed synchronization event occurred. Does not affect the frame validity.

LTR210_STATUS_FLAG_INVALID_HIST	0x0010	Attribute of the fact that pre-history of accepted frame is invalid (the event occurred less than in HistSize of readings after the record is enabled)
LTR210_STATUS_FLAG_CH1_EN	0x0040	Attribute of the fact that record is enabled in the initial channel
LTR210_STATUS_FLAG_CH2_EN	0x0080	Attribute of the fact that record is enabled in the secondary channel

4.1.10 Additional setting flags

Type: e_LTR210_CFG_FLAGS		
Description: Set of flags regulating module configuration during its set-up. Combination of these flags may be written in the field Flags of the structure TLTR210_CONFIG		
Constant	**Value**	**Description**
LTR210_CFG_FLAGS_KEEPALIVE_EN	0x001	Enable periodic transfer of the module status upon initiated acquisition
LTR210_CFG_FLAGS_WRITE_AUTO_SUSP	0x002	Enable automatic record delay mode while the frame is being transferred to the crate via interface. This mode allows for set-up of maximum frame size apart from ADC acquisition frequency
LTR210_CFG_FLAGS_TEST_CNTR_MODE	0x100	Test run mode where the counter is transferred instead of data

4.1.11 Data processing flags.

Type: e_LTR210_PROC_FLAGS		
Description: Flags regulating function operation LTR210_ProcessData()		
Constant	**Value**	**Description**
LTR210_PROC_FLAG_VOLT	0x0001	Flag to convert ADC codes in Volts. If this flag is not specified the ADC codes will be returned. In addition, the code LTR210_ADC_SCALE_CODE_MAX corresponds to maximum voltage for the specified range.
LTR210_PROC_FLAG_AFC_COR	0x0002	Indicates the necessity of AFC adjustment based on module factors recorded to the Flash-memory
LTR210_PROC_FLAG_ZERO_OFFS_COR	0x0004	Indicates the necessity of additional null adjustment using values From State.AdcZeroOffset, which may be measured by function LTR210_MeasAdcZeroOffset()
LTR210_PROC_FLAG_NONCONT_DATA	0x0100	Upon default LTR210_ProcessData() assumes that all received data is transmitted to be processed and checks the counter continuity not only within the accepted data block but among the calls. This flag shall be specified for the counter checking within the processed block only if not all data are processed or the same data are reprocessed.

4.1.12 Data output to interface rate

Type: e_LTR210_INTF_TRANSF_RATE		
Description: A set of constants which preset the rate of data reading from the LTR210 module buffer and data output to the crate interface.		
Constant	**Value**	**Description**
LTR210_INTF_TRANSF_RATE_500K	0	500 KWords/s
LTR210_INTF_TRANSF_RATE_200K	1	200 KWords/s
LTR210_INTF_TRANSF_RATE_100K	2	100 KWords/s
LTR210_INTF_TRANSF_RATE_50K	3	50 KWords/s
LTR210_INTF_TRANSF_RATE_25K	4	25 KWords/s
LTR210_INTF_TRANSF_RATE_10K	5	10 KWords/s

4.1.13 Additional bit operation mode in the input thread.

Type: e_LTR210_DIG_BIT_MODE		
Description: Determines which value shall be transferred as additional bit in readings accepted from the module.		
Constant	Value	Description
LTR210_DIG_BIT_MODE_ZERO	0	Bit value is always null
LTR210_DIG_BIT_MODE_SYNC_IN	1	Bit reflects state of the digital input of the SYNC module
LTR210_DIG_BIT_MODE_CH1_LVL	2	Bit is equal to "1" if signal level for the 1st channel of ADC exceeded upper level of synchronization and has not dropped beyond the lower level

LTR210_DIG_BIT_MODE_CH2_LVL	3	Bit is equal to "1" if signal level for the 2nd channel of ADC exceeded upper level of synchronization and has not dropped beyond the lower level
LTR210_DIG_BIT_MODE_INTERNAL_SYNC	4	Bit is equal to "1" for one reading upon response of software or periodic synchronization

4.2 Data types.

4.2.1 Calibration coefficients

Type: TLTR210_CBR_COEF		
Description: The structure storing the calibration coefficients for one channel and range.		
Field	**Type**	**Field description**
Offset	float	15-bit offset code
Scale	float	Scale coefficient

4.2.2 Parameters of infinite-impulse response filter.

Type: TLTR210_AFC_IIR_COEF		
Description: Parameters for calculation of infinite-impulse response filter factors used for adjustment of AFC ranges 10V and 5V.		
Field	**Type**	**Field description**
R	double	Equivalent filter circuit resistance
C	double	Equivalent filter circuit capacity

4.2.3 Module information.

Type: TINFO_LTR210		
Description: The structure containing the information about module circuit firmware versions and information from module Flash-memory (serial number, calibration coefficients). Apart from VerFPGA all fields are valid after invocation of LTR210_Open(). The field VerFPGA, if FPGA is not downloaded upon connection with the module, shall be valid only after successful invocation of LTR210_LoadFPGA().		
Field	**Type**	**Field description**
Name	CHAR [LTR210_NAME_ SIZE]	Module name (ASCII-string ending with zero)
Serial	CHAR [LTR210_SERIAL_ SIZE]	Serial number of the module (ASCII-string ending with zero)
VerFPGA	WORD	Firmware version of the module FPGA (valid only after downloading)
VerPLD	BYTE	PLD firmware version
CbrCoef	TLTR210_CBR_COEF [LTR210_CHANNEL_CNT] [8]	Factory calibration factors (initial LTR210_RANGE_CNT are valid per channel, the rest are reserved)
AfcCoefFreq	double	Frequency (Hz) that corresponds to AFC adjustment factors
AfcCoef	double [LTR210_ CHANNEL_CNT] [8]	Module AFC fall-off factor at the AfcCoefFreq frequency. Represents ratio of measured sine-wave signal amplitude at specified frequency to the amplitude of actual signal. Factors are downloaded from the module Flash-memory upon circuit activation. May be used for AFC adjustment, if any. Initial LTR210_RANGE_CNT of factors are valid per channel, the rest are reserved.
AfcIirParam	TLTR210_AFC_IIR_COEF [LTR210_CHANNEL_CNT] [8]	Parameters for calculation of IIR filter factors used for adjustment of AFC ranges 10V and 5V. These parameters are stored in the module Flash-memory. Initial LTR210_AFC_IIR_COR_RANGE_CNT of factors are valid per channel, the rest are reserved.
Reserved	DWORD [32]	Backup fields

4.2.4 ADC channel configurations.

Type: TLTR210_CHANNEL_CONFIG		
Description: The structure containing ADC one channel configurations.		
Field	**Type**	**Field description**
Enabled	BOOLEAN	Acquisition allowed by this channel flag
Range	BYTE	Preset range - constant form e_LTR210_ADC_RANGE
Mode	BYTE	Measurement mode - constant from e_LTR210_CH_MODE
DigBitMode	BYTE	Additional bit operation mode in the input data thread of the channel. Constant from e_LTR210_DIG_BIT_MODE
Reserved	BYTE [4]	Backup fields (shall not be changed by the user)
SyncLevelL	double	Lower hysteresis threshold in the analogue synchronization (V). Falling edge is a signal reduction beyond the SyncLevelL, if before the signal exceeded SyncLevelH. Rising edge is an excess of SyncLevelH, if before the signal was beyond SyncLevelL. Must be within preset range.
SyncLevelH	double	Upper hysteresis threshold in the analogue synchronization (V). Must be within preset range, not less than SyncLevelL.
Reserved2	DWORD [10]	Backup fields (shall not be changed by the user)

4.2.5 Module configurations.

Type: TLTR210_CONFIG		
Description: The structure contains all module configurations which shall be filled by the user prior to invocation of LTR210_SetADC().		
Field	**Type**	**Field description**
Ch	TLTR210_CHANNEL_CONFIG [LTR210_CHANNEL_CNT]	ADC channel configurations
FrameSize	DWORD	Point size per channel in the frame in the frame-by-frame acquisition mode
HistSize	DWORD	Pre-history size (number of points per channel in the frame, measured prior to occurrence of synchronization event)

SyncMode	BYTE	Frame acquisition condition (synchronization event) One of values of e_LTR210_SYNC_MODE
GroupMode	BYTE	Operating mode within a group of modules. One of values of e_LTR210_GROUP_MODE
AdcFreqDiv	WORD	ADC frequency divider value minus 1. May vary from 0 to LTR210_ADC_FREQ_DIV_MAX-1
AdcDcmCnt	DWORD	Value of ADC data reduction factor minus 1. May vary from 0 to LTR210_ADC_DCM_CNT_MAX-1.
FrameFreqDiv	DWORD	Frame acquisition start frequency divider for SyncMode = LTR210_SYNC_MODE_PERIODIC. Frame frequency is equal to $\frac{10^6}{FrameFreqDiv+1}$Hz
Flags	DWORD	Flags of configuration (combination of e_LTR210_CFG_FLAGS)
IntfTransfRate	BYTE	Data-to-interface output rate (one of values from e_LTR210_INTF_TRANSF_RATE). Maximum rate (500 KWords/s) is set by default. If preset rate exceeds maximum interface rate for crate whereto the module is installed, then maximum rate supported by this crate shall be set
Reserved	DWORD [39]	Backup fields (shall not be changed by the user)

4.2.6 Module state parameters.

Type: TLTR210_STATE		
Description: The structure containing the module parameters which shall be used read-only by the user because they are changed inside of ltr210api functions only.		
Field	**Type**	**Field description**
Run	BOOLEAN	Data acquisition start flag
RecvFrameSize	DWORD	Number of words in accepted frame including status (set after invocation of LTR210_SetADC())

AdcFreq	double	Calculated ADC acquisition frequency, Hz (set after invocation of LTR210_SetADC())
FrameFreq	double	Calculated frame sequence frequency for synchronization mode LTR210_SYNC_MODE_PERIODIC (set after invocation of LTR210_SetADC())
AdcZeroOffset	double [LTR210_ CHANNEL_CNT]	Measured ADC null offset values in codes (simplified to LTR210_ADC_SCALE_CODE_MAX) for current configurations. Set upon invocation of LTR210_MeasAdcZeroOffset(). May be used for additional adjustment of ADC null offset.
Reserved	DWORD [4]	Backup fields

4.2.7 Module handle.

Type: TLTR210		
Description: The structure contains full information about module configurations and current state of module connection. Applied by all functions for module operation.		
Field	**Type**	**Field description**
Size	INT	Structure size. Filled in LTR210_Init().
Channel	TLTR	The structure containing the state of client connection to ltrd service. Is not used by the user directly.
Internal	PVOID	Opaque structure index with internal parameters used by library only and unaccessible for the user.
Cfg	TLTR210_CONFIG	Module configurations. Filled by the user before invocation of LTR210_SetADC().
State	TLTR210_STATE	Module state and calculated parameters. Fields are changed by the library functions. Can be used read-only by the user program.
ModuleInfo	TINFO_LTR210	Module information

4.2.8 Additional information about accepted reading.

Type: TLTR210_DATA_INFO		
Description: The structure contains additional information about accepted and processed data word extracted from data fields of the accepted word.		
Field	**Type**	**Field description**
DigBitState	BYTE	The least significant bit corresponds to the additional bit value transferred along with the data thread. That means that this bit is set by one of constants from e_LTR210_DIG_BIT_MODE in the field DigBitMode of set-up per channel at the configuration stage. Other bits may be used in the future, thus the DigBitState \& 1 value shall be checked in the analysis.
Ch	BYTE	Number of channel which the accepted word corresponds to (0 - initial, 1 - secondary)
Range	BYTE	Channel range preset during measuring of this reading
Reserved	BYTE	Reserved field (always equal to 0)

4.2.9 Information about the processed frame status

Type: TLTR210_FRAME_STATUS		
Description: This structure contains fields that inform of status of processed frame. It should be checked to ensure that accepted frame data is valid and no error has occurred upon its recording. The field Result may be used for this purpose, and fields Flags provide additional information which, particularly, enables identification of error cause.		
Field	**Type**	**Field description**
Result	BYTE	Code of the frame processing result (one of values e_LTR210_FRAME_RESULT). Allows for determination whether the frame end is found and whether the frame data is valid
Reserved	BYTE	Reserved field (always equal to 0)
Flags	WORD	Additional flags from e_LTR210_STATUS_FLAGS, presenting information about status of the module and accepted frame. Several flags combined via logical "OR" can be transmitted.

4.2.10 Type of function for FPGA downloading process indication

Type: TLTR210_LOAD_PROGR_CB
Definition: typedef void(APIENTRY * TLTR210_LOAD_PROGR_CB) (void *cb_data, TLTR210 *hnd, DWORD done_size, DWORD full_size)
Description: Type for callback-function called upon downloading of the module FPGA firmware which may be used for indication of the process sequence. A pointer to the function of given type may be transmitted to LTR210_LoadFPGA().
Parameters: **cb_data** — a pointer transmitted to LTR210_LoadFPGA() as a parameter cb_data **hnd** — a pointer to the module handle for which FPGA firmware is downloaded **done_size** — number of successfully written bites **full_size** — full size of firmware file in bites

4.3 Functions

4.3.1 The functions of initialization and dealing with connection to the module.

4.3.1.1 Module handle initialization

Format: INT LTR210_Init (TLTR210 *hnd)
Description: The function initializes fields of the module handle structure with values set by default. This function must be called for every structure by TLTR210 prior to invocation of other functions.
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.1.2 Opening connection to module.

Format: INT LTR210_Open (TLTR210 *hnd, DWORD ltrd_addr, WORD ltrd_port, const CHAR *csn, WORD slot)
Description: The function makes connection to the module in accordance with parameters transmitted, check for module availability and reads the information about it. Shall be called prior to manipulate with the module. After completion of operation it is necessary to close connection using LTR210_Close().
Parameters: **hnd** — Module handle **ltrd_addr** — IP-address of the computer where ltrd service in 32-bit format has been started (described in section "IP-addresses setting format" of instruction for library ltrapi). If the ltrd service is started at the same computer as the program

calling this function the LTRD_ADDR_DEFAULT can be transmitted as the address. **ltrd_port** — TCP-port for connection to ltrd service. LTRD_PORT_DEFAULT is used by default. **csn** — serial number of the crate where the targeted module is located. Presenting ASCII-string ending with zero. Empty string or zero index can be transmitted to connect to the first found crate. **slot** — Number of crate slot where the targeted module is located. Value from LTR_CC_CHNUM_MODULE1 to LTR_CC_CHNUM_MODULE16.
Returned value: Error code

4.3.1.3 Closing connection to module.

Format: INT LTR210_Close (TLTR210 *hnd)
Description: The function closes previously opened connection using LTR210_Open() Shall be called after manipulating with the module completion. With any returned value after calling this function the relevant handle can not be used without opening a new connection.
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.1.4 Checking for opening connection to module.

Format: INT LTR210_IsOpened (TLTR210 *hnd)
Description: The function checks whether the connection to the module is currently opened. If the connection is opened the function returns LTR_OK, if it is closed — error code LTR_ERROR_CHANNEL_CLOSED.
Parameters: **hnd** — Module handle
Returned value: Error code (LTR_OK, if the connection is established).

4.3.1.5 Checking for module FPGA firmware downloading.

Format: INT LTR210_FPGAIsLoaded (TLTR210 *hnd)
Description: The function checks whether the module FPGA firmware is currently downloaded. If the firmware is downloaded, the function returns LTR_OK, otherwise — LTR_ERROR_FPGA_IS_NOT_LOADED. If FPGA is not downloaded, then it should be done via LTR210_LoadFPGA().
Parameters: **hnd** — Module handle

Returned value: Error code (LTR_OK, if the firmware is downloaded).

4.3.1.6 Downloading of the module FPGA firmware.

Format: INT LTR210_LoadFPGA (TLTR210 *hnd, const char *filename, TLTR210_LOAD_PROGR_CB progr_cb, void *cb_data)
Description: The function downloads the module FPGA firmware out of specified file. If null index or a blank string is used as the file name, then for OS Windows the firmware version integrated into the library as resource is downloaded, as for OS Linux the standard path file is used (installation path in the library assembly). This function should be executed after opening of connection with the module prior to further operation, if the firmware was not downloaded before. Check whether the FPGA firmware is downloaded may be performed via LTR210_FPGAIsLoaded().
Parameters: **hnd** — Module handle. **filename** — File name with the module FPGA firmware or a blank string. **progr_cb** — a pointer to the function which shall be used during download and may be used for downloading process indication. If it is not used, then the null index may be transmitted. **cb_data** — if the function progr_cb is indicated, then this index shall be transmitted to it as the homonym parameter.
Returned value: Error code

4.3.2 Module setting change functions

4.3.2.1 Setting storing in module.

Format: INT LTR210_SetADC (TLTR210 *hnd)
Description: The function transmits the configurations relevant to field values of module handle Cfg field to the module. Must be called prior to the module initiation. This function may be also called upon triggered data acquisition process to change limited number of parameters which may be changed immediately.
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.2.2 Setting of specified ADC acquisition frequency.

Format: INT LTR210_FillAdcFreq (TLTR210_CONFIG *cfg, double freq, DWORD flags, double *set_freq)
Description: The function fills the fields AdcFreqDiv and AdcDcmCnt with values for which ADC

acquisition frequency shall be as nearly to the given values as possible. It may be called prior to LTR210_SetADC() instead of manual filling of specified fields.
Parameters: **cfg** — a pointer to the structure with the module configuration **freq** — ADC acquisition frequency (Hz) which should be set **flags** — flags (reserved — 0 shall always be transmitted) **set_freq** — in the given variable the actual frequency value corresponding to selected parameters is stored. The null index can be transmitted if this information is no longer needed.
Returned value: Error code

4.3.2.3 Setting of specified frame spacing frequency.

Format: INT LTR210_FillFrameFreq (TLTR210_CONFIG *cfg, double freq, double *set_freq)
Description: The function sets value for the field FrameFreqDiv such that the frame sequence frequency in mode LTR210_SYNC_MODE_PERIODIC would be as nearly close to the given value as possible. It may be called prior to LTR210_SetADC() instead of manual filling of FrameFreqDiv.
Parameters: **cfg** — a pointer to the structure with the module configuration **freq** — frame sequence frequency (Hz) which should be set **set_freq** — in this variable the actual frequency value corresponding to selected parameters is stored. The null index can be transmitted if this information is no longer needed.
Returned value: Error code

4.3.3 Data acquisition control functions

4.3.3.1 Start of data acquisition.

Format: INT LTR210_Start (TLTR210 *hnd)
Description: ADC data record is initiated by the module when this function is called. When continuous data acquisition (SyncMode = LTR210_SYNC_MODE_CONTINUOUS) is triggered, this function initiates data acquisition from ADC and data output to the crate. In the frame-by-frame acquisition upon invocation of the function, the module switches to the mode of recording into the internal buffer and waiting for synchronization events to provide frame output to the crate. Also upon and after the function is called, the module starts periodic status transmission, if enabled. At least one ADC channel should be enabled and the module should be configured via LTR210_SetADC().

Parameters: **hnd** — Module handle
Returned value: Error code

4.3.3.2 Stopping of data acquisition.

Format: INT LTR210_Stop (TLTR210 *hnd)
Description: Upon invocation of this function the module stops data record, synchronization event expectation and periodic statuses output. In addition, all transmitted but not read data from the module are read and thrown.
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.3.3 Program initiation of frame acquisition.

Format: INT LTR210_FrameStart (TLTR210 *hnd)
Description: The function sends command to the module to initiate program synchronization event. Following the event the module starts frame acquisition (and generates pulse in the master mode at the SYNC outlet for slave modules). May be used exclusively in the mode LTR210_SYNC_MODE_INTERNAL. In the invocation of this function data record from ADC shall be already enabled via LTR210_Start().
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.3.4 Waiting for asynchronous event from the module.

Format: INT LTR210_WaitEvent (TLTR210 *hnd, DWORD *event, DWORD*status, DWORD tout)
Description: The function tries to accept a word from the module which corresponds to one of probable asynchronous events during specified timeout. The function returns control upon receipt of the first word corresponding to any event. If no event occurred during preset timeout, then it is deemed to be a normal termination (function returns LTR_OK), but LTR210_RECV_EVENT_TIMEOUT returns as the event code. To provide continuous data acquisition the function always returns the event LTR210_RECV_EVENT_SOF and does not accept any data.
Parameters: **hnd** — Module handle. **event** — in this variable the occurred event code is returned

(LTR210_RECV_EVENT_TIMEOUT, if no event occurred during specified time interval). status — unless NULL is indicated, then on an event LTR210_RECV_EVENT_KEEPALIVE in this variable the information about the module status is saved in the form of combination of flags e_LTR210_STATUS_FLAGS. On other events this parameter is not subject to change. **tout** — event timeout, in milliseconds.
Returned value: Error code

4.3.3.5 Data receiving from the module.

Format: INT LTR210_Recv (TLTR210 *hnd, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)
Description: The function receives the requested number of words from the module. The returned words are in the specific format containing the service information. To process the received words and to receive ADC values the function LTR210_ProcessData() is used. In the frame-by-frame acquisition mode the function is used for frame receipt after identification of the frame start via LTR210_WaitEvent(), and in the continuous acquisition it may be called without additional call, as for the rest LTR modules. The function returns control whether after receipt of requested number of words, or on expiration of timeout, and upon detection of the frame stop, if frame-by-frame acquisition mode is used.
Parameters: **hnd** — Module handle. **data** — Array where the received words will be saved. It must have size of "size" of 32-bit words. **tmark** — index to the array with size of "size" of 32-bit words, where values of synchro-labels will be saved, that correspond to the received data. The label generating is configurated for the crate or for special module individually. The synchro-labels are described in details in section "Synchro-labels" of the instruction for the library ltrapi. If the synchro-tags are not used the null index can be transmitted as the parameter. **size** — requested quantity of 32-bit words per receipt. In the frame-by-frame mode the frame size may be used from the field RecvFrameSize of the module handle. **timeout** — Timeout for operation execution in milliseconds. If the requested number of words is not received during the pre-set time, the function still will return control, having returned the actual number of the received words as a result.
Returned value: Negative value (less than zero) corresponds to the error code. The value greater than or equal to zero corresponds to the actual number of the received and stored in the array "data" words.

4.3.3.6 Processing of the words received from the module.

Format: INT LTR210_ProcessData (TLTR210 *hnd, const DWORD*src, double *dest, INT *size, DWORD flags, TLTR210_FRAME_STATUS*frame_status, TLTR210_DATA_INFO *data_info)
Description: This function is used to process words received from the module via LTR210_Recv(). The function checks service fields of the received words, extracts useful information with readings, and upon indicating of the flag LTR210_PROC_FLAG_VOLT, converts readings into Volts. The function also analyzes the frame status word, if any, and returns the result in parameter frame_status. To receive additional service information a pointer to the array data_info may be transmitted. The function checks data integrity using the counter from the service information. By default the function assumes, that all received data are processed and only once by checking the counter continuity and between the function calls. If this condition is ruled out it is necessary to transmit the flag LTR210_PROC_FLAG_NONCONT_DATA.
Parameters: **hnd** — module handle. **src** — a pointer to the array containing words received from the module via LTR210_Recv() which are to be processed. **dest** — index to the array where the processed data will be saved. Sequence order corresponds to the order in the input array (i.e. if both channels are enabled, then channel readings shall alternate). **size** — accepts the array size src for processing at the inlet. Returns the number of saved samples in the array dest at output upon successful completion. **flags** — Flags from e_LTR210_PROC_FLAGS regulating function operation. Some flags can be integrated through the logic OR. **frame_status** — in this structure the state of accepted frame is saved, which is generated based on the status word transmitted after the last word of status. Frame data validity may be inferred by status. In absence of status word in processed data (i.e. data corresponds only to the part of the frame, excluding its end), then status is returned with the code LTR210_FRAME_RESULT_PENDING. **data_info** — array whereto information extracted from service fields of processed words shall be saved. In particular, state of additional service bit is saved in it. If this information is not needed, the null index may be transmitted.
Returned value: Error code.

4.3.4 Auxiliary functions

4.3.4.1 Null offset measurement

Format: INT LTR210_MeasAdcZeroOffset (TLTR210 *hnd, DWORD flags)
Description: The function performs acquisition of one frame in the mode of own null measurement using range values preset in configuration and saves the result in fields State.AdcZeroOffset. After measurement the previous ADC settings are reset.

Measured values may be used for adjustment of accepted data upon transmission of the flag LTR210_PROC_FLAG_ZERO_OFFS_COR in LTR210_ProcessData(), that allows for consideration of nul escape for specific conditions. This function is called after module configuration immediately prior to initiation of ADC data record LTR210_Start(). In case of change in ranges it is necessary to remeasure null offset.
Parameters: **hnd** — Module handle. **flags** — Flags (reserved - 0 should be transmitted)
Returned value: Error code.

4.3.4.2 Acceptance of the previous interval upon acceptance of the last word.

Format: INT LTR210_GetLastWordInterval (TLTR210 *hnd, DWORD*interval)
Description: The function is used for acquisition of time interval passed since successful receipt of the last data word from the module (via LTR210_WaitEvent() or LTR210_Recv()). It may be used upon enabled periodic status sending (live signal) to check correct module operation. If during acquisition upon enabled sending of status this time exceeds permitted interval (which must include allowed delays in the interface crate -> PC), then this event may be deemed as the module failure.
Parameters: **hnd** — Module handle. **interval** — time in milliseconds since successful acceptance of the last word.
Returned value: Error code.

4.3.4.3 Receiving error message.

Format: LPCSTR LTR210_GetErrorString (INT err)
Description: The function returns the string that corresponds to the transmitted error code In CP1251 coding for OS Windows or UTF-8 coding for OS Linux. The function can process both the errors from ltr210api and general codes of errors from ltrapi.
Parameters: **err** — Error code
Returned value: Index for the string containing the message error.

4.3.4.4 Downloading of factors to FPGA.

Format: INT LTR210_LoadCbrCoef (TLTR210 *hnd)
Description: The function executes factor download from ModuleInfo.CbrCoef to the module FPGA for further auto-adjustment. As in API functions the factors are automatically downloaded from Flash-memory after downloading of FPGA, this function is

integrated only to provide application of factors. All it takes is filling in fields in ModuleInfo.CbrCoef and calling the function. The function may be called only if FPGA is downloaded.
Parameters: **hnd** — Module handle.
Returned value: Error code.