Multichannel data-acquisition systems

LTR25

Programmer manual

Revision 1.0.1
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	02.06.2014	The document first revision.
1.0.1	10.09.2014	The problem to connect the library to the projects is discussed in the individual document.

Contents

1. What this document is about 5

2. Library installation and connection to the project 6

3. General approach to working with the library 7

3.1 General algorithm to working with the module 7

3.2 Module setting 7

3.2.1 ADC channels setting 8

3.3 Continuity or short circuit testing 8

3.4 The peculiarities of data calibration by the module 9

4. Constants, types of data and library functions 10

4.1 Constants and enumerations. 10

4.1.1 Constants and macros. 10

4.1.2 Error codes specific to LTR25 module. 11

4.1.3 ADC acquisition frequency codes 11

4.1.4 Module data formats. 12

4.1.5 Current source values. 12

4.1.6 Data processing flags. 12

4.1.7 Input channel state. 13

4.2 Data types. 13

4.2.1 Calibration coefficients 13

4.2.2 Set of coefficients for module AFC correction 13

4.2.3 Module information 14

4.2.4 ADC channel configurations. 14

4.2.5 Module configurations. 15

4.2.6 Module current state parameters. 15

4.2.7 Module control structure. 16

4.3 Functions 16

4.3.1 The functions of initialization and dealing with connection to the module 16

4.3.2 Module setting change functions 18

4.3.3 Data acquisition control functions 18

4.3.4 Functions for manipulating with module flash-memory 21

4.3.5 Auxiliary functions 22

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 LTR25 module using library ltr25api provided by 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 chapter describes the typical sequence of actions during when operating with LTR25 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 TLTR25 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 LTR25_Init().
3. Make connection with targeted module with the LTR25_Open() function.
4. Fill the necessary fields with module configurations of module handle substructure Cfg and call LTR25_SetADC() to record configurations in the module.
5. Starting data acquisition using LTR25_Start().
6. Data receiving and processing as described below.
7. Upon completion of operation make stopping data acquisition with the LTR25_Stop().
8. Close connection with the module by calling the function LTR25_Close().

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

1. Receiving of set number of samples using LTR25_Recv().
2. Processing of received samples using LTR25_ProcessData().

3.2 Module setting

Module setting is performed in the same manner for the most of other LTR modules: the values of all module parameters are recorded firstly in the module handle structure relevant fields then the function LTR25_SetADC() is called which transfers the values of these fields to the module. The module shall be set prior to start initial data acquisition. Do not change the configurations during data acquisition started.

It should be noted that all fields relating to module setting are integrated in structure of TLTR25_CONFIG type (module handle Cfg field). These fields only can be changed by the user in manual way in module handle under normal operation and these fields only affect on the

parameters recorded by LTR25_SetADC(). The following parameters are defined during the module setting:

- ADC acquisition frequency. One of 8 predefined acquisition frequencies can be selected by setting the corresponding code in FreqCode field.
- ADC channels configurations (section ADC channels setting)
- The transmitted data format is defined by DataFmt field. Bit depth of transmitted samples and the number of transmitted words per one ADC sample depends on the format.
- Current source value (10 or 2.86 mA) is defined for all channels by ISrcValue field.

Upon completion LTR25_SetADC() some parameters being derivative of configurations from TLTR25_CONFIG are calculated. These parameters is recorded in corresponding structure fields of module state of TLTR25_STATE type (module handle State field). For example, the corresponding ADC acquisition frequency in Hz is recorded based on determinated frequency code in the field AdcFreq .

3.2.1 ADC channels setting

Each LTR25 module has eight channels handling the conversion in parallel. The record can be permitted as per any of these eight channels. However, it should be noted, that the maximum number of synchronously allowed channels depends on configurated acquisition frequency and sample formats (see in detail the document "LTR Crate System. User Manual").

Each channel configurations are integrated in structure TLTR25_CHANNEL_CONFIG. The array of structure of LTR25_CHANNEL_CNT 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 allowed for this channel. Defined by the field Enabled.

3.3 Continuity or short circuit testing

LTR25 module allows to determine continuity or short circuit events for each allowed channel under acuisition started. The specialized code indicating occurrence of the event is inserted in transmitted word instead of ADC sample itself when determining one of these situations. The function LTR25_ProcessData() analyses these codes and generates channel state for each allowed channel. The state is calculated across the block (but individually per channel), in other words, if disconnection sign is detected for at least one sample of the block processed the state of this channel LTR25_CH_STATUS_OPEN will be returned. State order corresponds to allowed channels order.

It should be noted, that the determining of disconnection and SC state is the inertial process and state configuration is performed with delay relatively to real events.
For more detail, see in the document "LTR Crate System. User Manual".

3.4 The peculiarities of data calibration by the module

It is to be noted, that unlike most of the rest LTR modules (for example, LTR24) the data calibration and AFC correction is performed by hardware inside the module but not by software. Thus, there are no instructions on calibration performance in LTR25_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.

It is to be noted, that calibration coefficients are determined and stored individually for the first LTR25_CBR_FREQ_CNT acquisition frequencies. Because the rest frequencies are obtained from the first LTR25_CBR_FREQ_CNT by filtration and decimation in FPGA the initial frequency coefficients are used for them. The peculiar calibration coefficients are used for each of 8 channels.

FPGA performs calibration immediately by the formula $Y = (X+Offset)*Gain$, where, X — sample from ADC expanded up to 32-bits, Y — calibrated data, Offset — scale offset (32-bit code), a Gain — scale coefficient. In addition, before calibration performing the ADC input value is expanded up to 32 bits. The calibrated 32-bit samples are at the output. For 32-bit format these samples transmitted to crate and for 20-bit format the high-order samples only are transmitted.

In addition, the code LTR25_ADC_SCALE_CODE_MAX corresponds to voltage equal to maximum voltage for the given range.

If the user wants to set his calibration coefficients he shall change field values CbrCoef in the structure with information about the module prior to call LTR25_SetADC() where coefficients are downloaded in FPGA based on ADC given frequency.

Chapter 4

Constants, types of data and library functions

4.1 Constants and enumerations.

4.1.1 Constants and macros.

Constant	Value	Description
LTR25_CHANNEL_CNT	8	Number of ADC channels in one LTR25 module
LTR25_FREQ_CNT	8	Number of sampling frequencies.
LTR25_CBR_FREQ_CNT	2	Number of frequencies for which the calibration coefficients are saved
LTR25_I_SRC_VALUE_CNT	2	Number of current source values
LTR25_NAME_SIZE	8	Field size with module name.
LTR25_SERIAL_SIZE	16	Field size with module serial number.
LTR25_ADC_RANGE_PEAK	10	Maximum peak value in Volts for the module measuring range
LTR25_ADC_SCALE_CODE_MAX	2000000000	ADC code corresponding to maximum peak value
LTR25_FLASH_USERDATA_ADDR	0x0	Address of the flash-memory user area origin
LTR25_FLASH_USERDATA_SIZE	0x100000	Size of user area flash-memory
LTR25_FLASH_ERASE_BLOCK_SIZE	4096	Store erasing block minimum size All erasing operations shall be a multiple of this size

4.1.2 Error codes specific to LTR25 module.

Type: e_LTR25_ERROR_CODES		
Description: Error code defined and used in ltr25api only. Other error codes used by different modules are defined in ltrapi.h		
Constant	**Value**	**Description**
LTR25_ERR_FPGA_FIRM_TEMP_RANGE	-10600	FPGA firmware for invalid temperature range is downloaded
LTR25_ERR_I2C_ACK_STATUS	-10601	Exchange error when accessing to ADC registers through the interface I2C
LTR25_ERR_I2C_INVALID_RESP	-10602	Failing response on command when addressing to ADC registers through the interface I2C
LTR25_ERR_INVALID_FREQ_CODE	-10603	Invalid ADC frequency code
LTR25_ERR_INVALID_DATA_FORMAT	-10604	Invalid ADC data format
LTR25_ERR_INVALID_I_SRC_VALUE	-10605	Invalid current source value
LTR25_ERR_CFG_UNSUP_CH_CNT	-10606	Invalid number of ADC channels for the given frequency and format
LTR25_ERR_NO_ENABLED_CH	-10607	No one ADC channel enabled
LTR25_ERR_ADC_PLL_NOT_LOCKED	-10608	ADC PLL capture error
LTR25_ERR_ADC_REG_CHECK	-10609	ADC recorded register values check error
LTR25_ERR_LOW_POW_MODE_NOT_ CHANGED	-10610	ADC low power mode change error
LTR25_ERR_LOW_POW_MODE	-10611	Module is under low power mode

4.1.3 ADC acquisition frequency codes

Type: e_LTR25_FREQS		
Description: ADC acquisition frequency codes		
Constant	**Value**	**Description**
LTR25_FREQ_78K	0	78.125 kHz
LTR25_FREQ_39K	1	39.0625 kHz
LTR25_FREQ_19K	2	19.53125 kHz
LTR25_FREQ_9K7	3	9.765625 kHz

LTR25_FREQ_4K8	4	4.8828125 kHz
LTR25_FREQ_2K4	5	2.44140625 kHz
LTR25_FREQ_1K2	6	1.220703125 kHz
LTR25_FREQ_610	7	610.3515625 Hz

4.1.4 Module data formats.

Type: e_LTR25_FORMATS		
Description: Module data formats		
Constant	**Value**	**Description**
LTR25_FORMAT_20	0	20-bit integer format (1 word per sample)
LTR25_FORMAT_32	1	32-bit integer format (2 words per sample)

4.1.5 Current source values.

Type: e_LTR25_I_SOURCES		
Description: Current source values.		
Constant	**Value**	**Description**
LTR25_I_SRC_VALUE_2_86	0	2.86 mA.
LTR25_I_SRC_VALUE_10	1	10 mA.

4.1.6 Data processing flags.

Type: e_LTR25_PROC_FLAGS		
Description: Flags controlling the operation of the function LTR25_ProcessData()		
Constant	**Value**	**Description**
LTR25_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 LTR25_ADC_SCALE_CODE_MAX corresponds to maximum voltage for the specified range.
LTR25_PROC_FLAG_NONCONT_DATA	0x0100	LTR25_ProcessData() assumes by default that all received data are transmitted to it to be processed and check the counter continuity not only within the passed 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.

4.1.7 Input channel state.

Type: e_LTR25_CH_STATUS		
Description: LTR25_ProcessData() is passed back for each allowed channel and defines whether disconnection or short circuit were detected for this channel in the block processed by LTR25_ProcessData()		
Constant	**Value**	**Description**
LTR25_CH_STATUS_OK	0	On-load channel
LTR25_CH_STATUS_SHORT	1	Short circuit is detected
LTR25_CH_STATUS_OPEN	2	Electrical open is detected

4.2 Data types.

4.2.1 Calibration coefficients

Type: TLTR25_CBR_COEF		
Description: The structure storing the calibration coefficients for one channel and range.		
Field	**Type**	**Field description**
Offset	float	Offset code
Scale	float	Scale coefficient

4.2.2 Set of coefficients for module AFC correction

Type: TLTR25_AFC_COEFS		
Description:		
Field	**Type**	**Field description**
AfcFreq	double	Signal frequency for which the amplitude ratio from FirCoef is cleared
FirCoef	double [LTR25_ CHANNEL_CNT]	Set of sine signal measured amplitude and actual amplitude ratios for maximum sampling frequency and frequency of the signal from AfcFreq for each channel

4.2.3 Module information

Type: TINFO_LTR25		
Description: The structure containing the information about module circuit firmware versions and information from module Flash-memory (serial number, calibration coefficients). All fields are filled when calling LTR25_Open()		
Field	**Type**	**Field description**
Name	CHAR [LTR25_NAME_SIZE]	Module name (“LTR25”).
Serial	CHAR [LTR25_SERIAL_ SIZE]	Module serial number.
VerFPGA	WORD	FPGA firmware version
VerPLD	BYTE	PLD firmware version
BoardRev	BYTE	Board revision
Industrial	BOOL	Flag indicating industrial version or not
Reserved	DWORD [8]	Reserved fields. Always equal to 0
CbrCoef	TLTR25_CBR_COEF [LTR25_CHANNEL_CNT] [LTR25_CBR_FREQ_CNT]	Module calibration coefficients. Are read from module Flash-memory when calling LTR25_Open() or LTR25_GetConfig() and are downloaded in FPGA to be used when calling LTR25_SetADC()
AfcCoef	TLTR25_AFC_COEFS	Coefficients for module AFC correction
Reserved2	double [32 *LTR25_CHANNEL_ CNT-sizeof(TLTR25_ AFC_ COEFS)/sizeof(double)]	Backup fields

4.2.4 ADC channel configurations.

Type: TLTR25_CHANNEL_CONFIG		
Description: The structure containing ADC one channel configurations.		
Field	**Type**	**Field description**
Enabled	BOOL	Acquisition allowed by this channel flag
Reserved	DWORD [11]	Backup fields (shall not be changed by the user)

4.2.5 Module configurations.

Type: TLTR25_CONFIG		
Description: The structure contains all module configurations which shall be filled by the user prior to call LTR25_SetADC().		
Field	**Type**	Field description
Ch	TLTR25_CHANNEL_CONFIG [LTR25_CHANNEL_CNT]	ADC channel configurations
FreqCode	BYTE	Code defining the required ADC acquisition frequency. One of the values of e_LTR25_FREQS
DataFmt	BYTE	The format where ADC samples are transmitted from the module. One of the values of e_LTR25_FORMATS. The format determines also the number of transmitted words for one sample and affects on maximum number of allowed channels
ISrcValue	BYTE	Current source value used One of the values of e_LTR25_I_SOURCES
Reserved	DWORD [50]	Backup fields (shall not be changed by the user)

4.2.6 Module current state parameters.

Type: TLTR25_STATE		
Description: The structure containing the module parameters which shall be used read-only by the user because they are changed withing ltr25api function only.		
Field	**Type**	**Field description**
FpgaState	BYTE	FPGA current state. One of the values of e_LTR_FPGA_STATE
EnabledChCnt	BYTE	Number of allowed channels. Is defined after call LTR25_SetADC()
Run	BOOL	Data acquisition start flag
AdcFreq	double	ADC defined frequency Updated after call LTR25_SetADC()
LowPowMode	BOOL	Module low power mode flag. Do not configurate ADC or start data acquisition under this mode. This mode control is performed used LTR25_SetLowPowMode()
Reserved	DWORD [31]	Backup fields

4.2.7 Module control structure.

Type: TLTR25		
Description: Stores the module current settings, information about its state, communication circuit structure. Is transmitted to the most of library functions. Some structure fields can be changed by the user to configurate module parameters. Requires initialization by function LTR25_Init() before use.		
Field	**Type**	**Field description**
Size	INT	Structure size. Filled in LTR25_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	TLTR25_CONFIG	Module configurations. Filled by the user before call LTR25_SetADC().
State	TLTR25_STATE	Module state and calculated parameters. Fields are changed by the library functions. Can be used read-only by the user program.
ModuleInfo	TINFO_LTR25	Module information

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 LTR25_Init (TLTR25 *hnd)
Description: The function initializes the structure fields of the module handle using default values. This function must be called for every structure by TLTR25 prior to call other functions.
Parameters: hnd — Module handle
Returned value: Error code

4.3.1.2 Opening connection to module.

Format: INT LTR25_Open (TLTR25 *hnd, DWORD ltrd_addr, WORD ltrd_port, const CHAR *csn, INT 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. Upon completion of work it is necessary to close connection using LTR25_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 LTR25_Close (TLTR25 *hnd)
Description: The function closes previously opened connection using LTR25_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 LTR25_IsOpened (TLTR25 *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).

4.3.2 Module setting change functions

4.3.2.1 Setting storing in module.

Format: INT LTR25_SetADC (TLTR25 *hnd)
Description: The function transmits the configurations relevant to field values of module handle Cfg field to the module. Shall be called prior to start data acquisition using LTR25_Start().
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.3 Data acquisition control functions

4.3.3.1 Start of data acquisition.

Format: INT LTR25_Start (TLTR25 *hnd)
Description: ADC data acquisition of the modules are started when this function is called. Upon successful completion of this function the ADC is started and the module starts to transmit the received samples to PC which are to be read using LTR25_Recv(). Upon measuring completion it is necessary to call LTR25_Stop() to stop data acquisition. At least one ADC channel should be allowed before it and the module should be configurated using LTR25_SetADC().
Parameters: **hnd** — Module handle
Returned value: Error code

4.3.3.2 Stopping of data acquisition.

Format: INT LTR25_Stop (TLTR25 *hnd)
Description: The module stops data acquisition and ADC data output when this function is being called. 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 Data receiving from the module.

Format: INT LTR25_Recv (TLTR25 *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. The format and number of words for one sample are determined by configuration Cfg.DataFmt To process the received words and to received ADC values the function LTR25_ProcessData() is used. The function returns control either when receives the requested number of words or after timeout. With that the actual received number of words can checked by the returned value.
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 descrived in details in section "Synchro-labels" of the instruction for the library ltrapi. If the synchro-labels are not used the zero index can be transmitted as the parameter. **size** — 32-bit words quantity requested per receive. timeout — timeout to perform the operation 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.

4.3.3.4 Processing of the words received from the module.

Format: INT LTR25_ProcessData (TLTR25 *hnd, const DWORD *src, double *dest, INT *size, DWORD flags, DWORD *ch_status)
Description: This function is used to process the words received from the module using LTR25_Recv(). The function checks the service fields of the received words, takes useful information with samples and upon the indicating of the flag LTR25_PROC_FLAG_VOLT converts the samples in Volts. The function assumes, that the transmitted words are aligned to frame beginning (first word of the first allowed channel). Otherwise, at the beginning the incomplete frame will be thrown and the function will return the error LTR_ERROR_PROCDATA_UNALIGNED.

The function analyses also the disconnection and short circuit flags in the allowed channels. The corresponding status is defined in array element ch_status if such flag is available for at least one sample of the relevant channel.

The sample calibration and AFC correction in module 25 unlike module 24 is performed withing the module by hardware, so, LTR25_ProcessData() has not the such flags.

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 flag LTR25_PROC_FLAG_NONCONT_DATA.

Parameters:

hnd — module handle.

src — index to array containing the words received from the module using LTR25_Recv() which are to be processed.

dest — index to the array where the processed data will be saved. The sequencing corresponds to sequencing in input array (i.e. first sample of the first allowed channel is first, then the first sample of the second channel and, etc.).

size — Takes array size of src at input to be processed. Returns the number of saved samples in the array dest at output upon successful completion.

flags — Flags from e_LTR25_PROC_FLAGS controlling function operation. Some flags can be integrated through the logic OR.

ch_status — The array with size of number of elements relevant to number of allowed channels. Each element saves the channel status (one of the values of e_LTR25_CH_STATUS) determining whether the disconnection or SC flags are available in the corresponding channel. The zero index can be transmitted if this information is no longer needed.

Returned value: Error code.

4.3.3.5 Searching the first frame beginning.

Format: INT LTR25_SearchFirstFrame (TLTR25 *hnd, const DWORD *data, DWORD size, DWORD *index)

Description: The function detects the index of the first word of the first frame beginning in the raw data array received from the module. Can be used to align the data frame on beginning in the event of power outage without acqusition stopping.

If the frame beginning is not found in the received array the function will return the error LTR_ERROR_FIRSTFRAME_NOTFOUND.

Parameters:

hnd — Module handle.

data — index for the array containing the words received from the module using LTR25_Recv() where the frame beginning is being searched.

size — Number of words in array data

index — In this variable the element index corresponding to first frame beginning is returned in the event of the function successful completion.

Returned value: Error code.

4.3.4 Functions for manipulating with module flash-memory

4.3.4.1 Reading of data from module flash-memory

Format: INT LTR25_FlashRead (TLTR25 *hnd, DWORD addr, BYTE *data, DWORD size)
Description: The function reads the data recorded in the module flash-memory per the specified address. Memory array from the address LTR25_FLASH_USERDATA_ADDR with size ofLTR25_FLASH_USERDATA_SIZE bytes is possible for the user.
Parameters: **hnd** — Module handle. **addr** — Memory address starting from which it is necessary to read data. **data** — array for size byte where the data read from the Flash-memory will be recorded **size** — data quantity in bytes to be read
Returned value: Error code.

4.3.4.2 Data writing in module flash-memory

Format: INT LTR25_FlashWrite (TLTR25 *hnd, DWORD addr, const BYTE *data, DWORD size)
Description: The function writes the data in the module flash-memory per the specified address. The writable area should be previously erased using LTR25_FlashErase(). Memory array from the address LTR25_FLASH_USERDATA_ADDR with size ofLTR25_FLASH_USERDATA_SIZE bytes is possible for the user.
Parameters: **hnd** — module handle. **addr** — Memory address beginning from which it is necessary to make writing. **data**— array from size byte with data which shall be written. size — data quantity in bytes to be written
Returned value: Error code.

4.3.4.3 Erasing of module flash-memory space

Format: INT LTR25_FlashErase (TLTR25 *hnd, DWORD addr, DWORD size)
Description: The function erases the area in the module flash-memory per the specified address. The erasing should be performed before data writing. The erasing can be performed by blocks only multiple of LTR25_FLASH_ERASE_BLOCK_SIZE bytes. Memory array from the address LTR25_FLASH_USERDATA_ADDR with size ofLTR25_FLASH_USERDATA_SIZE bytes is possible for the user.
Parameters:

hnd — module handle. **addr** — memory address beginning from which it is necessary to perform erasing size — Size of the erased area in bytes. Shall be multiple of LTR25_FLASH_ERASE_BLOCK_SIZE.
Returned value: Error code.

4.3.5 Auxiliary functions

4.3.5.1 Receiving error message.

Format: LPCSTR LTR25_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 ltr25api and general codes of errors from ltrapi.
Parameters: **err** — Error code
Returned value: Index for the string containing the message error.

4.3.5.2 Reading of information and calibration coefficients.

Format: INT LTR25_GetConfig (TLTR25 *hnd)
Description: The function reads the information from the module flash-memory and updates ModuleInfo fields in module control structure. Because this operation is performed when calling LTR25_Open() this function calling is generally not required. However, this function can be used to recover the coefficients changed into factory in ModuleInfo.
Parameters: **hnd** — Module handle.
Returned value: Error code.

4.3.5.3 Setting the module in low power mode.

Format: INT LTR25_SetLowPowMode (TLTR25 *hnd, BOOL lowPowMode)
Description: The function sets the module in low power mode or sets it from this mode into operational one. Under low power mode the ADC is disconnected and current sources are set at 2.86 mA. No access to ADC registers. This mode can be used for full reset of ADC for this the module shall be under this mode for minimum 5 sec.
Parameters: **hnd** — module handle. **lowPowMode** — If FALSE — module is in operating mode, otherwise— in low power

mode.
Returned value: Error code.

4.3.5.4 Checking for FPGA operation authorization

Format: INT LTR25_FPGAIsEnabled (TLTR25 *hnd, BOOL *enabled)
Description: The function checks for module FPGA operation authorization. FPGA shall be always allowed for data configuration and acquisition.
Parameters: **hnd** — Module handle. **enabled** — In the event of the function successful completion the FALSE is returned, if FPGA is unauthorized or TRUE otherwise.
Returned value: Error code.

4.3.5.5 Module FPGA operation authorization

Format: INT LTR25_FPGAEnable (TLTR25 *hnd, BOOL enable)
Description: The function enables or disables the operation of the module FPGA. FPGA shall be always allowed for data configuration and acquisition. Enabling of FPGA operation is performed in LTR25_Open() in case when FPGA firmware has been found in the module memory and it has been successfully downloaded, for this reason this function is not used under normal operation.
Parameters: **hnd** — module handle. **enable** — If FALSE — FPGA is disabled, otherwise— enabled
Returned value: Error code.