Multichannel data-acquisition systems

Itrapi library

Programmer Manual

Revision 2.0.1
November 2016

L-CARD

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

DAQ SYSTEMS DESIGN, MANUFACTURING & DISTRIBUTION

Authors of the manual:
A.V. Kodorkin, A.S. Emelyanov, 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	23.01.2006	The first revision available for user.
1.0.1	23.04.2006	Time labels description is added.
1.0.2	25.07.2006	The function LTR_SetServerProcessPriority is added
1.0.3	02.04.2007	Multithreading operation features are stated
1.0.4	23.04.2007	More detailed description of the LTR_WARNING_MODULE_IN_USE warning is added and examples are changed
1.0.5	04.05.2008	The description of the function LTR_GetCrateRawData is added
1.0.6	04.09.2008	The examples for the function LTR_GetCrateRawData are added
1.0.7	05.09.2008	The description of LTR_GetCrateRawData and TLTR structure flags is added
1.0.8	03.09.2009	The text was corrected, the description of the function LTR_GetCrateInfo is added, the constant and structure list is complemented, the description of the synchronization and switching function for the LTR-EU crate-controllers service signals
1.0.9	11.01.2010	The descriptions of new API functions are added — control of the server and connections with IP crates from the application program. The paragraph "Network security consideration" is added.
1.0.10	31.03.2010	The description of the function LTR_GetServerVersion is added
2.0.0	14.10.2016	The description is modified. Changes in the "Distinguishing features of the document second version" section are described
2.0.1	07.11.2016	The description of the flag LTR_CRATE_IP_FLAG_RECONNECT for IP-writing and the flag LTR_GETCRATES_FLAGS_WORKMODE_ONLY of the function of acquisition of connected crates information is added

Contents

Chapter 1

What this document is about

This document is mainly intended for the programmers who are going to code for PC for operation with LTR crates using libraries, provided by L-Card Company.

The document assumes that the user has read the document "Starting operating 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. Also the document assumes that the user has read the document "LTR crating system. User Manual".

This document contains description of the only library from the library set for work with LTR - the base library for work with LTR crates, file of which is called ltrapi (as the common library set). In this library common control functions for work with the service ltrd, crate control functions and function of acquisition of the information on the connected crates and modules are contained. Also this document describes some typical principles of work with LTR crates that are common for all modules. With that this document does not contain descriptions of the functions for work with the specific modules as for each module its own designated library with the name ltrXXXapi (where XXX — module number) is provided, that uses function of the base library, and for each library there is its own document ltrXXXapi.pdf with the detailed description of all functions and types of these libraries.

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.

Chapter 2

Distinguishing features of the document second version

This document is the second one, significantly modified version of the descriptions of the library ltrapi. This version contains upgraded information taking into account substitution of the previously used LTR Server program for the service ltrd, capabilities of application of the libraries not only for OS Windows, and includes amendments (compatible with the older versions of the libraries), made in the library version 1.31.x with respect to the library version 1.27.x, to which the first version of the document belongs.

Also, the description of the common approach is modified, and in places the working principle of several functions is more detailed. General description considers some additional issues, including the description of the client connection types with indication of what functions are applicable to what type, the description of the principle of working with synchro-labels, etc.

The following can be noted in the changes associated with newer versions of the libraries:

- All constants and types used in this document have prefix LTR_ or TLTR_, respectively, in order to avoid mix-up with names of other libraries. Old constants and types without prefixes are remained in the library for reverse compatibility and saved in the separate header file “ltrapi_compat_defs.h”, that is ON by default, but, if necessary, can be distinctly disabled by the definition LTRAPI_DISABLE_COMPAT_DEFS.

- Auxiliary functions are added for more convenient opening of the control connection of the required type — LTR_OpenSvcControl() and LTR_OpenCrate().

- Description of new function working only with ltrd, that were not supported by the LTR Server program (stated in the function description) is added.

- New error codes are added, including generic codes that are the same for all modules. Modules' libraries use, first of all, these generic codes from ltrapi and retrieve the same error in the same situations instead of defining their own code in each module for the same cases. Besides, some functions return the error code that in more details indicates the error cause rather than the function where error has occurred (as upper level application already knows what function has returned an error).

- Library behavior in case of attempt to open the connection with the module, connection with which has been already opened, is changed. Now it does not lead to disruption of the first connection, however, re-opened connection has limited capabilities and may not enable to work with the module appropriately. For forced

reset of the first connection the function LTR_ResetModule() has been introduced. More detailed description is provided in the section Multiple connection.

- Infrequently-accessed functions, that were not implemented in ltrd as there are no clients needs for them, were deleted from the description. With that the functions are remained in the library for reverse compatibility. These functions are:
  - LTR_GetCrateRawData()
  - – LTR_SetServerProcessPriority()
  - – LTR_GetServerProcessPriority().
- Also, the functions that were introduced, but actual operation of which was not supported by LTR crates, are deleted (with that the functions are remained in the library for reverse compatibility).
  - LTR_GetIPCrateDiscoveryMode()
  - LTR_SetIPCrateDiscoveryMode()

Chapter 3

Installation and connection of the library to the project

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

Chapter 4

Common approach to operation with the library

4.1 Functions call sequence

When working with the LTR crate system via the library ltrapi the library functions do not work directly with the crates or the modules. Instead, they use the service ltrd, that performs interaction with the crates, establishing the client connection with the service that can be already related to the specific module or crate.

To describe the client connection the structure TLTR, hereafter referred to as the connection descriptor, is used, with which almost all functions of the give library are executed. The program can have several connections with ltrd and for each parallel connection it is necessary to create its own connection descriptor (instance of the TLTR structure).

Thus, before execution of any operations, firstly, it is necessary to establish the client connection and after their execution this connection should be closed.

Consequently, the typical call sequence is as follows:

1. Create the instance of the TLTR structure and initialize it calling the function LTR_Init().
2. Establish the connection with ltrd by one of two ways
    - calling the function LTR_OpenSvcControl() or LTR_OpenCrate() depending on the connection type.
    - or fill in the respective fields of the TLTR structure manually and call LTR_Open()
3. Calling of the control functions or data exchange with the module
4. Upon completion of operation close the connection using LTR_Close()

Possible actions when the connection is established depend on the connection type. Connection types are detailed in the following section.

The client connection between the user program and the service ltrd is implemented via sockets. It enables to establish connection from the application to the service ltrd run on another computer if necessary. Due to this fact connection opening functions use the parameters that state the address of the computer with the service ltrd and the TCP port for connection (when using LTR_Open() these data are set via the fields of the connection type descriptor saddr and sport). In standard mode, when the service and the program are run in the same computer, and default TCP port is used, as an address the constant LTRD_ADDR_DEFAULT is used, and as a port — LTRD_PORT_DEFAULT.

4.2 Types of client connections

Three types of client connections can be specified:

- control connection with the service ltrd
- control connection with the crate
- connection with the specific module

4.2.1 Control connection with the service ltrd

To open the connection the function LTR_OpenSvcControl() is used. Also, for this purpose the function LTR_Open() can be used, previously setting up the channel number in the field cc equal to LTR_CC_CHNUM_CONTROL, and the field csn should be filled in with the special line LTR_CSN_SERVER_CONTROL.

Connection of this type can be implemented even if there are no crates connected.

Using this connection the control functions described in the following sections can be executed:

- Information type functions
- Functions of ltrd service control
- Control functions for crate connection over Ethernet

4.2.2 Control connection with the crate

To open the connection of this type the function LTR_OpenCrate() is used. Also, for this purpose the function LTR_Open() can be used, previously setting up the channel number (field cc) equal to LTR_CC_CHNUM_CONTROL and fill in the field csn with the crate serial number (or an empty line, if it is necessary to establish connection with the first crate).

To establish this type of connection there must be at least one connected crate (at least one crate in the list of active crates of the service ltrd).

This connection enables to execute commands described in the section Crate control functions. Technically, all commands of the control connection with ltrd can be executed via this connection, however, the respective connection is more suitable for this purpose as it does not require mandatory availability of the connected crate.

To determine with what crate this connection will be related, it is usually enough to indicate the crate serial number (or an empty line it the crate is always the only one). However, single crate can be simultaneously connected via two interfaces, e.g. if the crate is configured for operation over Ethernet and connected via this interface, but also connected over USB in the set-up mode. Due to this fact, generally, connection to the crate is determined by two parameters - crate serial number and crate connection interface. Without distinct indication of the interface the connection will be related to the crate using the interface that is the operating one for the crate, i.e. via the interface that can exchange data with the crate modules that is necessary for majority of programs. However, if necessary, crate connection interface can be distinctly indicated using the parameter of the function LTR_OpenCrate() or using the flags en_LTR_CC_Iface of the field cc when using LTR_Open().

4.2.3 Connection with the specific module

Connection of this type is implemented using the function LTR_Open(), with the information on the crate with the required module, previously filled in the TLTR structure, similar to the control connection with the crate, but with the crate slot number indicated in the field cc (instead of LTR_CC_CHNUM_CONTROL), where the required module is inserted, using the values from LTR_CC_CHNUM_MODULE1 to LTR_CC_CHNUM_MODULE16.

Via this connection the client exchanges data with the module over the designated protocol of this module using the functions from the section Functions for data exchange with the modules.

However, the user usually does not need to distinctly open this connection via LTR_Open() and operate using the functions ltrapi, as for work with the modules the designated libraries with their own functions, that use the function ltrapi inside themselves, are used.

4.3 Multiple connection

In case of the control connections the client can simultaneously open several connections both with the service and with the crates (both with one crate and with multiple crates). Each new connection does not depend on the previous one and enables to execute any set of function available for this connection type.

A different situation arises with the connections for work with the specific modules. Although the service ltrd enables to establish several connections with a single module simultaneously, but the protocol of working with the modules is so that for correct working with the specific module only one connection must be established at a time.

When tying to open the connection with the module with which the client connection has been already established (from this or other program), the connection opening function returns an error LTR_WARNING_MODULE_IN_USE.

Processing of this situation by the libraries is changed in the later versions of ltrapi in comparison with the version 1.27.

In the version 1.27 and earlier versions the connection opening function returned this error, however, executed all actions including complete reset and initial initialization of the module. It enabled to work with the module the same way as if there was no first connection, but only provided that no commands that could disturb the sequence of operation with the module, were sent over the first connection. In this situation this warning could be completely ignored and it was possible to work with the module as if opening was successful. However, the problem of this option is as follows: if the first connection is actually used and data acquisition is in progress, an attempt to re-establish the connection (possibly, from other program that knows nothing about the first connection) will lead to disturbance of data acquisition for the first connection and, moreover, does not ensure proper operation of the second connection if the first one is not closed. Particularly, starting the program that firstly finds all modules and receive on formation from them leads to disturbance of all other programs.

In the versions after the version 1.27 module reset and transmitting of any other commands to the module are not executed when opening the connection with the module if there already is the opened connection with this module. I.e. it does not disturb operation of the first connection, but the module for the second connection is in unknown

status and, besides, operations from the opening function that are specific to the modules (e.g. for the modules, where information is read from the flash-memory during connection opening, the information will not be read) turns to be not executed. Thus, in the current version of ltrapi, if the connection opening function returned an error LTR_WARNING_MODULE_IN_USE, this connection can be used only for the limited set of tasks, and in case of normal operation it is recommended to consider this warning as an error and to close the connection as in case of any other error. In this case starting of the program described in the previous example will lead to the situation when the program can not read information on the modules with which operating connections are already opened, but operation of the previously run programs will not be disturbed. And if it is necessary to open new connection reseting another connection you can use the function LTR_ResetModule() using the control connection with the service ltrd.

4.4 Acquisition of the list of connected crates and modules

The functions of ltrapi enable to obtain the list of crates with which the service ltrd established the connection, that is called as the list of active crates of the service ltrd. With that the service itself adds all found and successfully initialized crates connected via the USB interface, to this list, while connection to the crates over Ethernet is implemented using the configured entries with the IP-addresses of the crates, function for working with which are described in the section Control functions for the crates connected over Ethernet.

The list of active crates can be obtained via the control connection with the service ltrd, while to obtain the list of crate modules it is necessary to establish the control connection with the respective crate. Thus, call sequence to obtain the full list of active crates and modules can look as follows:

1. Create the instance of the TLTR structure and initialize it calling the function LTR_Init().
2. Establish the control connection with the service ltrd using the function LTR_OpenSvcControl().
3. Obtain the list of the serial numbers of all active crates using the function LTR_GetCrates() or LTR_GetCratesEx().
4. For each valid serial number of the crate obtained from the list perform the following actions:
   - Create a new instance of the TLTR structure and initialize it calling the function LTR_Init().
   - Establish the control connection with the crate, with the current serial number from the list calling LTR_OpenCrate()
   - Obtain the list of modules in the crates calling LTR_GetCrateModules()
   - Close the control connection with the crate using LTR_Close()
5. Close the control connection with the service ltrd, calling LTR_Close() for the connection descriptor, generated at step 1.

4.5 Format of IP-address assignment

IP-addresses (hereafter, IP-address means the address for the IP protocol of the version 4, that is supported by the crates and libraries) are used both for crate connection via Ethernet interface (crate IP-address), and when establishing the client connection to the service ltrd, run on the remote computer (IP-address of the computer with the service ltrd).

IP address in the textual view is four digits from 0 to 255, divided by points, e.g. “192.168.1.12”. To generate IP-addresses a 32-bit number (DWORD type) is usually used in the functions and the structures of this library (and in other libraries associated with ltrapi). Each bit of this number sets one digit from the textual entry, respectively. With that the number high-order byte corresponds to the first digit and the lower-order byte corresponds to the last digit.

Thus, for the address “a.b.c.d” 32-bit the number will be determined as (a << 24) | (b << 16) | (c << 8) | d. E.g. to set up the number 192.168.1.12 the parameter value in hexadecimal form will be equal to 0xC0A8010C.

4.6 Synchro-labels

Synchro-labels is the common mechanism of LTR crate synchronization that is described in the section "Synchronization principle of data acquisition in the LTR system” of the user manual. The common principle consists in that the special labels can be inserted to the total data flow from the crate. Due to the fact that data from all modules are sent in total flow in the order in which they entered the crate, using the position of the label for each module on the receiving side it is possible to determine between which module's counts the event occurred that corresponds to this label.

E.g. this mechanism enables to bind the following data to the accuracy up to one sampling period of the module:

- data from different modules of the same crate relative to each other
- data from different modules of different crates if crates are connected through the synchronization connector and label generation according to the "master-slaves" principle is used.
- data from different modules of one or more crates to external events if label generation from the external signal connected to the crate synchronization connector is used.

Two types of the synchro-labels are used in the LTR system - "START" label and "SECOND" label. Labels' names show their initial designation at initial generation, however, in case of generation from the external signal their designation can be random and they can only be considered as the binding to two different types of synchronization events.

Labels can be inserted to the flow either with the special modules (LTR41, LTR42, LTR43), or the crate itself, if it supports this mechanism (it is supported in the crates LTR-EU, LTR-CEU, LTR-CU). Also, in incomplete form the synchro-labels are supported by the crate LTR-U-1-4, however, label generation capabilities in this crate and accuracy of their binding to the data are limited, and if application of the synchro-labels is required in a single-place crate, it is recommended to use crates LTR-CEU-1 or LTR-CU-1.

The functions for configuring labels generated by the modules are described in programmer manuals for this modules. The functions for configuring the labels generated by the crates are implemented in ltrapi and described in this document. The exception is the crate LTR-U-1-4, for which the functions for configuring the labels are implemented in the library ltr021api and are not considered in this document.

Configuring the crate label generation is performed via the control connection with the crate using the functions LTR_MakeStartMark(), LTR_StartSecondMark() and LTR_StopSecondMark(). For label translation to the synchronization connector outputs it is also necessary pre-configure synchronization connector lines via LTR_Config().

4.6.1 Receiving and comparison of the synchro-labels with data

The service ltrd receives the synchro-labels in the data flow from the crate and counts their total number starting from the moment of crate connection by the service ltrd. When receiving the data from the module using the function LTR_Recv() the information on synchro-labels occurrence moment with their counter (in the libraries of the specific modules their own function to receive data is used, however, its operating principle and parameters are similar to the parameters of the function LTR_Recv()) is also received. As a result, the function LTR_Recv() forms a separate array tmark from 32-bit words on the basis of this information, if necessary. Each word of this array corresponds to the word received from the module with the same element number in the array and contains information on the counters of the synchro-labels at the time moment that corresponds the given module word. Every 32-bit word of tmark contains "START" label counter in the high-order half, and "SECOND" label counter in the lower-order half as shown in the table.

Bit number	Value
Bits 31-16	Quantity of "START" labels, generated prior to receiving of the respective word by the crate
Bits 15-0	Quantity of "SECOND" labels, generated prior to receiving of the respective word by the crate

Consequently, using the moment of counter change it is possible to determine between which words from the module a synchronization event has occurred. E.g. if 10 words are received from the module, and time of synchro-label generation corresponds to the time between 4th and 5th word, and N is a quantity of synchro-labels of the given type, received prior to the first word, in the first 4 elements of the array mark the counter of the synchro-labels of the given type will be indicated, that is equal to N, and in other 6 elements - N + 1.

Insert of synchro-labels to the data flow and, respectively, synchro-label binding to the module data are performed on the crate level when receiving words from the module (prior to buffering), i.e. technically, synchro-label counter change between two module words means that the synchronization event occurred after the crate had received the first word but prior to receiving of the second word. For more accurate binding of the data from different types of modules it is also necessary to consider delays caused by the module itself, e.g. if the A/D converter with the filter is used, you should consider delay of this filter. These delays are fixed for specific settings of the specific module and can be obtained, for example, experimentally for the respective case.

Also, it should be noted, that as the synchro-label counter starts from the establishment of the connection of the service ltrd with the crate and is not related to the client connections with the modules, by the time of acquisition start synchro-labels counters' values can differ from zero if labels were received from the crate prior to working with the module. Also, it should be noted that the counters' value itself for the words, that correspond to the same time, is the same for all modules of the same crate but can differ for modules from different crates. Respectively, it is necessary to consider different initial values of the counters when starting and to bind to their changes but not to absolute value.

4.6.2 Synchronization of several crates

Synchronization of these modules from several crates using synchro-labels is possible for the crates with the connector SYNC (LTR-EU, LTR-CEU, LTR-CU), or if there are modules in the crate that can generate labels. In this document only the first option is considered. When using the special modules to generate labels the principle is the same, but it uses modules' functions, that are not related to this document, for setting up.

With that, both synchronization of several crates between each other according to the principle "Master - Slaves" and synchronization using the external signal (in this case all crates are configured as "slave") are possible. For this purpose it is necessary to connect one of the DIGOUT lines of the synchronization connector of the master crate with one of the DIGIN lines of the synchronization connector of each slave crate (in case of synchronization using the external signal - connect this signal to one of the DIGIN lines of all crates to be used). If both labels are used, for each label the individual input of the master crate and the individual input of the slave crates must be used for each label. Electrical issues of this connection are not considered in this manual. The issue of correct application of the functions of this library when using this synchronization is described below.

Let's consider the option of synchronization according to the principle “Master -- Slaves” as an example of configuration. With that assume that the DIGOUT1 output of the master crate is connected with
DIGIN1 of all slave crates and used for "START" label, and the DIGOUT2 output of the master crate — with DIGIN2 of the slave crates and used for "SECOND" label.

General sequence of configuration is as follows:

1. At the initial moment synchro-label generation in all involved crates must be OFF. If it is not possible, it is necessary to switch OFF synchro-label generation for each involved crate, calling the following through the control connection with each crate:
   - LTR_MakeStartMark() with indication of the mode LTR_MARK_OFF to switch “START” label generation OFF
   - LTR_StopSecondMark() to switch “SECOND” label generation OFF
2. Configuration of the DIGOUT outputs of the master crate synchronization connector. For this purpose call the function LTR_Config() for the control connection with the master crate, having transferred the structure TLTR_CONFIG, filled in as follows:
   - in the field digout_en the value 1 is set
   - in the field digout[0] the value LTR_DIGOUT_START is set
   - in the field digout[1] the value LTR_DIGOUT_SECOND is set

3. Start of acquisition for all modules data from which should be synchronized. This item can be also executed after configuration of synchro-labels of the slave crates (item 4) it there is a guarantee that the synchronization event does not occur during acquisition start when a part of modules is started and another part is not started. In case of application of labels from the master crate it is guaranteed as labels will be generated only after calling the respective functions for the master crate.

4. Configuration of synchro-label generation using the signals at the respective DIGIN inputs of the synchronization connector for all slave crates. For this purpose for each slave crate the following should be performed via the control connection with the respective crate:

    - call the function LTR_MakeStartMark() with indication of the mode LTR_MARK_EXT_DIGIN1_RISE to set up "START" label generation start using the signal edge at the DIGIN1 input
    - call the function LTR_StartSecondMark() with indication of the mode LTR_MARK_EXT_DIGIN2_RISE to set up "SECOND" label generation using the signal edge at the DIGIN2 input

5. Start of second labels for the master crate using LTR_MakeStartMark() with indication of the mode LTR_MARK_INTERNAL via the control connection of the master crate

6. "START" labels generation by the master crate using LTR_MakeStartMark() with indication of the mode LTR_MARK_INTERNAL via the control connection of the master crate

Configuration in case of label generation using the external signal is almost the same, the difference is that there is no master crate and the respective items can be omitted.

Chapter 5

Constants, types of data and library functions

5.1 Constants and tabulations.

5.1.1 Constants and macro definitions.

Constant	Value	Description
LTRD_ADDR_LOCAL	(0x7F000001l)	IP-address for connection to the service ltrd that corresponds to the case when the service is started on the local computer (the same computer from which the connection is begin established)
LTRD_ADDR_DEFAULT	(LTRD_ADDR_LOCAL)	Default IP-address for connection to the service ltrd
LTRD_PORT_DEFAULT	(11111)	TCP-port, used by default, for connection to the service ltrd
LTR_CRATES_MAX	16l	Maximum number of crates that can be obtained using the function LTR_GetCrates(). If the number of crates can exceed the above mentioned number, you can use the function LTR_GetCratesWithInfo(), that does not have any restriction of the crate number
LTR_MODULES_PER_CRATE_MAX	16l	Maximum number of modules in one crate

LTR_CSN_SERVER_CONTROL	“#SERVER_CONTROL”	If this line is used instead of the serial number of the crate when establishing connection, the control connection with ltrd that is not related to any crate will be established
LTR_MID_MODULE	(((x) & 0xFF) | (((x) & 0xFF) « 8))	Macro for setting up the module identifier with the specified number
LTR_MODULE_NAME_SIZE	16	Size of the line with the module name in the module description
LTR_CRATE_DEVNAME_SIZE	32	Size of the line with the device name in the crate description
LTR_CRATE_SERIAL_SIZE	16	Size of the line with the serial number of the crate
LTR_CRATE_SOFTVER_SIZE	32	Size of the line with firmware version of the crate in its description
LTR_CRATE_REVISION_SIZE	16	Size of the line with crate revision in its description
LTR_CRATE_BOARD_OPTIONS_SIZE	16	Size of the line with description of the board options in the crate description
LTR_CRATE_BOOTVER_SIZE	16	Size of the line with loader version in the crate description
LTR_CRATE_CPUTYPE_SIZE	16	Size of the line with processor description in the crate description
LTR_CRATE_TYPE_NAME	16	Size of the line with crate type description
LTR_CRATE_SPECINFO_SIZE	48	Size of the additional information on the crate
LTR_CRATE_FPGA_NAME_SIZE	32	Size of the line with the description of FPGA type in the crate description

LTR_CRATE_FPGA_VERSION_SIZE	32	Size of the line with FPGA firmware version in the crate description
LTR_CRATE_THERM_MAX_CNT	8	Maximum number of thermometers in the crate reading of which are shown in the statistics
LTR_DEFAULT_SEND_RECV_TIMEOUT	10000UL	Default time-out in ms for execution of requests to ltrd

5.1.2 Error codes.

Type: en_LTR_ERRORS		
Description: Error codes that can return the ltrapi library functions. Also include general error codes for the modules.		
Constant	Value	Description
LTR_OK	0	Operation is executed without errors
LTR_ERROR_UNKNOWN	-1	Unknown error.
LTR_ERROR_PARAMETERS	-2	One of the function parameters is set up incorrectly.
LTR_ERROR_MEMORY_ALLOC	-3	Heap allocation error.
LTR_ERROR_OPEN_CHANNEL	-4	Error of initialization of the channel for exchange with ltrd
LTR_ERROR_OPEN_SOCKET	-5	Error of connection to ltrd
LTR_ERROR_CHANNEL_CLOSED	-6	Channel for exchange with ltrd is not created or is closed
LTR_ERROR_SEND	-7	Error of data transmission to ltrd
LTR_ERROR_RECV	-8	Error of data receiving from ltrd
LTR_ERROR_EXECUTE	-9	Error of exchangewith the crate controller
LTR_WARNING_MODULE_IN_USE	-10	Warning: active connection with the given module has been already estblished
LTR_ERROR_NOT_CTRL_CHANNEL	-11	This operation is only available for the control connection
LTR_ERROR_SRV_INVALID_CMD	-12	The command is not supported by ltrd

LTR_ERROR_SRV_INVALID_CMD_PARAMS	-13	ltrd does not support the specified command parameters
LTR_ERROR_INVALID_CRATE	-14	The specified crate is not found
LTR_ERROR_EMPTY_SLOT	-15	There is no module in the specified slot
LTR_ERROR_UNSUP_CMD_FOR_SRV_CTL	-16	The command is not supported by the control connection
LTR_ERROR_INVALID_IP_ENTRY	-17	Incorrect entry of the crate network address
LTR_ERROR_NOT_IMPLEMENTED	-18	This capability is not implemented
LTR_ERROR_CONNECTION_CLOSED	-19	The connection was closed by the service ltrd

LTR_ERROR_LTRD_UNKNOWN_RETCODE	-20	Unknown error code of the service ltrd
LTR_ERROR_LTRD_CMD_FAILED	-21	Error of execution of the control command of ltrd
LTR_ERROR_INVALID_CON_SLOT_NUM	-22	Incorrect slot number is specified during connection opening
LTR_ERROR_INVALID_MODULE_DESCR	-40	Incorrect module descriptor
LTR_ERROR_INVALID_MODULE_SLOT	-41	Incorrect slot for the module is specified
LTR_ERROR_INVALID_MODULE_ID	-42	Incorrect ID of the module in the response to reset
LTR_ERROR_NO_RESET_RESPONSE	-43	No response to module reset
LTR_ERROR_SEND_INSUFFICIENT_DATA	-44	Less words were transmitted to the module than were requested
LTR_ERROR_RECV_INSUFFICIENT_DATA	-45	Less words were received from the module than were requested
LTR_ERROR_NO_CMD_RESPONSE	-46	No response to the transmitted command
LTR_ERROR_INVALID_CMD_RESPONSE	-47	Incorrect response to the command was transmitted
LTR_ERROR_INVALID_RESP_PARITY	-48	Parity error in the received response to the command
LTR_ERROR_INVALID_CMD_PARITY	-49	Parity error in the transmitted command
LTR_ERROR_UNSUP_BY_FIRM_VER	-50	Capability is not supported by this firmware version

LTR_ERROR_MODULE_STARTED	-51	Operation is not possible with acquisition started
LTR_ERROR_MODULE_STOPPED	-52	Data acquisition is stopped
LTR_ERROR_RECV_OVERFLOW	-53	The buffer of the service ltrd is overfilled the receiving data
LTR_ERROR_FIRM_FILE_OPEN	-54	Error of firmware file opening
LTR_ERROR_FIRM_FILE_READ	-55	Error of firmware file reading
LTR_ERROR_FIRM_FILE_FORMAT	-56	Error of firmware file format
LTR_ERROR_FPGA_LOAD_READY_TOUT	-57	Time-out of waiting for FPGA readiness for loading is exceeded
LTR_ERROR_FPGA_LOAD_DONE_TOUT	-58	Time-out of waiting for FPGA switching to the operating mode is exceeded
LTR_ERROR_FPGA_IS_NOT_LOADED	-59	FPGA firmware is not loaded
LTR_ERROR_FLASH_INVALID_ADDR	-60	Incorrect address of flash-memory
LTR_ERROR_FLASH_WAIT_RDY_TOUT	-61	Time-out of waiting for record completing or erasing of Flash-memory is exceeded

LTR_ERROR_FIRSTFRAME_NOTFOUND	-62	Start of frame is not found in the flow from the module
LTR_ERROR_CARDSCONFIG_UNSUPPORTED	-63	Crate does not support module configuration storing
LTR_ERROR_FLASH_OP_FAILED	-64	Error of execution of the operation with flash-memory
LTR_ERROR_FLASH_NOT_PRESENT	-65	Flash-memory is not found
LTR_ERROR_FLASH_UNSUPPORTED_ID	-66	Unsupported type of flash-memory is found
LTR_ERROR_FLASH_UNALIGNED_ADDR	-67	Non-aligned address of flash-memory
LTR_ERROR_FLASH_VERIFY	-68	Error when checking data written to flash-memory
LTR_ERROR_FLASH_UNSUP_PAGE_SIZE	-69	Unsupported size of flash-memory page is set up
LTR_ERROR_FLASH_INFO_NOT_PRESENT	-70	No information on the module in flash-memory
LTR_ERROR_FLASH_INFO_UNSUP_FORMAT	-71	Unsupported format of the information on the module in flash-memory
LTR_ERROR_FLASH_SET_PROTECTION	-72	Setting up of the flash-memory protection failed
LTR_ERROR_FPGA_NO_POWER	-73	No power supply of FPGA microcircuit
LTR_ERROR_FPGA_INVALID_STATE	-74	Invalid status of FPGA loading

LTR_ERROR_FPGA_ENABLE	-75	FPGA switching to the required status failed
LTR_ERROR_FPGA_AUTOLOAD_TOUT	-76	FPGA automatic loading timeout
LTR_ERROR_PROCDATA_UNALIGNED	-77	The data to be processed are not aligned with the frame edge
LTR_ERROR_PROCDATA_CNTR	-78	Error of the counter in the data to be processed
LTR_ERROR_PROCDATA_CHNUM	-79	Incorrect number of the channel in the data to be processed
LTR_ERROR_PROCDATA_WORD_SEQ	-80	Incorrect sequence of words in the data to be processed
LTR_ERROR_FLASH_INFO_CRC	-81	Bad checksum in the recorded information on the module

5.1.3 Crate's processor client outputs connection mode

Type: en_LTR_UserIoCfg		
Description: These constants set the connection mode for the specific processor pin, that can be used in some crates when coding the client processor firmware. For the standard firmware they are not used.		
Constant	Value	Description
LTR_USERIO_DIGIN1	1	Pin is an input and connected to DIGIN1
LTR_USERIO_DIGIN2	2	Pin is an input and connected to DIGIN2
LTR_USERIO_DIGOUT	0	Foot is an output
LTR_USERIO_DEFAULT	LTR_USERIO_DIGOUT	Default value

5.1.4 Operating mode of the crate's DIGOUTx outputs.

Type: en_LTR_DigOutCfg		
Description: These values determine what signal will be set up at the specific DIGOUT output of the crate's synchronization connector.		
Constant	Value	Description
LTR_DIGOUT_CONST0	0	Constant level of logical “0”
LTR_DIGOUT_CONST1	1	Constant level of logical “1”
LTR_DIGOUT_USERIO0	2	Output is connected to the pin userio0
LTR_DIGOUT_USERIO1	3	Output is connected to the pin userio1

LTR_DIGOUT_DIGIN1	4	Output is connected to the DIGIN1 input
LTR_DIGOUT_DIGIN2	5	Output is connected to the DIGIN2 input
LTR_DIGOUT_START	6	Pulses that correspond to the times of "START" label generation are sent to the output
LTR_DIGOUT_SECOND	7	Pulses that correspond to the times of "SECOND" label generation are sent to the output
LTR_DIGOUT_IRIG	8	Check of standard time signals IRIG (digout1: ready, digout2: second)
LTR_DIGOUT_DEFAULT	LTR_DIGOUT_CONST0	Default value

5.1.5 Synchro-label generation mode.

Type: en_LTR_MarkMode		
Description: These values are used to indicate condition according to which "START" and "SECOND" labels will be generated.		
Constant	Value	Description
LTR_MARK_OFF	0	Label generation is OFF
LTR_MARK_EXT_DIGIN1_RISE	1	Label is generated by the signal edge at the input DIGIN1
LTR_MARK_EXT_DIGIN1_FALL	2	Label is generated by the signal fall at the input DIGIN1
LTR_MARK_EXT_DIGIN2_RISE	3	Label is generated by the signal edge at the input DIGIN2
LTR_MARK_EXT_DIGIN2_FALL	4	Label is generated by the signal fall at the input DIGIN2
LTR_MARK_INTERNAL	5	Internal label generation by the crate controller. For the "START" label generation is performed only once when calling LTR_MakeStartMark(). For the "SECOND" label — by the crate timer (once per second), starting from the calling LTR_StartSecondMark().
LTR_MARK_SEC_IRIGB_DIGIN1	16	The source of the label is the standard time signal decoder IRIG-B006. Can be used only foe second labels. The signal from the input DIGIN1 is used as an input signal
LTR_MARK_SEC_IRIGB_nDIGIN1	17	Similarly LTR_MARK_SEC_IRIGB_DIGIN1, but the inverted signal from the input DIGIN1 is used as an input signal
LTR_MARK_SEC_IRIGB_DIGIN2	18	Similarly

		LTR_MARK_SEC_IRIGB_DIGIN1, but the signal from the input DIGIN2 is used as an input signal
LTR_MARK_SEC_IRIGB_nDIGIN2	19	Similarly LTR_MARK_SEC_IRIGB_DIGIN1, but the inverted signal from the input DIGIN2 is used as an input signal

5.1.6 Level of history log output by the service ltrd.

Type: en_LTR_LogLevel		
Description: Level of history log output by the service ltrd.		
Constant	Value	Description
LTR_LOGLVL_ERR_FATAL	0	Fatal errors
LTR_LOGLVL_ERR	1	Errors
LTR_LOGLVL_WARN	2	Warnings
LTR_LOGLVL_INFO	3	Informational messages
LTR_LOGLVL_DETAIL	4	Details
LTR_LOGLVL_DBG_HIGH	5	Debugging messages of the higher level of concern
LTR_LOGLVL_DBG_MED	6	Debugging messages of the medium level of concern
LTR_LOGLVL_DBG_LOW	7	Debugging messages of the lower level of concern

5.1.7 Flags of the functions of acquisition of data on the connected crates.

Type: en_LTR_GetCratesFlags		
Description: These flags can control operation of the function LTR_GetCratesEx(). As the "flags" parameter the set of these flags combined by bit "OR", is transmitted to the function.		
Constant	Value	Description
LTR_GETCRATES_FLAGS_WORKMODE_ONLY	0x1	The flag indicates that the function must return only the list of crates with which the service established operating connection where crate module control is possible. As currently the crate has the only interface (configured in the settings) at a time that enables to work with the modules, with this flag several entries related to the same crate will

		not be returned in case of simultaneous crate connection via several interfaces. This flag is available starting from the version ltrd 2.1.5.0 and ltrapi 1.31.1.

5.1.8 Adjustable parameters of the service ltrd.

Type: en_LTRD_Params		
Description: List of setting parameters that control operation of the service ltrd. These parameters are configured via the configuration file or via LTR_SetServerParameter(). The description of each code specifies, using what method the parameter value is set.		
Constant	Value	Description
LTRD_PARAM_ETH_CRATE_POLL_TIME	0x100	Crate polling interval to check if the connection with crate via Ethernet interface is operating and to detect crate disconnection. DWORD type parameter sets time in ms.
LTRD_PARAM_ETH_CRATE_CON_TOUT	0x101	Timeout for establishing the service connection with the crate over Ethernet. DWORD type parameter sets time in ms.
LTRD_PARAM_ETH_CRATE_CTLCMD_TOUT	0x102	Timeout for response to control command for the crate over Ethernet. DWORD type parameter sets time in ms.
LTRD_PARAM_ETH_INTF_CHECK_TIME	0x103	Host address scan interval to start auto-connection. DWORD type parameter sets time in ms.
LTRD_PARAM_ETH_CRATE_RECONNECT_TIME	0x104	Time at the lapse of which reconnection to the crate will be performed over Ethernet in case of error, if the flag is set for this IP entry LTR_CRATE_IP_FLAG_RECONNECT. The parameter is available starting from the version ltrd 2.1.5.0 and ltrapi 1.31.1.

5.1.9 Numbers of channels for connection with the service ltrd

Type: en_LTR_CC_ChNum		
Description: Numbers of channels for connection with the service ltrd		
Constant	Value	Description
LTR_CC_CHNUM_CONTROL	0	Channel for command request transmission to the crate or the service ltrd
LTR_CC_CHNUM_MODULE1	1	Channel for operating with the module in slot 1
LTR_CC_CHNUM_MODULE2	2	Channel for operating with the module in slot 2
LTR_CC_CHNUM_MODULE3	3	Channel for operating with the module in slot 3
LTR_CC_CHNUM_MODULE4	4	Channel for operating with the module in slot 4
LTR_CC_CHNUM_MODULE5	5	Channel for operating with the module in slot 5
LTR_CC_CHNUM_MODULE6	6	Channel for operating with the module in slot 6
LTR_CC_CHNUM_MODULE7	7	Channel for operating with the module in slot 7
LTR_CC_CHNUM_MODULE8	8	Channel for operating with the module in slot 8
LTR_CC_CHNUM_MODULE9	9	Channel for operating with the module in slot 9
LTR_CC_CHNUM_MODULE10	10	Channel for operating with the module in slot 10
LTR_CC_CHNUM_MODULE11	11	Channel for operating with the module in slot 11
LTR_CC_CHNUM_MODULE12	12	Channel for operating with the module in slot 12
LTR_CC_CHNUM_MODULE13	13	Channel for operating with the module in slot 13
LTR_CC_CHNUM_MODULE14	14	Channel for operating with the module in slot 14
LTR_CC_CHNUM_MODULE15	15	Channel for operating with the module in slot 15
LTR_CC_CHNUM_MODULE16	16	Channel for operating with the module in slot 16

5.1.10 Indicators of the communication channel ltrd for definite crate interface setting up

Type: en_LTR_CC_Iface		
Description: Indicators of the communication channel ltrd for definite crate interface setting up		
Constant	Value	Description
LTR_CC_IFACE_USB	0x0100	Distinct indication that the connection must be established with the crate connected via USB-interface
LTR_CC_IFACE_ETH	0x0200	Distinct indication that the connection must be established with the crate connected over Ethernet (TCP/IP)

5.1.11 Additional flags of the channel for communication with ltrd

Type: en_LTR_CC_Flags		
Description: Additional flags of the channel for communication with ltrd		
Constant	Value	Description

5.1.12 Connection status flags

Type: en_LTR_ChStateFlags		
Description: Connection status flags		
Constant	Value	Description
LTR_FLAG_RBUF_OVF	(1u«0)	Flag of buffer over-flow Indicates that because the client has not read data, client queue overflow has occurred in ltrd, therefore, there is a gap in received data
LTR_FLAG_RFULL_DATA	(1u«1)	Flag of data receiving in the full format in the function LTR_GetCrateRawData()

5.1.13 Modules' identifiers

Type: en_LTR_MIDs		
Description: Modules' identifiers		
Constant	Value	Description
LTR_MID_EMPTY	0	Empty slot
LTR_MID_IDENTIFYING	0xFFFF	Module is in the process of type definition
LTR_MID_LTR01	LTR_MID_MODULE(1)	Identifier of the module LTR01
LTR_MID_LTR11	LTR_MID_MODULE(11)	Identifier of the module LTR11

LTR_MID_LTR22	LTR_MID_MODULE(22)	Identifier of the module LTR22
LTR_MID_LTR24	LTR_MID_MODULE(24)	Identifier of the module LTR24
LTR_MID_LTR25	LTR_MID_MODULE(25)	Identifier of the module LTR25
LTR_MID_LTR27	LTR_MID_MODULE(27)	Identifier of the module LTR27
LTR_MID_LTR34	LTR_MID_MODULE(34)	Identifier of the module LTR34
LTR_MID_LTR35	LTR_MID_MODULE(35)	Identifier of the module LTR35
LTR_MID_LTR41	LTR_MID_MODULE(41)	Identifier of the module LTR41
LTR_MID_LTR42	LTR_MID_MODULE(42)	Identifier of the module LTR42
LTR_MID_LTR43	LTR_MID_MODULE(43)	Identifier of the module LTR43
LTR_MID_LTR51	LTR_MID_MODULE(51)	Identifier of the module LTR51
LTR_MID_LTR114	LTR_MID_MODULE(114)	Identifier of the module LTR114
LTR_MID_LTR210	LTR_MID_MODULE(210)	Identifier of the module LTR210
LTR_MID_LTR212	LTR_MID_MODULE(212)	Identifier of the module LTR212

5.1.14 Crate types

Type: en_LTR_CrateTypes		
Description: Crate types		
Constant	Value	Description
LTR_CRATE_TYPE_UNKNOWN	0	Unknown crate type
LTR_CRATE_TYPE_LTR010	10	Crate LTR-U-8 or LTR-U-16
LTR_CRATE_TYPE_LTR021	21	Crate LTR-U-1
LTR_CRATE_TYPE_LTR030	30	Crate LTR-EU-8 or LTR-EU-16
LTR_CRATE_TYPE_LTR031	31	Crate LTR-EU-2
LTR_CRATE_TYPE_LTR_CU_1	40	Crate LTR-CU-1
LTR_CRATE_TYPE_LTR_CEU_1	41	Crate LTR-CEU-1
LTR_CRATE_TYPE_BOOTLOADER	99	Crate is in the loader mode (if the type can not be determined in this mode)

5.1.15 Crate connection interface

Type: en_LTR_CrateIface		
Description: Crate connection interface		
Constant	Value	Description
LTR_CRATE_IFACE_UNKNOWN	0	Unknown code of the crate interface. When transmitting to the functions this value can indicate that crate connection interface is of no importance
LTR_CRATE_IFACE_USB	1	Crate is connected via USB interface
LTR_CRATE_IFACE_TCPIP	2	Crate is connected over Ethernet (TCP/IP)

5.1.16 Status of connection with the crate that corresponds to the entry with IP-address

Type: en_LTR_CrateIpStatus		
Description: Status of connection with the crate that corresponds to the entry with IP-address		
Constant	Value	Description
LTR_CRATE_IP_STATUS_OFFLINE	0	Crate is not connected
LTR_CRATE_IP_STATUS_CONNECTING	1	Connection with the crate is in progress (operation is started but is not completed)
LTR_CRATE_IP_STATUS_ONLINE	2	Crate is connected
LTR_CRATE_IP_STATUS_ERROR	3	Error of the crate connection. Fail to establish the connection.

5.1.17 Flags corresponding to the entry with crate's IP-address

Type: en_LTR_CrateIpFlags		
Description: Flags corresponding to the entry with crate's IP-address		
Constant	Value	Description
LTR_CRATE_IP_FLAG_AUTOCONNECT	0x1	The flag indicates that when starting or in case of detection of a new network the service ltrd must establish connection to the crate with IP-address that corresponds to this entry

LTR_CRATE_IP_FLAG_RECONNECT	0x2	The flag indicates that in case of an error of establishment of connection with the crate or disconnection from the operating crate due to an error the service ltrd must perform new attempt of connection. If the flag is not set up, the entry passes to the status LTR_CRATE_IP_STATUS_ERROR and no actions are executed. If the flag is set up, in a set time interval a new attempt of connection will be performed and these attempts will be repeatedly performed until the connection is established or a distinct disconnection command is executed, this flag is deleted or the respective entry with the IP-address is deleted. This capability is available starting from the version ltrd 2.1.5.0 and ltrapi 1.31.1.

5.1.18 Flags from the module description

Type: en_LTR_ModuleDescrFlags		
Description: Flags from the module description		
Constant	Value	Description
LTR_MODULE_FLAGS_HIGH_BAUD	0x0001	The indicator showing that the module uses high-speed of the interface for word transmission to the module
LTR_MODULE_FLAGS_USE_HARD_SEND_FIFO	0x0100	The indicator showing that the module uses the statistics of the internal hardware FIFO for data transmission
LTR_MODULE_FLAGS_USE_SYNC_MARK	0x0200	The indicator showing that the module supports synchro-label generation

5.1.19 Crate operation mode

Type: en_LTR_CrateMode		
Description: Crate operation mode		
Constant	Value	Description
LTR_CRATE_MODE_BOOTLOADER	1	Crate is in the loader mode
LTR_CRATE_MODE_WORK	2	Crate is in the operating mode
LTR_CRATE_MODE_CONTROL	3	Crate is in the mode when it receives only command requests (e.g. if the crate is not connected via the interface to which it is set)

5.1.20 FPGA status

Type: e_LTR_FPGA_STATE		
Description: Constants that determine the current FPGA status. These constants are used in the functions and added to ltrapi as they are common for the module group but they do not use the function of this library distinctly		
Constant	Value	Description
LTR_FPGA_STATE_NO_POWER	0x0	No FPGA power supply signal
LTR_FPGA_STATE_NSTATUS_TOUT	0x1	Timeout of FPGA readiness for loading
LTR_FPGA_STATE_CONF_DONE_TOUT	0x2	Timeout of FPGA loading completion (usually means the there is no valid firmware in Flash)
LTR_FPGA_STATE_LOAD_PROGRESS	0x3	FPGA is being loaded
LTR_FPGA_STATE_POWER_ON	0x4	State after POWER_ON. Indicate that unexpected power supply failure has occurred
LTR_FPGA_STATE_LOAD_DONE	0x7	FPGA lading is completed, but FPGA operation is still disabled
LTR_FPGA_STATE_WORK	0xF	Normal operating FPGA status

5.2 Data types

5.2.1 Connection descriptor.

Type: TLTR		
Description: This structure contains all information on the connection with the service ltrd. This structure is used in majority of functions as the first parameter. A part of fields (saddr, sport, csn, cc) are intended for filling-in by the user prior to connection establishment to set up connection parameters and then is not changed. A part of parameters (flags, tmark) is intended only for reading and shows additional information that is updated during receiving data through LTR_Recv() or in other cases.		
Field	Type	Field description
saddr	DWORD	IP-address of the computer on which the service ltrd has been run, in 32-bit format. By default it is set up in LTRD_ADDR_DEFAULT equal to LTRD_ADDR_LOCAL that corresponds to the case when the service has been run on the same computer with the user program run.
sport	WORD	Number of the TCP port that will be used when connecting to the service ltrd. LTRD_PORT_DEFAULT is used by default.
csn	CHAR [LTR_CRATE_ SERIAL_SIZE]	Serial number of the crate to which the connection must respond. If an empty line (by default) is set, connection with the first crate from the current list of active crates of ltrd will be established. In this case after connection establishment this field will be changed to the actual serial number of the crate with which the connection is established. If the line LTR_CSN_SERVER_CONTROL is written, the control connection with the ltrd service will be established that is not related to any crate and can be established when no active crates are available.

cc	WORD	Type of the channel for connection with ltrd. It is configured by one of the constants en_LTR_CC_ChNum. Indicates whether this connection is control (LTR_CC_CHNUM_CONTROL) or it is the connection with the module in the specified slot (LTR_CC_CHNUM_MODULE1 .. LTR_CC_CHNUM_MODULE16). Also, if necessary, can be combined with the flags en_LTR_CC_Iface and en_LTR_CC_Flags to indicate additional connection parameters.
flags	DWORD	Connection status indicators Set of values from en_LTR_ChStateFlags combined through the logical OR. This field is intended only for reading and must not be distinctly changed by the user.
tmark	DWORD	The last value of synchro-labels received for this connection. This field is updated when executing LTR_Recv(), if synchro-labels were detected during reception. This field is intended only for reading by the reader and must not be distinctly changed by the user.
Internal	LPVOID	Opaque pointer to the structures with parameters required to ensure exchange. The user is prohibited to use this field.

5.2.2 Configuration of the synchronization connector lines.

Type: TLTR_CONFIG		
Description: This structure is used to configure functions of the outputs DIGOUT on the synchronization connector SYNC of the crates LTR-EU, LTR-CEU and LTRCU. Also, this structure enables to configure the functions of the special user pins of the processor BlackFin for the crates LTR-EU, however, the last function is required only when coding user firmware.		
Field	Type	Field description
userio	WORD [4]	Configuration of processor user pin functions. Each element corresponds to its own pin userio0 - userio3 and sets its mode by one of values en_LTR_UserIoCfg. (for LTR-EU userio0 — PF1 (board revision 0) or PF0 (in revision 1+), userio1 — PG13, userio2 — PF3 (only revision 1+), userio3 — reserve). In the standard crate firmware these settings do not influence operation and must be set to the value LTR_USERIO_DEFAULT.

digout	WORD [2]	Configuration of the functions of the outputs DIGOUT on the crate's synchronization connector. Each array element sets the mode of the respective output (element 0 — DIGOUT1, and 1 — DIGOUT2) by one of the values from en_LTR_DigOutCfg. Besides, to use this functions the outputs must be enabled via digout_en.
digout_en	WORD	Enabling of the outputs DIGOUT1 and DIGOUT2 on the crate's synchronization connector (0 — disabled, 1 — enabled). All output lines are enabled or disabled simultaneously. If the outputs are enabled the signal will be generated at them in accordance with the configured function indicated in the respective element of the array digout. If disabled, the outputs are in the third status regardless of the set configured functions in the array digout.

5.2.3 Information on the type and interface of crate connection

Type: TLTR_CRATE_INFO		
Description: This structure is filled in with the function LTR_GetCrateInfo() and contains the information on the crate type and its connection interface.		
Field	Type	Field description
CrateType	BYTE	Crate type — value from en_LTR_CrateTypes
CrateInterface	BYTE	Crate connection interface — value from en_LTR_CrateIface

5.2.4 Entry with the crate IP-address

Type: TLTR_CRATE_IP_ENTRY		
Description: This structure contains the information on the log of IP-address of the crate, stored in the settings of the service ltrd, and the status of the crate connection via Ethernet interface (TCP/IP), that corresponds to this entry		
Field	Type	Field description
ip_addr	DWORD	Crate IP-address. Format is similar to the field saddr in TLTR
flags	DWORD	Set of flags related to this entry from en_LTR_CrateIpFlags
serial_number	CHAR [LTR_CRATE_ SERIAL_SIZE]	If the crate is connected, this field contains the serial number of the connected crate. This number can be used to open the connection with the crate. For other status values this field contains the empty line as the serial number is unknown

is_dynamic	BYTE	Reserve field. Always equal to 0
status	BYTE	Crate connection status corresponding to this entry. Single value from en_LTR_CrateIpStatus

5.2.5 Crate statistics

Type: TLTR_CRATE_STATISTIC		
Description: The structure contains information on the crate status and the statistics parameters of operation with the crate that is gathered by the service ltrd. This statistics can be obtained via the control connection using the function LTR_GetCrateStatistic(). The statistics is gathered from the moment of connection establishment between ltrd and the crate.		
Field	Type	Field description
size	DWORD	Size of all valid structure fields, including the "size" field itself
flags	DWORD	Flags — reserve
crate_type	WORD	Crate type from en_LTR_CrateTypes
crate_intf	WORD	Interface via which the crate from en_LTR_CrateIface is connected
crate_state	WORD	Reserve
crate_mode	WORD	Operating mode of the crate from en_LTR_CrateMode
con_time	ULONGLONG	Set-up time for the connection between service and the crate (format unixtime)
res	WORD [11]	Reserve

modules_cnt	WORD	Number of slots in this crate type
mids	WORD [LTR_MODULES_ PER_CRATE_MAX]	ID of the modules for all crate slots
res2	WORD [3 *LTR_ MODULES_PER_CRATE_ MAX]	Reserve
ctl_clients_cnt	WORD	Number of clients connected via the control channel to the crate
total_mod_clients_ cnt	WORD	Number of clients connected to all crate modules
res3	DWORD [11]	Reserve

wrd_sent	ULONGLONG	Total number of words transmitted to the crate (crate and all its modules)
wrd_recv	ULONGLONG	Total number of words received from the crate (from crate itself and all its modules)
bw_send	double	Current rate of word transmission to the crate (word/s)
bw_recv	double	Current frequency of word receive from the crate (word/s)
crate_wrd_recv	ULONGLONG	Number of words received directly from the crate
internal_rbuf_miss	ULONGLONG	Number of lost buffers in the crate due to internal overflow
internal_rbuf_ovfls	DWORD	Number of overflows of the crate internal buffer
rbuf_ovfls	DWORD	Number of overflows of the buffer for data receiving from the modules in the service ltrd for the crate modules (total number for all modules)
total_start_marks	DWORD	Number of the "Start" labels received both from the crate and the modules
total_sec_marks	DWORD	Number of the second labels received both from the crate and the modules
crate_start_marks	DWORD	Number of "Start" labels received directly from the crate
crate_sec_marks	DWORD	Number of second labels received directly from the crate
crate_unixtime	ULONGLONG	The last value of the extended second label (unixtime format), if supported by the crate
therm_mask	DWORD	Mask of valid thermometer readings (if not supported — 0)
therm_vals	float [LTR_CRATE_ THERM_MAX_CNT]	crate thermometer readings value. Valid only if the respective bit in herm_mask is in 1
res4	DWORD [19]	Reserve

5.2.6 Module statistics

Type: TLTR_MODULE_STATISTIC		
Description: The structure contains information on the module status and the statistics parameters of operation with this module that is gathered by the service ltrd. This statistics can be obtained via the control connection using the function LTR_GetModuleStatistic(). The statistics is gathered from the moment when the module is found and reset along with module resetting via LTR_ResetModule()		
Field	Type	Field description
size	DWORD	Size of all valid structure fields, including the "size" field itself
client_cnt	WORD	Number of clients established the connection with the module
mid	WORD	Identifier of the module from en_LTR_MIDs
flags	DWORD	Set of flags describing module features from en_LTR_ModuleDescrFlags
name	CHAR [LTR_MODULE_ NAME_SIZE]	Line with the name of the module (possibly with modifications, if ltrd can determine them)
res	DWORD [5]	Reserve
wrd_sent	ULONGLONG	Number of words, transmitted to the module
wrd_recv	ULONGLONG	Number of words received from the module
bw_send	double	Current rate of word transmission to the module (word/s)
bw_recv	double	Current frequency of word receive from the module (word/s)
wrd_sent_to_client	ULONGLONG	Number of words, transmitted to the client
wrd_recv_from_ client	ULONGLONG	Number of words received from the client
wrd_recv_drop	ULONGLONG	Number of omitted words due to overflow of the buffer for receiving in the service ltrd
rbuf_ovfls	DWORD	Number of overflows of the buffer for receiving in the service ltrd
send_srvbuf_size	DWORD	Size of the buffer in ltrd for the module for transmission
recv_srvbuf_size	DWORD	Size of the buffer in ltrd for the module for receiving

send_srvbuf_full	DWORD	By how many words the buffer for transmission is filled
recv_srvbuf_full	DWORD	By how many words the buffer for receiving is filled
send_srvbuf_full_ max	DWORD	By how many words the buffer for transmission was max. filled
recv_srvbuf_full_ max	DWORD	By how many words the buffer for receiving was max. filled
res2	DWORD [17]	Reserve
start_mark	DWORD	Number of "START" labels received from the module
sec_mark	DWORD	Number of second labels received from the module
hard_send_fifo_size	DWORD	Size of hardware queue inside the module. This field and all other fields hard_send_... are valid only for the output modules with available controlled sequential queue ltrd in the module (the respective flag for these modules is set up in the "flags" field)
hard_send_fifo_ unack_words	DWORD	Allocated status of the hardware queue (number of transmitted but not confirmed words)
hard_send_fifo_ underrun	DWORD	Number of queue "hungers" (the queue is empty when trying to output the value from it) from the moment of the last module reset
hard_send_fifo_ overrun	DWORD	Number of queue overflows from the moment of the last module reset
hard_send_fifo_ internal	DWORD	Internal status of the hardware queue
res3	DWORD [25]	Reserve

5.2.7 Information on the crate and its firmware

Type: TLTR_CRATE_DESCR		
Description: The structure contains the information on the crate including all versions related to the crate. Majority of the fields are presented in the form that ends with the line null symbol. This statistics can be obtained via the control connection using LTR_GetCrateDescr().		
Field	Type	Field description
size	DWORD	Size of all valid structure fields, including the "size" field itself
devname	CHAR [LTR_CRATE_DEVNAME_SIZE]	Crate name
serial	CHAR [LTR_CRATE_SERIAL_SIZE]	Serial number
soft_ver	CHAR [LTR_CRATE_SOFTVER_SIZE]	Firmware version
brd_revision	CHAR [LTR_CRATE_REVISION_SIZE]	Board revision
brd_opts	char [LTR_CRATE_BOARD_OPTIONS_SIZE]	Board options
bootloader_ver	CHAR [LTR_CRATE_BOOTVER_SIZE]	Loader version
cpu_type	CHAR [LTR_CRATE_CPUTYPE_SIZE]	Type of microcontroller
fpga_name	CHAR [LTR_CRATE_FPGA_NAME_SIZE]	FPGA name in the crate
fpga_version	char [LTR_CRATE_FPGA_VERSION_SIZE]	FPGA firmware version
crate_type_name	CHAR [LTR_CRATE_TYPE_NAME]	Line with crate type
spec_info	CHAR [LTR_CRATE_SPECINFO_SIZE]	Reserve
protocol_ver_major	BYTE	Version of the protocol between ltrd and the crate (major)
protocol_ver_minor	BYTE	Version of the protocol between ltrd and the crate (minor)

5.3 Function

5.3.1 Functions of initialization and working with connection

5.3.1.1 Initialization of the connection descriptor

Format: INT LTR_Init (TLTR *hnd)
Description: The function initializes the structure fields of the connection descriptor using default values. This function must be called first for every structure TLTR prior to calling other functions.
Parameters: hnd — Connection descriptor.
Returned value: Error code.

5.3.1.2 Connection opening

Format: INT LTR_Open (TLTR *hnd)
Description: The function establishes the client connection in accordance with the set up fields saddr, sport, csn and cc of the module descriptor. Chapter Client connection types contain detailed information on connection types. Upon completion of work it is necessary to close connection using LTR_Close(). If the function returns an error, in some cases the connection can remain opened (e.g. fr the error LTR_WARNING_MODULE_IN_USE), therefore even if this function returns an error, you should call LTR_Close(). It is recommended to interpret all returned errors (including LTR_WARNING_MODULE_IN_USE) as an indicator showing that it is not possible to work with the connection, and the only allowed function that can and must be called, is LTR_Close().
Parameters: hnd — Connection descriptor.
Returned value: Error code.

5.3.1.3 Opening of the control connection with the service ltrd

Format: INT LTR_OpenSvcControl (TLTR *hsrv, DWORD ltrd_addr, WORD ltrd_port)
Description: This function establishes the control connection with the service ltrd. This connection can be established even when there are no connected crates and it enables to execute commands that control service operation (except for the commands for crate control, that require to open the control connection with the specific crate). Upon completion of work with the connection, it is necessary to close it calling LTR_Close(). Function action is similar to correct filling in the fields of the structure TLTR using the line LTR_CSN_SERVER_CONTROL as the serial number and to calling LTR_Open() and serves for convenience in order not to fill in manually. Also can be executed and having filled in the fields of the structure TLTR and having called LTR_Open(). This function is introduced for convenience in order not to fill the fields in manually. The function is available in ltrapi of version 1.31.0 or later versions.
Parameters: hsrv — Control connection descriptor. ltrd_addr — IP-address of the computer on which the service ltrd has been run, in 32-bit format. To use default value the value LTRD_ADDR_DEFAULT can be transmitted. ltrd_port — Number of the TCP port that will be used when connecting to the service ltrd. To use the default port the value LTRD_PORT_DEFAULT can be transmitted.
Returned value: Error code.

5.3.1.4 Opening of the control connection with the crate

Format: INT LTR_OpenCrate (TLTR *hcrate, DWORD ltrd_addr, WORD ltrd_port, INT crate_iface, const char *crate_sn)
Description: This function establishes the control connection with the crate by its serial number (or with the first crate if serial number is not set). The function enables to distinctly indicate the interface via which the connection between the service ltrd and the crate is established, if the crate is connected via two interfaces simultaneously (e.g. if the crate is configured and connected over Ethernet, but also connected over USB in the set-up mode). Generally, it is sufficient to indicate LTR_CRATE_IFACE_UNKNOWN as an interface, that means that the crate can be connected via any interface. With that if the crate is connected via two interfaces the opened connection will be related to the crate using the interface that is the operating one for the crate,

i.e. via the interface that can exchange data with the crate modules that is necessary for majority of programs. Upon completion of work with the connection, it is necessary to close it calling LTR_Close(). Function action is similar to correct filling in the fields of the structure TLTR and to calling LTR_Open(). This function is introduced for convenience in order not to fill the fields in manually. The function is available in ltrapi of version 1.31.0 or later versions.
Parameters: hcrate — Connection descriptor. ltrd_addr — IP-address of the computer on which the service ltrd has been run, in 32-bit format. To use default value the value LTRD_ADDR_DEFAULT can be transmitted. ltrd_port — Number of the TCP port that will be used when connecting to the service ltrd. To use the default port the value LTRD_PORT_DEFAULT can be transmitted. crate_iface — Value from en_LTR_CrateIface, indicating the interface via which the crate must be connected. If the value LTR_CRATE_IFACE_UNKNOWN is set, the crate can be connected via any interface. crate_sn — Line with the serial number of the crate with which it is ncessary to establish connection. If the line is empty, connection will be established with the first connected crate
Returned value: Error code.

5.3.1.5 Opening of the connection with the set time-out

Format: INT LTR_OpenEx (TLTR *hnd, DWORD timeout)
Description: This function is the same as LTR_Open(), but enables to distinctly indicate the maximum set-up time of connection with ltrd.
Parameters: hnd — Connection descriptor. timeout —Time in ms for connection opening. If during the pre-set time connection with ltrd is not established, function will be finished with an error.
Returned value: Error code.

5.3.1.6 Closing of connection

Format: INT LTR_Close (TLTR *hnd)
Description: The function closes the previously opened connection using the functions LTR_Open(), LTR_OpenEx(), LTR_OpenCrate() or LTR_OpenSvcControl() . With any returned value after calling this function the respective descriptor of the connection can not be already used without opening a new connection.
Parameters: hnd — Connection descriptor.
Returned value: Error code.

5.3.1.7 Check if the connection is opened

Format: INT LTR_IsOpened (TLTR *hnd)
Description: The function checks whether the connection that corresponds to the specified descriptor, is currently opened by the client. If the connection is opened the function returns LTR_OK, if it is closed — error code LTR_ERROR_CHANNEL_CLOSED. This function does not check by any methods whether this connection is currently valid, and its result is determined only by the function calling sequence - whether the connection with the client was successfully opened and whether this connection was closed.
Parameters: hnd — Connection descriptor.
Returned value: Error code (LTR_OK, if the connection is established).

5.3.2 Information type functions

Set of functions that enable to obtain general information on the service ltrd, connected crates and exchange statistics. These functions can be used by any control connection.

5.3.2.1 Acquisition of the ltrd service version.

Format: INT LTR_GetServerVersion (TLTR *hsrv, DWORD *version)
Description: The function returns version number of the service ltrd, with which the control connection is established. In textual view the version of ltrd consists of four numbers separated by points. The value returned by this function is a 32-bit value, each byte of which corresponds to one part of the number of the version, separated by a point in the textual entry, with that the higher-order byte corresponds to the main number (the first number). E.g. 0x02010403 corresponds to the version 2.1.4.3. Versions of the service ltrd start with 2.0.0.0 and higher, while the versions with the higher-order number 1 were returned only by the previously used LTR program. Server.
Parameters: hsrv — Control connection descriptor. version — In this variable the version of the service ltrd is returned in the format described above.
Returned value: Error code.

5.3.2.2 Acquisition of serial numvers of the connected crates

Format: INT LTR_GetCrates (TLTR *hsrv, BYTE *csn)
Description: The function returns the list of crate serial numbers with which the service ltrd established connection (list of active crates). Obtained serial numbers can be used to establish the control connection with the crates (for their configuration, obtaining the module list, etc.) or connection with the modules. This function can return max. LTR_CRATES_MAX of serial numbers (if more crates are connected, only first LTR_CRATES_MAX will be returned). If number of crates may exceed LTR_CRATES_MAX, you can use the function LTR_GetCratesEx(), where there is no limitation of the number of crates to be returned. The function always returns different serial numbers, i.e. if the crate is connected both over USB (in the set-up mode) and over Ethernet (in the operating mode) the function will fill in only one element in the output array.

Parameters: hsrv — Control connection descriptor. csn —2D array with size [LTR_CRATES_MAX] [LTR_CRATE_SERIAL_SIZE] bytes. Serial numbers of the connected crates will be saved in this array (each line corresponds to its own crate serial number). All LTR_CRATES_MAX of serial numbers are always filled in. If the number of crates is less than LTR_CRATES_MAX, instead of serial numbers after the last valid serial number the empty line will be written (one symbol with null code)ю
Returned value: Error code

5.3.2.3 Acquisition of the information on the connected crates

Format: INT LTR_GetCratesEx (TLTR *hsrv, DWORD max_crates, DWORD flags, DWORD *crates_found, DWORD *crates_returned, CHAR serials[] [LTR_CRATE_SERIAL_SIZE], TLTR_CRATE_INFO *info_list)
Description: This function enables to obtain the list of crate serial numbers with which the service ltrd established connection (list of active crates) with additional information on these crates. As opposed to LTR_GetCrates() the function does not limit the number of the connected crates, information on which can be returned. Also, as opposed to LTR_GetCrates(), if one crate is connected via several interfaces, this function will return two entries about the crate, i.e. its serial number will be indicated in the "serials" array twice, but with that in the respective entries with the information on the crate different connection interface will be indicated (can be changed with flags). The function can be used with zero value of max_crates to obtain the number of connected crates in order to create arrays of the required sizes and to call the function for the second time to obtain information on all crates. This function is only supported by the service ltrd. The function is available in ltrapi of version 1.31.0 or later versions.

Parameters:
hsrv — Control connection descriptor. max_crates — Maximum number of crates, information on which can be returned by the function.
flags — Flags from en_LTR_GetCratesFlags, controlling function operation.
crates_found — In this parameter the total number of connected crates is returned. This value can exceed max_crates. If this function is not required the null indicator can be sent. crates_returned — In this variable the number of valid serial numbers, recorded in the array serials, is returned. This returned value can not exceed the value transmitted in the parameter max_crates. If max_crates is equal to zero, the null indicator can be transmitted as this parameter.
serials — Array to store serial numbers of the connected crates. It must have sufficient size to store max_crates of serial numbers (each serial number is the line of LTR_CRATE_SERIAL_SIZE symbols). Upon function completion the first crates_returned of serial numbers will be filled in. If max_crates is equal to zero, the null indicator can be transmitted as this parameter.
info_list — Array to store information on the connected crates. It must have sufficient size to store max_crates of the structures TLTR_CRATE_INFO. Each element corresponds to the serial number from serials with the same number. If the information on crates is not required, the null indicator can be transmitted.

Returned value: Error code.

5.3.2.4 Acquisition of the crate description

Format: INT LTR_GetCrateDescr (TLTR *hsrv, INT crate_iface, const char *crate_sn, TLTR_CRATE_DESCR *descr, DWORD size)

Description:
The function enables to obtain the structure of TLTR_CRATE_DESCR type with detailed description of the specified crate. Description of any connected crate can be obtained via single control connection — it is not necessary to establish individual control connection with the crate.
This function is only supported by the service ltrd (not supported by the previously used LTR Server program).

Parameters:
hsrv — Control connection descriptor. crate_iface — Crate connection interface (similar to the so-named parameter LTR_OpenCrate()).
crate_sn — Crate serial number (similar to the so-named parameter LTR_OpenCrate()).
descr — The structure where crate description will be saved size —
Size of the structure transmitted as the parameter descr.

Returned value: Error code.

5.3.2.5 Acquisition of the statistics on the crate

Format: INT LTR_GetCrateStatistic (TLTR *hsrv, INT crate_iface, const char *crate_sn, TLTR_CRATE_STATISTIC *stat, DWORD size)
Description: The function returns additional parameters of the statistics, that is gathered by the service ltrd, related to the specified crate, as the structure TLTR_CRATE_STATISTIC. This function is only supported by the service ltrd (not supported by the previously used LTR Server program).
Parameters: hsrv — Control connection descriptor. crate_iface — Crate connection interface (similar to the so-named parameter LTR_OpenCrate()). crate_sn — Crate serial number (similar to the so-named parameter LTR_OpenCrate()). stat — The structure where crate statistics parameters will be saved. size — Size of the structure transmitted as the parameter stat.
Returned value: Error code.

5.3.2.6 Acquisition of the statistics on the module

Format: INT LTR_GetModuleStatistic (TLTR *hsrv, INT crate_iface, const char *crate_sn, INT module_slot, TLTR_MODULE_STATISTIC *stat, DWORD size)
Description: The function returns additional parameters of the statistics, that is gathered by the service ltrd, related to the specified module, as the structure TLTR_MODULE_STATISTIC. This function is only supported by the service ltrd (not supported by the previously used LTR Server program).
Parameters: hsrv — Control connection descriptor. crate_iface — Crate connection interface (similar to the so-named parameter LTR_OpenCrate()). crate_sn — Crate serial number (similar to the so-named parameter LTR_OpenCrate()). module_slot —Number of themodule slot (from LTR_CC_CHNUM_MODULE1 to LTR_CC_CHNUM_MODULE16). stat — The structure where crate statistics parameters will be saved. size — Size of the structure transmitted as the parameter stat.
Returned value: Error code.

5.3.3 Crates control functions

Set of functions that implement common commands to control LTR crates. These functions can be used only with the control connection with the crate.

5.3.3.1 Acquisition of the list of modules in the crate

Format: INT LTR_GetCrateModules (TLTR *hcrate, WORD *mid)
Description: The function enables to obtain the list of modules' identifiers that are set in the crate. It is necessary to send the array of LTR_MODULES_PER_CRATE_MAX elements, that will be filled in with the required values by the function, to the function input. Each element corresponds to its slot in the crate (mid[0] — identifier of the module, inserted to the first slot, mid[15] — in the 16th slot) and in case of successful function execution it is set on one of the values from en_LTR_MIDs. If no module is inserted to the slot or if there is no slot, the value LTR_MID_EMPTY will be set. If the module is found but its type is still not determined, the value LTR_MID_IDENTIFYING will be returned.
Parameters: hcrate — Descriptor of the control connection with the crate. mid — Pointer to the array of LTR_MODULES_PER_CRATE_MAX elements where identifiers of the installed modules will be returned in case of successful execution.
Returned value: Error code.

5.3.3.2 Obtaining of the information on the type and interface of crate connection

Format: INT LTR_GetCrateInfo (TLTR *hcrate, TLTR_CRATE_INFO *CrateInfo)
Description: The function fills in the structure TLTR_CRATE_INFO with the information on the crate with which the control connection is established.
Parameters: hcrate — Descriptor of the control connection with the crate. CrateInfo — In this structure the information on the crate is returned in case of success.
Returned value: Error code.

5.3.3.3 Configuration of the crate synchronization connector lines

Format: INT LTR_Config (TLTR *hcrate, const TLTR_CONFIG *conf)
Description: The function sets up the configuration of the SYNC synchronization connector lines according to the parameters set in the structure TLTR_CONFIG. This function is only applicable for the crates with this connector available (LTR-EU, LTR-CU, LTR-CEU).
Parameters: hcrate — Descriptor of the control connection with the crate. conf — Configuration of the crate synchronization connector lines.
Returned value: Error code.

5.3.3.4 Setting of the “START” label generation

Format: INT LTR_MakeStartMark (TLTR *hcrate, INT mode)
Description: The function sets up the mode of "START" label generation by the crate. This function only operates with the crates that support the standard synchro-label generation mechanism (LTR-EU, LTR-CEU, LTR-CU). The crate can generate a label both from external event and internal event — on command from a PC. Function behavior slightly varies depending on the mode value. If the mode LTR_MARK_INTERNAL is set up, when executing this function the crate generates a single "START" label, then does not generate labels until the next call of this function. I.e. in case of internal generation of "START" label it is necessary to call this function twice, when the label must be generated. Other modes corresponds to the external label generation. In this modes the function just configures the crate so that it waits for the event set up in the mode and generates a label in case of each this event. To disable external label generation you can call this function indicating the mode LTR_MARK_OFF. If connection with the crate is closed without disabling label generation, the crate will still generate labels until distinct disabling.
Parameters: hcrate — Descriptor of the control connection with the crate. mode — "START" label generation mode — value from en_LTR_MarkMode.
Returned value: Error code.

5.3.3.5 Start of “SECOND” label generation

Format: INT LTR_StartSecondMark (TLTR *hcrate, INT mode)
Description: The function starts second label generation in the specified mode. This function only operates with the crates that support the standard synchro-label generation mechanism (LTR-EU, LTR-CEU, LTR-CU). If the mode LTR_MARK_INTERNALis set up, once this function is called the crate starts to generate a second label once per second (from the internal timer). In other modes the crate waits for the external event and generates a second label in case of each event. Generation is stopped with the function LTR_StopSecondMark(). If connection with the crate is closed without disabling second label generation, the crate will still generate labels until distinct disabling.
Parameters: hcrate — Descriptor of the control connection with the crate. mode — "SECOND" label generation mode.
Returned value: Error code.

5.3.3.6 Stop of “SECOND” label generation.

Format: INT LTR_StopSecondMark (TLTR *hcrate)
Description: When calling this function the crate stops generation of second labels that was previously started with LTR_StartSecondMark()
Parameters: hcrate — Descriptor of the control connection with the crate.
Returned value: Error code.

5.3.4 Functions of ltrd service control

Set of additional functions of the control connection related to control of operation of the service ltrd.

5.3.4.1 Reset of the specified module

Format: INT LTR_ResetModule (TLTR *hsrv, INT crate_iface, const char *crate_sn, INT module_slot, DWORD flags)
Description: This function enables to reset any module via the control connection. Upon this function execution the following actions are executed in the service: This function can be useful when there is a not closed client connection with the module that does not allow a new operating connection to be established (the error LTR_WARNING_MODULE_IN_USE is returned). Calling of this function enables to reset a not closed connection. Also, the function can be useful if the program closes not having correctly completed operation with the module, e.g. having remained data acquisition started. Module reset enables to stop garbage transmission from the module switching it to the initial state. This function is only supported by the service ltrd (not supported by the previously used LTR Server program).
Parameters: hsrv — Control connection descriptor. crate_iface — Crate connection interface (similar to the so-named parameter LTR_OpenCrate()). crate_sn — Crate serial number (similar to the so-named parameter LTR_OpenCrate()). module_slot — Number of themodule slot (from LTR_CC_CHNUM_MODULE1 to LTR_CC_CHNUM_MODULE16). flags — Additional flags controlling function operation. Now they are not used therefore the field must be always equal to zero.
Returned value: Error code.

5.3.4.2 Setting of the history log level

Format: INT LTR_SetLogLevel (TLTR *hsrv, INT level, BOOL permanent)
Description: The function sets, messages of what level will be output to the history log by the service ltrd. All messages with less important level will not be output.
Parameters: hsrv — Control connection descriptor. level — Configurable level of output to the history log — value from en_LTR_LogLevel. permanent — If FALSE, changes concern only the current running of the service ltrd. If TRUE — changes are saved in the settings and will be considered after service re-start.
Returned value: Error code.

5.3.4.3 Acquisition of the current history log level

Format: INT LTR_GetLogLevel (TLTR *hsrv, INT *level)
Description: The function returns the set output level to the history log of the service ltrd.
Parameters: hsrv — Descriptor of the control connection. level — In this variable the set history log level is returned — the value from en_LTR_LogLevel.
Returned value: Error code.

5.3.4.4 Setting of the operation parameter of the service ltrd

Format: INT LTR_SetServerParameter (TLTR *hsrv, DWORD param, void *val, DWORD size)
Description: The function configures one parameter from the settings of the service ltrd operation. The format and the meaning of the value to be transmitted are determined by the parameter code and described in the description of each parameter. The parameter to be set up is applied immediately and save in the settings of the service ltrd. This function is only supported by the service ltrd (not supported by the previously used LTR Server program).
Parameters: hsrv — Control connection descriptor. param — Parameter code — value from en_LTRD_Params. val — Pointer to the data with configurable value. Format of data can depend on the parameter and be specified in the parameter description. size — Size of data to be transmitted as the parameter value (to which val points).
Returned value: Error code.

5.3.4.5Reading of the operation parameters of the service ltrd.

Format: INT LTR_GetServerParameter (TLTR *hsrv, DWORD param, void *val, DWORD *size)
Description: The function reads one parameter from the settings of the service ltrd operation. The format and the meaning of the value to be transmitted are determined by the parameter code and described in the description of each parameter. This function is only supported by the service ltrd (not supported by the previously used LTR Server program).
Parameters: hsrv — Control connection descriptor. param — Parameter code — value from en_LTRD_Params val — Pointer to data where parameter value will be saved. Format of data can depend on the parameter and be specified in the parameter description. size — Size of the array to which the variable val points. Different parameters can require different size to store the value.
Returned value: Error code.

5.3.4.6 Re-start of the service ltrd.

Format: INT LTR_ServerRestart (TLTR *hsrv)
Description: In case of successful execution of this command the service ltrd closes all client connections and all connections with crates and starts operation from the beginning, reading its settings again. Consequently, the current control connection via which the command is transmitted, becomes void - the only function, that can and must be called next, is LTR_Close().
Parameters: hsrv — Control connection descriptor.
Returned value: Error code.

5.3.4.7 Stop of the service ltrd.

Format: INT LTR_ServerShutdown (TLTR *hsrv)
Description: In case of successful execution of this command the service ltrd closes all client connections and all connections with crates and shuts down. Consequently, the current control connection via which the command is transmitted, becomes void - the only function, that can and must be called next, is LTR_Close().
Parameters: hsrv — Control connection descriptor.
Returned value: Error code.

5.3.5 Control functions for crate connection over Ethernet

These functions are used with the control connection. They enable to control the list of entries with IP-addresses of crates of the service ltrd, and to execute commands to establish and close connection between ltrd and the crates by IP-addresses using the entries from this list.

5.3.5.1 Acquisition of the entries list with crate IP-addresses

Format: INT LTR_GetListOfIPCrates (TLTR *hsrv, DWORD max_entries, DWORD ip_net, DWORD ip_mask, DWORD *entries_found, DWORD *entries_returned, TLTR_CRATE_IP_ENTRY *info_array)
Description: The function returns the list of entries with IP-addresses of the crates from the settings of the service ltrd (both with established connection with the crate and with not established connection). Also, with the entry the status of crate connection, that corresponds to this entry, is returned. The function enables to return not the full list of entries, but only those entries addresses of which satisfy the pre-set filter — only those IP-addresses are returned, that belong to the subnet configured by the standard method using IP-address (ip_net) and the subnet mask (ip_mask). I.e. if the address “192.168.1.0” and the subnet mask “255.255.255.0” are set, the entries with IP-addresses in the form of “192.168.1.x” will be returned, where x - any value within the range from 0 to 255. If it is necessary to obtain all entries, null indicators can be transmitted as an address and a mask. To obtain information on the status of the specific entry you can set up the required IP-address in full, and set up the mask equal to “255.255.255.255” To obtain random number of IP-entries the function can be firstly called with zero value of max_entries, in order to obtain number of entries in entries_found and then you can specify the array to receive the required number of entries and call the function again to obtain information on these entries.

Parameters:
hsrv — Control connection descriptor.
max_entries — Maximum number of entries, that can be received by the array info_array
ip_net — IP-address of the network, that is used to filter the entries to be returned, in 32-bit format.
ip_mask — Subnetmask to filter the entriesto be returned in 32-bit format.
entries_found — In this variable the total number of the found entries that satisfy the filter condition, is returned. This value can exceed max_entries.
entries_returned — Number of entries that was returned in the array info_array. In case of successful execution this value is equal to the least value of max_entries и entries_found.
info_array — Array of the structures TLTR_CRATE_IP_ENTRY, where the found entries with IP-addresses will be returned. This array must have sufficient size to store all entries max_entries. If in max_entries a zero value is transmitted, the null indicator can be transmitted as this parameter.

Returned value: Error code.

5.3.5.2 Adding of the entry with crate IP-address

Format: INT LTR_AddIPCrate (TLTR *hsrv, DWORD ip_addr, DWORD flags, BOOL permanent)

Description:
The function adds the entry with the specified parameter to the list of entries with IP-addresses of the crates of the service ltd.
To establish connection with the crate over Ethernet (using LTR_ConnectIPCrate()) its IP-address must already be in the list of entries.
If the list already has the entries, this function only changes entry flags (similar to LTR_SetIPCrateFlags()).

Parameters:
hsrv — Control connection descriptor.
ip_addr — IP-address of the crate in 32-bit format.
flags — Set of flags related to entry to be added. Combining of the values from en_LTR_CrateIpFlags by OR.
permanent — If FALSE, changes concern only the current running of the service ltrd. If TRUE — changes are saved in the settings and will be considered after service re-start.

Returned value: Error code.

5.3.5.3 Deleting of the entry with crate IP-address

Format: INT LTR_DeleteIPCrate (TLTR *hsrv, DWORD ip_addr, BOOL permanent)
Description: The function deletes the entry with the specified IP-address from the list of crate IP-addresses of the service ltrd. With that no connection should be established with the respective crate (entry status must differ from LTR_CRATE_IP_STATUS_ONLINE or LTR_CRATE_IP_STATUS_CONNECTING), otherwise the function returns an error. To delete the entry with the connected crate, firstly, you should disconnect from the crate via LTR_DisconnectIPCrate(). If the specified entry is not available in the list, function will not influence the list of addresses and immediately finish without an error.
Parameters: hsrv — Control connection descriptor ip_addr — IP-address of the crate in 32-bit format, the entry with which must be deleted. permanent — If FALSE, changes concern only the current running of the service ltrd. If TRUE — changes are saved in the settings and will be considered after service re-start.
Returned value: Error code.

5.3.5.4 Setting up of the connection with the crate using IP-address

Format: INT LTR_ConnectIPCrate (TLTR *hsrv, DWORD ip_addr)
Description: The function is the command of the service ltrd, indicating that the service should establish connection with the specified IP-address via Ethernet interface. The entry with the specified address must be in the list of entries with crae IP-addresses of the service (this list can be obtained using LTR_GetListOfIPCrates()), otherwise the function returns an error. This function finish means that the service received the command and started to connect to the crate, but connection still can not be established. With that the status of connection with the crate for the entry changes to LTR_CRATE_IP_STATUS_CONNECTING. Upon establishing connection this status changes to LTR_CRATE_IP_STATUS_ONLINE in case of success connection or to LTR_CRATE_IP_STATUS_ERROR in cse of an error that is the sign of operation completion. Obtaining of the current connection status is possible by obtaining the information on the netry using LTR_GetListOfIPCrates(), Also, in case of successful connection the crate will appear in the list of active crates that can be obtained using LTR_GetCrates() or LTR_GetCratesEx(). If the crate is already connected during this function calling (connection status LTR_CRATE_IP_STATUS_ONLINE) or connection is in progress (status LTR_CRATE_IP_STATUS_CONNECTING), the function will be successfully completed without execution of any actions.
Parameters: hsrv — Control connection descriptor. ip_addr — IP-address of the crate in 32-bit format, with which connection must be established.
Returned value: Error code.

5.3.5.5Breaking of the connection with the crate using IP-address

Format: INT LTR_DisconnectIPCrate (TLTR *hsrv, DWORD ip_addr)
Description: When calling this function the service breaks the connection with the crate connected over Ethernet with the specified address. The crate disappears from the list of active crates and the connection status for the entry with this IP-address changes to LTR_CRATE_IP_STATUS_OFFLINE. The entry with the same address must be in the list of entries with IP-addresses of the crates of the service ltrd, otherwise the function returns an error. If the entry with the specified address is available but there is no active connection (connection status differs from LTR_CRATE_IP_STATUS_ONLINE or LTR_CRATE_IP_STATUS_CONNECTING), the function is successfully completed without execution of any actions.
Parameters: hsrv — Control connection descriptor. ip_addr — IP-address of the crate in 32-bit format, with which connection must be broken.
Returned value: Error code.

5.3.5.6 Setting up the connection with all crates with auto-connection attribute

Format: INT LTR_ConnectAllAutoIPCrates (TLTR *hsrv)
Description: The function commands the service ltrd to establish the connection via Ethernet interface with all crates, for entries with IP-addresses of which the flag LTR_CRATE_IP_FLAG_AUTOCONNECT is set up. With that connection is started only for those entries for which there is no active connection, i.e. for the entries with the current status LTR_CRATE_IP_STATUS_ONLINE or LTR_CRATE_IP_STATUS_CONNECTING function calling does not have any effect. Similar to LTR_ConnectIPCrate() function finish only means that the connection process is started, and if necessary you can know that connection is established by the change of the connection status of the respective entries (that can be checked using LTR_GetListOfIPCrates(). If there are noentries with the set up flag LTR_CRATE_IP_FLAG_AUTOCONNECT, using which connection with the crate is established, the function is successfully completed without execution of any actions.
Parameters: hsrv — Control connection descriptor.
Returned value: Error code.

5.3.5.7 Breaking of the connection with all crates connected over Ethernet.

Format: INT LTR_DisconnectAllIPCrates (TLTR *hsrv)
Description: On this command the service ltrd closes all active connections with the crates, connected via Ethernet interface. All crates will be deleted from the list of active crates and all IP-addresses will change their connection status to LTR_CRATE_IP_STATUS_OFFLINE. If there were no crates connected over Ethernet when the function is called, the function is successfully completed without execution of any actions.
Parameters: hsrv — Control connection descriptor.
Returned value: Error code.

5.3.5.8 Configuration of the flags for entry with the crate IP-address

Format: INT LTR_SetIPCrateFlags (TLTR *hsrv, DWORD ip_addr, DWORD flags, BOOL permanent)
Description: The function changes the value of the flags, related to already available log with IP-address, to the specified value. The entry with set IP-address must be available in the list of entries of the service ltrd, otherwise the function will be completed with an error.
Parameters: hsrv — Control connection descriptor. ip_addr — IP-address of the crate in 32-bit format, for entry with which the flags should be changed. flags — New set of flags related to the entry. Combining of the values from en_LTR_CrateIpFlags by OR. permanent — If FALSE, changes concern only the current running of the service ltrd. If TRUE — changes are saved in the settings and will be considered after service re-start.
Returned value: Error code.

5.3.6 Functions of data exchange with modules

These functions are used to transmit and receive module's data. Generally, the user does not need to use these functions directly, as to work with the modules (specific modules) the functions from the designated libraries are used.

These functions are used only for connections with the modules and not applicable to the control connections.

5.3.6.1 Data receiving from the module

Format: INT LTR_Recv (TLTR *hmodule, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)
Description: The function receives data from the module in 32-bit words in the special format of LTR words. Besides, the function analyzes the information on the received synchro-labels and on buffer overflow in the service ltrd, updating values of the field TLTR::flags of the module descriptor and filling-in the array tmark. The function returns control either when receives the requested number of words or after timeout. With that actual received number of words can checked by the returned value.
Parameters: hmodule — Descriptor of the connection with the module. data — Array where the received words will be saved. It must have size of "size" of 32-bit words tmark — Pointer 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. I.e. the element tmark[i] corresponds to the received word data[i], indicating the number of "START" and "SECOND" labels corresponding to this word. Format of these words is described in the chapter on synchro-labels. If synchro-labels are not used, you can transmit NULL as the parameter. size — The requested number of 32-bit words that should be received from the module. timeout — Timeout for operation execution in milliseconds. Value 0 means the default time-out value. 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 words that were stored in the array "data".

5.3.6.2 Data transmission to the module

Format: INT LTR_Send (TLTR *hmodule, const DWORD *data, DWORD size, DWORD timeout)
Description: The function sends data from to the module in 32-bit words in the special format of LTR words. The function returns control either when all data are recorded to the buffer for transmission or after timeout. With that the actual number of words recorded in the buffer for transmission can be checked by the returned value. I.e. the returned value does not guarantee that this number of words was set for transmission, but still these data may not reach the module by the moment of function finish.
Parameters: hmodule — Descriptor of the connection with the module. data — Array containing data in the form of size 32-bit words that must be transmitted to the module. size — Number of words that must be transmitted to the module. timeout — Timeout for operation execution in milliseconds. Value 0 means the default time-out value. If there is no space to record the requested number of words in the buffer for transmission during the pre-set time, the function still will return control, having returned the actual number of words recorded in the buffer 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 words recorded in the buffer for transmission, that were set for transmission to the module.

5.3.6.3 Reading of the time of the last second label.

Format: INT LTR_GetLastUnixTimeMark (TLTR *hmodule, LONGLONG *unixtime)
Description: This functions returns time value that corresponds to the last found reception of the extended "SECOND" label with absolute time when receiving data via this connection using LTR_Recv(). This function operates only if the crate supports generation of the extended "SECOND" label with indication of absolute time by any method. E.g. using the time server that transmits the time over the protocol IRIG-B. If the "SECOND" label with absolute time was not received the value 0 will be returned.
Parameters: hmodule — Descriptor of the connection with the module unixtime — Absolute time value in seconds from January, 1, 1970. (unixtime).
Returned value: Error code.

5.3.7 Auxiliary functions

5.3.7.1 Acquisition of the text error message

Format: LPCSTR LTR_GetErrorString (INT err)
Description: The function returns the line that corresponds to the transmitted error code In CP1251 coding for OS Windows or UTF-8 coding for OS Linux. The function supports only error codes determined by the given library and returned by the functions of the given library. The libraries for the specific modules can have additional error codes and their own functions of receiving of the textual error description that support these additional error codes.
Parameters: err — Error code.
Returned value: Pointer to the line containing the message error.

5.3.7.2 Timeout default setup for connection

Format: INT LTR_SetTimeout (TLTR *hnd, DWORD tout)
Description: The function sets up timeout by default for execution of operations for the specified connection. When opening connection this timeout is equal to LTR_DEFAULT_SEND_RECV_TIMEOUT. For the control connection this timeout determines the time of execution of any command (from request transmission to the service to response receiving). For the connection with the module — it is timeout that is used in LTR_Recv() or LTR_Send(), if timeout zero value is transmitted in them.
Parameters: hnd — Connection descriptor. tout — Default time-out in ms.
Returned value: Error code.