User interface library of LTR24 module

Programmer Manual

Revision 1.0.1

L-CARD

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

DAQ SYSTEMS DESIGN, MANUFACTURING & DISTRIBUTION

Authors:
O.A. Kovalev, A.V. Borisov

You can download the last version of the manual on the web-site of LLC "L-Card"

LLC "L-Card" reserves the right to update the documentation without notifying the users.

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

Revision history of this document.

Revision	Date	Notes to the updates
1.0.0	24.04.2013	The first revision available for user
1.0.1	3.09.2013	Description of the possibility to correct the AFC module, new modes for LTR24-2 us added, changes of LTR24_ProcessData() and structures' fields are taken into account.

Contents

1. What this document is about

This document is a programmer manual. Here the issues of applied programming for the LTR24 module using the library *ltr24api* are addressed. Issues concerned connection, operating principles and hardware structure of the LTR24 module are not addressed here. Information on this subject is provided in the document "LTR Crate System. User manual".

2. General information

The *ltr24api* library is an interface of the applied programming of the LTR24 data acquisition module of the LTR crate system. In the context of programming this module is a 24-bit 4-channel ADC.

The main features of the LTR24 module:

- 4 channels that can operate either in "Dif. output" or "ICP-input" mode.
- 2 ranges (±2V and ±10V in the "Dif. output" mode or ~1V and ~5V in the "ICP-input" mode)
- 16 sampling frequencies (from 610.352 Hz to 117.188 kHz)
- 2 data formats (20- and 24-bit)
- Possibility to switch one of the test modes on - measurement of the own zero or "ICP-test"
- Constant component cutoff mode per channel

"ICP-mode" and "ICP-test" modes are available only for LTR24-2 modification, that has additional inputs to connect ICP-sensors. Programmatically, you can check whether these modes are available using the field `SupportICP` of the structure with the information on the module (type TINFO_LTR24) after opening the communication channel for the module.

The module has certain setting restrictions. In the following sections they are described in details.

2.1. Data formats

The module enables to operate in two data formats: 20-bit and 24-bit. These formats are slightly different in terms of capabilities. IN

Table 2-1 contains the parameters by which they are different.

Table 2-1. Comparison of data format capabilities

Parameter	**Data format**	
	20-bit	**24-bit**
Amount of raw data per count, 32-bit word	1	2
Data continuity check	Bit, set to 1 in every 15th word	Counter for the module 15

Maximum number of switched on channels	4	See Table 2-2
Monitoring of input path overload	–	+

Application of 24-bit data format increases accuracy but also increases data flow from the module twice that can be a problem for application of a large number of modules in a single crate. Restriction of the maximum number of channels is caused by the limited capacity of the interface with the module. The module also enables to monitor input path overload that must be considered in certain situations. See detailed information on input path overload in "LTR Crate System. User manual".

Table 2-2. Maximum number of channels when using 24-bit data format

Sampling frequency, kHz	**Maximum number of channels**
117.1875	2
78.125	3
58.59375 and below	4

2.2. Calibration

The module is shipped in factory calibrated state. Calibration factors are stored in the module ROM and read using the function `LTR24_GetConfig`. Factory factors read are stored in the field `ModuleInfo` of the module control structure. These factors must not be changed by the user. Also, the copy of factors is stored in the fields `CalibCoef` and `AfcCoef` of the control structure itself, and they are used when indicating the respective flags. Thus, the user has the possibility to change factory factors for his/her own factors without changing information in `ModuleInfo`. E.g. this can be useful it is necessary to calibrate the whole analog path up to the data acquisition module.

For each channel, each sampling frequency and each range individual calibration factors are used. With that two factors are used for calibration: scaling factor (scale factor) and offset.

Also, the module's ROM stores factors for calculation of filters for AFC correction. Detailed information is provided in section *"AFC correction"* of this document.

Besides, for the LTR24-2 module in the module's ROM the measured precise values of the current sources are stored for each channel to connect ICP-sensors. These values can be used to measure external resistive-strain sensors.

3. Application

3.1. Connection to the project

To connect the *ltr24api* library to the project in C/C++ language it is necessary to perform the following:

For **OS Windows**:

1. The ltrdll.exe libraries must be installed.
2. Connect the header file *ltr24api.h*:

```
#include <ltr/include/ltr24api.h>
```

3. Add the directory to the catalog list with respect to which

   headers are placed in “ltr/include”. In case of default installation, this path: “C:\Program Files\L-Card” or for 64-bit systems “C:\Program Files (x86)\L-Card”.
4. Connect the *ltr24api.lib* import library for the desired compiler.

   - *Microsoft Visual C++* – from “ltr\lib\msvc”

   - *Borland C++/Borland C++ Builder* – from “ltr\lib\borland”
5. To start the assembled program it is necessary that the *ltr24api.dll* library (and the *ltrapi.dll* and *ltrmcs.dll* library on which it depends) are in the same directory as the program, or in the directory from the PATH environmental variable (the installer installs them in “%WINDIR%/system32”).

For **OS Linux**:

1. Install the libraries either assembling packets or assembling in your own way ltr_cross_sdk.pdf)
2. Connect the header file *ltr24api.h*:

```
#include <ltr/include/ltr24api.h>
```

3. If the directory “ltr/include” is not located in the standard path, add the respective path to search for the header files, e.g. using the key – I<path> when assembling GCC. When installing the packets the headers are installed in the standard directory “/usr/include” and you do not need to indicate the path.

   Connect the libltr24api.so library to the project (e.g using the key –lltr24api when assembling GCC). If the libraries are not located in the standard path, it should be indicated using the key – L<path>. When installing packets the libraries are installed in /usr/lib and you do not need to indicate the path.
4. To start libltr24api.so and all libraries on which it depends must be available in one of the standard directories or directories set via the variable LD_LIBRARY_PATH or any other method.

3.2. Working with the library

The LTR24 module is controlled via the control structure (`TLTR24`), reflectig the current state of the module, communication channel, etc. One structure of `TLTR24` is used to control one module.

When working with the library *ltr24api* it is necessary to observe the following mandatory execution sequence:

1. Initialization of control structure fields (LTR24_Init).
2. Opening the communication channel for the module (LTR24_Open).
3. Wotking with the module, other functions calling.
4. Closing the communication channel for the module (LTR24_Close).

Typical calls sequence is as follows:

1. Initialization of control structure fields (LTR24_Init).
2. Opening the communication channel for the module (LTR24_Open).
3. Receipt of the information from ROM, including calibration factors (LTR24_GetConfig)
4. Filling in the fields of the module control structure responsible for the module configuration
5. Recording the set configuration to the module (LTR24_SetADC).
6. Data acquisition start (LTR24_Start)
7. Receipt of the data chunk using LTR24_Recv
8. Processing of the received data chunk using LTR24_ProcessData
9. If it is necessary to receive additional data, for switching to item 7, otherwise switching to item 10.
10. Data acquisition stop (LTR24_Stop).
11. Closing the communication channel for the module (LTR24_Close).

When opening the communication channel the module can be in two modes: configuration and data acquisition. In the configuration mode data acquisition parameters are set, the module operates in the "request/response" mode. In the data acquisition mode stream data transmission is performed from all switched on ADC channels.

In the configuration mode the information is read from the module's ROM (LTR24_GetConfig), module parameters are set via filling in the fields of the control structures, parameters are recorded to the module (LTR24_SetADC). Upon completion of module setting data acquisition is started using the function LTR24_Start, then the module switches to the data acquisition mode.

In the data acquisition mode you can set only constant component cutoff mode per each channel individually (LTR24_SetACMode) and the mode of own zero measurement for

all channels simultaneously (LTR24_SetZeroMode). In this mode the parameters are set only using the special functions but not using the fields of the control structure. The main purpose of this mode is to receive data from the module and their processing (functions LTR24_Recv, LTR24_RecvEx, LTR24_ProcessData). Data are received as frames (see Frame). Switching to the configuration mode is performed when data acquisition is stopped (LTR24_Stop).

3.3. Module setting

To set the module it is necessary to fill in the fields of the control structure that are responsible for the module parameters, then call LTR24_SetADC. The settings that are set for the module or for all channels simultaneously are represented by the fields of the control structure itself, and the settings that are set individually for each channel - by the fields of the structures' array ChannelMode. Data acquisition should not be started during setting.

The following parameters can be set:

- ADC sampling frequency (field ADCFreqCode)
- Operation mode for each channel (see Table 3-1)
- Selection of the channels via which data receipt is permitted (field Enable from ChannelMode)
- Range for each channel (field Range from ChannelMode)
- The value of the current source for ICP-inputs – set for all inputs (field ISrcValue)
- Data format (field DataFmt)

Table 3-1. Determination of the mode for each channel

TestMode	ICPMode	AC	Mode
FALSE	FALSE	FALSE	Differential input without constant component cutoff
FALSE	FALSE	TRUE	Differential input with constant component cutoff
FALSE	TRUE	X	ICP-input mode
TRUE	FALSE	X	Own zero measurement mode
TRUE	TRUE	X	"ICP-test" mode

3.4. AFC correction

The library functions enable to correct the module's AFC using the additional filters. For all ranges AFC cut of the module input path is corrected using the feedback filter as described in the article “Method of AFC slope fine correction using a simple digital filter”. The module's ROM stores the value of the ratio of the pre-set frequency signal amplitude at maximum AD sampling frequency measured by the module and the actual fed signal amplitude and the value of the signal frequency in Hz.

Besides, for ADC frequencies of 39.0625 kHz and below the additional AFC correction of the ADC itself is performed using the infinite-impulse response filter of the 2nd order, factors of which are also stored in the module's ROM.

To perform AFC correction it is necessary to transmit the flag LTR24_PROC_FLAG_AFC_COR to the function

LTR24_ProcessData when processing the data (factors for this purpose must have been already read using the function LTR24_GetConfig). With that, by default it is assumed that all received data are transmitted to

LTR24_ProcessData one after another without interruptions and repetitions, and the filters are not reset between callings of

LTR24_ProcessData. Otherwise and when the next data chunk to be processed follows not immediately the previous one, you should indicate this using the flag LTR24_PROC_FLAG_NONCONT_DATA.

Of course, the filters are always reset when starting acquisition using LTR24_Start.

4. API description

4.1. Constants

`LTR24_VERSION_CODE    0x02000000UL`

The current library version (2.0.0.0).

`LTR24_CHANNEL_NUM     4`

Number of channels.

`LTR24_RANGE_NUM       2`

Number of ranges in the differential input mode.

`LTR24_ICP_RANGE_NUM   2`

Number of ranges in the ICP-input mode.

`LTR24_FREQ_NUM        16`

Number of sampling frequencies.

`LTR24_I_SRC_VALUE_NUM      16`

Number of current source values.

`LTR24_NAME_SIZE          8`

Size of the name field.

`LTR24_SERIAL_SIZE  16`

Size of the serial number field.

Sampling frequency codes

`LTR24_FREQ_117K` `0`

117.1875 kHz

`LTR24_FREQ_78K` `1`

78.125 kHz

`LTR24_FREQ_58K` `2`

58.59375 kHz

`LTR24_FREQ_39K` `3`

39.0625 kHz

`LTR24_FREQ_29K` `4`

29.296875 kHz

`LTR24_FREQ_19K` `5`

19.53125 kHz

`LTR24_FREQ_14K` `6`

14.6484375 kHz

`LTR24_FREQ_9K7` `7`

9.765625 kHz

`LTR24_FREQ_7K3`	8	7.32421875 kHz
`LTR24_FREQ_4K8`	9	4.8828125 kHz
`LTR24_FREQ_3K6`	10	3.662109375 kHz
`LTR24_FREQ_2K4`	11	2.44140625 kHz
`LTR24_FREQ_1K8`	12	1.8310546875 kHz
`LTR24_FREQ_1K2`	13	1.220703125 kHz
`LTR24_FREQ_915`	14	915.52734375 Hz
`LTR24_FREQ_610`	15	610.3515625 Hz

Range codes in the differential input mode

`LTR24_RANGE_2`	0	Range ±2 V.
`LTR24_RANGE_10`	1	Range ±10 V.

Range codes in the ICP-input mode

`LTR24_ICP_RANGE_1`	0	Range ~1 V.
`LTR24_ICP_RANGE_5`	1	Range ~5 V.

Current source values

`LTR24_I_SRC_VALUE_2_86`	0	2.86 mA.
`LTR24_I_SRC_VALUE_10`	1	10 mA.

Format codes

`LTR24_FORMAT_20        0`

20-bit data format.

`LTR24_FORMAT_24        1`

24-bit data format.

Flags controlling data processing

`LTR24_PROC_FLAG_CALIBR           0x00000001`

Sign that you should apply the calibration factors to the data.

`LTR24_PROC_FLAG_VOLT             0x00000002`

Flag to convert ADC codes in Volts.

`LTR24_PROC_FLAG_AFC_COR          0x00000004`

Sign that it is necessary to perform AFC correction.

`LTR24_PROC_FLAG_NONCONT_DATA    0x00000100`

The sign that non-continuous data are being processed.

Errors codes

`LTR24_ERR_INVAL_FREQ        -10100`

Incorrect sampling frequency is set.

`LTR24_ERR_INVAL_FORMAT      -10101`

Invalid data format is set.

`LTR24_ERR_CFG_UNSUP_CH_CNT -10102`

For the set frequency and 24-bit format the pre-set number of channels is not supported.

`LTR24_ERR_INVAL_RANGE       -10103`

Invalid channel range.

`LTR24_ERR_WRONG_CRC   -10104`

Invalid check sum of EEPROM.

`LTR24_ERR_VERIFY_FAILED     -10105`

Verification error of the record in EEPROM.

`LTR24_ERR_DATA_FORMAT       -10106`

Invalid data format in the processed counts.

`LTR24_ERR_UNALIGNED_DATA    -10107`

Non-aligned data.

LTR24_ERR_DISCONT_DATA -10108

Failure of the data counter in the processed counts.

LTR24_ERR_CHANNELS_DISBL -10109

No channel is enabled.

LTR24_ERR_UNSUP_VERS -10110

Version of the format of the control structure is not supported.

LTR24_ERR_FRAME_NOT_FOUND -10111

Start of frame is not found.

LTR24_ERR_OPEN_MCS_MOD -10112

Failure to open the module to work with saving the context.

LTR24_ERR_NO_SAVED_MCS -10113

No saved context.

LTR24_ERR_MCS_NOT_VALID -10114

Saved context is invalid.

LTR24_ERR_MCS_DIFF_MID -10115

The saved context belongs to other module.

LTR24_ERR_UNSUP_FLASH_FMT -10116

Unsupported data format in the module's Flash-memory.

LTR24_ERR_INVAL_I_SRC_VALUE -10117

Incorrect current source value is set.

LTR24_ERR_UNSUP_ICP_MODE -10118

This module modification does not support ICP-mode.

4.2. Data types and structures

TLTR24_AFC_IIR_COEF

```
typedef struct {
double a0;     double
a1;     double b0; }
TLTR24_AFC_IIR_COEF;
```

Infinite-impulse response filter factors for AFC correction

TLTR24_AFC_COEFS

```
typedef struct {
double AfcFreq;
    double FirCoef[LTR24_CHANNEL_NUM][LTR24_RANGE_NUM];
    TLTR24_AFC_IIR_COEF AfcIirCoef;
} TLTR24_AFC_COEFS;
```

Set of factors for module's AFC correction.

`AfcFreq`

Signal frequency for which the ratio of amplitudes is measured and saved in `FirCoef`

`FirCoef`

Set of sine signal measured amplitude and actual amplitude ratios for maximum sampling frequency and frequency of the signal from `AfcFreq` for each channel and each range

`AfcIirCoef`

Infinite-impulse response filter factors for ADC's AFC correction on the sampling frequencies of 39.0625 kHz and below

TINFO_LTR24

```
typedef struct {
    CHAR        Name[LTR24_NAME_SIZE];
    CHAR        Serial[LTR24_SERIAL_SIZE];
    BYTE        VerPLD;
    BOOL        SupportICP;
DWORD       Reserved[8];
struct {         float
Offset;         float
Scale;
    } CalibCoef[LTR24_CHANNEL_NUM][LTR24_RANGE_NUM][LTR24_FREQ_NUM];
TLTR24_AFC_COEFS AfcCoef;
    double ISrcVals[LTR24_CHANNEL_NUM][LTR24_I_SRC_VALUE_NUM];
} TINFO_LTR24;
```

Contains information on the module. All information except for values of the field `SupportICP` and `VerPLD`, is taken from ROM of the module and valid only after calling `LTR24_GetConfig`.

`Name`

Module name (“LTR24”).

`Serial`

Module serial number.

`VerPLD`

FPGA firmware version.

`SupportICP`

Sign, whether the module supports measurement mode from the ICP-sensors. For LTR24-2 this field is equal to TRUE, for other modifications – FALSE.

`Reserved`

Reserved fields. Always equal to 0.

`CalibCoef`

Factory calibration factors for each channel, range and frequency.

`Offset`

Offset.

`Scale`

Scaling factor.

`AfcCoef`

Factors for AFC correction.

`ISrcVals`

Measured values of the current sources for each channel (only for LTR24-2).

TLTR24

```
struct TLTR24 {
  INT  Size;
  TLTR Channel;
  BOOL Run;
  BYTE ADCFreqCode;
  double ADCFreq;     BYTE
DataFmt;
  BYTE    ISrcValue;
  BOOL    TestMode;   DWORD
Reserved[16];    struct {
       BOOL Enable;
       BYTE Range;
       BOOL AC;
       BOOL ICPMode;
       DWORD   Reserved[4];
  } ChannelMode[];    TINFO_LTR24
ModuleInfo;      struct {
       float Offset;
       float Scale;
 } CalibCoef[LTR24_CHANNEL_NUM][LTR24_RANGE_NUM][LTR24_FREQ_NUM];
  TLTR24_AFC_COEFS AfcCoef;
  PVOID Internal;
};
```

Module control structure. Stores the module current settings, information about its state, communication circuit structure. Is transmitted to the most of library functions. Some structure fields can be changed by the user to configurate module parameters. Prior to application requires initialization using the function `LTR24_Init`.

`Size`

Size of the structure `TLTR24`. Filled in automatically when calling the function `LTR24_Init`.

`Channel`

Communication channel for the LTR server.

`Run`

The current data acquisition state (`TRUE` – data acquisition is started).

`ADCFreqCode`

Sampling frequency code. Set equal to one of constants "Sampling frequency codes". **Specified by the user**.

`AdcFreq`

Sampling frequency value in Hz. Filled in with the sampling frequency value that corresponds to the code in the field `ADCFreqCode`, after execution of the function `LTR24_SetADC`.

`DataFmt`

Data format. Set equal to one of the constant of "Range codes in the ICP-mode"

`LTR24_ICP_RANGE_1`   `0`

Range ~1 V.

`LTR24_ICP_RANGE_5`   `1`

Range ~5 V.

Current source values

`LTR24_I_SRC_VALUE_2_86` 0

2.86 mA.

`LTR24_I_SRC_VALUE_10` 1

10 mA.

Format codes. **Specified by user.**

`ISrcValue`

Value of the current source for all ICP-sensors connection channels. Set equal to one of constants "Current source values". Relevant only for LTR24-2. **Specified by user.**

`TestMode`

Switching on the test modes ("Zero measurement" or "ICP-test" depending on the value of the field `ICPMode` for each channel) for all channels (TRUE – ON). **Specified by user.**

`Reserved`

Reserved. The field must not be changed by the user.

`ChannelMode`

Channel modes. **All fields are specified by the user.**

`Enable`

Enabling the channel. If it is equal to TRUE, the module will transmit words corresponding to the count from the given channel, FALSE – not.

`Range`

Channel range. Set equal to one of the constants "Range codes in the differential input mode" or "Range codes in the ICP-input code" depending on the value of the field `ICPMode`.

`AC`

Constant component cutoff mode (TRUE – ON). It is relevant only if the field `ICPMode` is equal to FALSE.

`ICPMode`

Switching the ICP-input measurement mode on. If FALSE – the mode "Dif. input" or "Zero measurement) is used (depending on the field TestMode), if TRUE – the mode "ICP input" or "ICP test".

`Reserved`

Reserved. The field must not be changed by the user.

`ModuleInfo`

Module information

`CalibCoef`

Calibration factors applied for data correction n the function

LTR24_ProcessData for each channel, range and frequency. When calling `LTR24_GetConfig` factory calibration factors are copied to these fields (th same as in `ModuleInfo`). But, if necessary, the user can record his/her own factors here.

`Offset`

Offset.

`Scale`

Scaling factor.

`AfcCoef`

Factors for AFC correction applied in the function

LTR24_ProcessData. When calling `LTR24_GetConfig` the values from the module's ROM are copied to these fields (the same as in `ModuleInfo`).

`Internal`

Pointer to the structure with the parameters that are used only by the library and not available for the user.

4.3. Functions

LTR24_GetVersion

```
DWORD  LTR24_GetVersion    (void);
```

Used to determine compatibility of the software and the current library version by its version number. Library version number with which the program was compiled is available via the constant `LTR24_VERSION_CODE`.

Returns:

The current *ltr24api* library version.

LTR24_GetErrorString

```
LPCSTR      LTR24_GetErrorString (INT     error);
```

Returns the textual description of an error by its code. Textual description is a line ending with the null symbol. Description coding – WINDOWS-1251 for OS Windows or UTF-8 for OS Linux.

`error  [in]`

Error code.

Returns:

Textual description of the error code.

LTR24_Init

```
INT    LTR24_Init     (TLTR24  *ltr24);
```

Initializes the fields of module control structure. Prior to application of the control structure in other functions it should be initialized.

`ltr24  [in]`

Module control structure.

Returns: `LTR_OK`.

LTR24_Open

```
INT        LTR24_Open             (TLTR24  *ltr24,
                                   DWORD  ip_addr,
                                   WORD     port,
                                   const CHAR *serial,
                                   INT      slot
                                   );
```

Opens the communication channel for the module. Connection is established via the LTR server, started on the host with IP-address `addr` and listening to the TCP-port `port`. The specific module is selected by the crate serial number `serial` and slot number in the crate `slot`.

If serial is equal to `NULL` or an empty line (“”), the first crate in the list of the LTR server is selected. As IP-address and port number the respective constants `SADDR_DEFAULT` and `SPORT_DEFAULT` can be used, setting default values

(127.0.0.1:11111). Bit order in IP-address: 1.2.3.4 > `0x01020304`.

Upon completion of work with the module it is necessary to close the communication channel using the function `LTR24_Close`.

`ltr24  [in]`

Module control function.

`ip_addr     [in]`

IP-address of the host, on which the LTR server is started.

`port   [in]`

The port to be listened by the LTR server.

`serial      [in]`

Crate serial number.

`slot   [in]`

Slot number in the crate. Slots are numbered from 1.

Returns:

`LTR_OK` or error code.

LTR24_Close

```
INT    LTR24_Close    (TLTR24  *ltr24);
```

Closes the communication channel for the module. Upon completion of work with the module it is necessary to close the communication channel for it.

`ltr24  [in]`

Module control structure.

Returns:

`LTR24_OK` or error code.

LTR24_IsOpened

```
INT    LTR24_IsOpened      (TLTR24  *ltr24);
```

Checks whether the communication channel for the module is opened.

`ltr24  [in]`

Module control structure.

Returns:

`LTR_OK`, if the module is opened or error code.

LTR24_GetConfig

```
INT    LTR24_GetConfig     (TLTR24  *ltr24);
```

Reads the information from the module's ROM, updates the structure `ModuleInfo`, `CalibCoef` and `AfcCoef` of the module control structure.

`ltr24  [in]`

Module control structure.

Returns:

`LTR_OK` or error code.

LTR24_SetADC

```
INT    LTR24_SetADC   (TLTR24  *ltr24);
```

Configures the module in accordance with the selected settings. Setting is performed by filling in the fields of the control structure intended for changing by the user.

`ltr24 [in]`

Module control structure.

Returns:

`LTR_OK` or error code.

LTR24_Start

```
INT    LTR24_Start    (TLTR24  *ltr24);
```

Starts data acquisition from the module. Prior to data acquisition the module must be configured using the function `LTR24_SetADC`. Module configuration during data acquisition is not available, except for the own zero measurement and the constant component cutoff mode. Changing of these parameters in the data acquisition mode is performed with the functions `LTR24_SetZeroMode` and `LTR24_SetACMode`.

`ltr24 [in]`

Module control structure.

Returns:

`LTR_OK` or error code.

LTR24_Stop

```
INT    LTR24_Stop     (TLTR24  *ltr24);
```

Stops data acquisition from the module. When data acquisition is stopped the module can be re-configured.

`ltr24 [in]`

Module control structure.

Returns:

`LTR_OK` or error code.

LTR24_Recv

```
INT        LTR24_Recv             (TLTR24     *ltr24,
                                   DWORD      *data,
                                   DWORD      *tmark,
                                   DWORD      size,
                                   DWORD      timeout,
                                   );
```

Receives raw data from the module and adds them to the array data. The function takes control again either when the requested number of 32-bit words is received or when the time interval specified in the parameter timeout is elapsed. In 24-bit format two data words correspond to each ADC count, and in 20-bit format - one data word.

Word sequence order: firstly, the first count of the first enabled channel, then the first count from the second channel, … the first count from the n-th channel., then the seconds counts per each enabled channel, etc. Counts are received only for those channels for which data acquisition is enabled.

Values of the SECOND and START labels are added to the array tmark. Each element of the array data is assigned to the element tmark. If it is not necessary to receive the second labels the parameter tmark is set equal to NULL.

The amount of data requested is measured in 32-bit words.

Received raw data are transmitted to the function LTR24_ProcessData for correction and conversion to physical values.

ltr24 [in]

Module control structure.

data [out]

Array for recording data.

tmark [out]

Array for recording the SECOND and START labels.

size [in]

Amount of data requested.

timeout [in]

Timeout for waiting for data, in ms.

Returns:

Amount of data words received (≥ 0) or error code (< 0).

LTR24_RecvEx

```
INT        LTR24_RecvEx             (TLTR24  *ltr24,
                                      DWORD    *data,
                                      DWORD    *tmark,
                                      DWORD    size,
                                      DWORD  timeout,
                                      LARGE_INTEGER *time
                                      );
```

Receives raw data from the module and adds them to the array data. The function is similar to LTR24_Recv, duty additionally records absolute time of receipt for each data word measured by hours in the crate-controller. Time has the format of POSIX, 64 bits. Of it not necessary to receive the absolute time labels the parameter `time` is set equal to `NULL`, or the function `LTR24_Recv` is used.

`ltr24 [in]`

Module control structure.

`data [out]`

Array for recording data.

`tmark [out]`

Array for recording the SECOND and START labels.

`size [in]`

Amount of data requested.

`timeout [in]`

Timeout for waiting for data, in ms.

`time [out]`

Array for recording the absolute time of receipt.

LTR24_ProcessData

```
INT        LTR24_ProcessData      (TLTR24   *ltr24,
                                   const DWORD *input,
                                   double   *output,
                                   INT       *size,
                                   DWORD     flags,
                                   BOOL   *overload
                                   );
```

Converts raw data, applies calibration factors, checks data continuity. Raw data must be transmitted adjusted by the frame edge and contain integer number of frames (see *Frame*). In case of non-adjusted frame transmission the function cuts off incomplete frames and returns an error. In case of interruptions the function sends an error.

By default the function assumes that all data received from the specific module are processed by the function`LTR24_ProcessData` and processed once (i.e. the data chunk transmitted to the function corresponds to the data following immediately the previously processed data). If it is not so it is necessary to indicate it with the flag `LTR24_PROC_FLAG_NONCONT_DATA`.

Output data are returned either in ADC codes or in Volts (if the flag `LTR24_PROC_FLAG_VOLT` is specified).

If the flag `LTR24_PROC_FLAG_CALIBR` is specified, the calibration factors from the array `CalibCoef` of the module control structure are applied.

The function can also perform module's AFC correction using the factors from the field AfcCoef of the module control structure. To do this it is necessary to transmit the flag `LTR24_PROC_FLAG_AFC_COR`.

When working in 24-bit format the count consists of two raw words therefore the number of elements in the output arrays must be 2 times less. The array `overload` must contain the same number of elements as in the array `output`.

`ltr24` `[in]`

Module control structure. `input` `[in]`

Raw data array.

`output` `[out]`

The array for recording the processed data.

`size` `[in,out]`

Amount of raw data. After execution – the amount of data in the output array.

`flags` `[in]`

Set of the flags from "Flags controlling data processing". Several flags combined via logical "OR" can be transmitted.

`overload` `[out]`

The array for recording information on input overload (this sign is monitored only in 24-bit data format).

Returns:

`LTR_OK` or error code.

LTR24_SetZeroMode

```
INT         LTR24_SetZeroMode       (TLTR24     *ltr24,
                                     BOOL       enable
                                     );
```

Changes state of the own zero measurement mode for all channels. This function is only used during data acquisition. For setting in the configuration mode the field `TestMode` of the module control structure is used.

`ltr24` `[in]`

Module control structure.

`enable` `[in]`

State of the own zero measurement mode.

Returns:

`LTR_OK` or error code.

LTR24_SetACMode

```
INT         LTR24_SetACMode       (TLTR24     *ltr24,
                                   BYTE       chan,
                                   BOOL       enable
                                   );
```

Changes the state of the constant component cutoff mode for the selected channel. This function is only used during data acquisition. For setting in the configuration mode the field AC is used for each channel in the module control structure.

ltr24 [in]

Module control structure. chan [in]

Channel number.

enable [in]

State of the constant component cutoff mode.

Returns:

LTR_OK or error code.

LTR24_StoreMcs

```
INT    LTR24_StoreMcs     (TLTR24  *ltr24);
```

Stores the module control structure in the crate controller. In case of connection fault it enables to restore the whole context without data acquisition stop.

This possibility is only available for the crate-controllers with MCS extension (only in the crate-controllers LTR032).

ltr24 [in]

Module control structure.

Returns:

LTR_OK or error code.

LTR24_RestoreMcs

```
INT         LTR24_RestoreMcs      (TLTR24  *ltr24,
                                   DWORD  ip_addr,
                                   WORD     port,
                                   const CHAR *serial,
                                   BYTE     slot
                                   );
```

Restores the module control structure from the crate controller. The function is similar to `LTR24_Open`, except for the fact that it attempts to restore the saved context without module resetting. The communication channel for the module must be closed.

This possibility is only available for the crate-controllers with MCS extension (only in the crate-controllers LTR032).

`ltr24 [in]`

Module control function.

`ip_addr [in]`

IP-address of the host, on which the LTR server is started.

`port [in]`

TCP port, that is listened by the LTR server.

`serial [in]`

Crate serial number.

`slot [in]`

Slot number in the crate. Slots are numbered from 1.

Returns:

`LTR_OK` or error code.

LTR24_ClearMcsSlot

```
INT    LTR24_ClearMcsSlot  (TLTR24   *ltr24);
```

Deletes saved data on the control structure.

This possibility is only available for the crate-controllers with MCS extension (only in the crate-controllers LTR032).

`ltr24 [in]`

Module control structure.

Returns:

`LTR_OK` or error code.

LTR24_InvalidateMcsSlot

```
INT    LTR24_InvalidateMcsSlot (TLTR24   *ltr24);
```

Makes data stored in the slot invalid. Used to avoid the situation during module configuration when the saved data and the actual module state are different. After re-configuration it is necessary to save the context again.

This possibility is only available for the crate-controllers with MCS extension (only in the crate-controllers LTR032).

`ltr24  [in]`

Module control structure.

Returns:

`LTR_OK` or error code.

LTR24_FindFrameStart

```
INT        LTR24_FindFrameStart    (TLTR24  *ltr24,
                                     const DWORD *data,
                                     INT       size,
                                     INT       *index
                                     );
```

Finds number of a word that is start of frame. Used to restore aligning by the frame edge in the disordered data flow after restoration of the control structure from the crate controller.

`ltr24  [in]`

Module control structure.

`data   [in]`

Raw data array.

`size   [in]`

Amount of raw data.

`index  [out]`

Index of start of frame..

Returns:

`LTR_OK` or error code.

4.4. Data formats

Frame

Raw data from the module are transmitted as frames. A frame is a sequence of counts for all switched on channels in ascending order of the channel number. For 20-bit data format the count corresponds to one 32-bit word, for 24-bit format – two 32-bit words. Word format in the count is shown in the section "Data and commands formats".

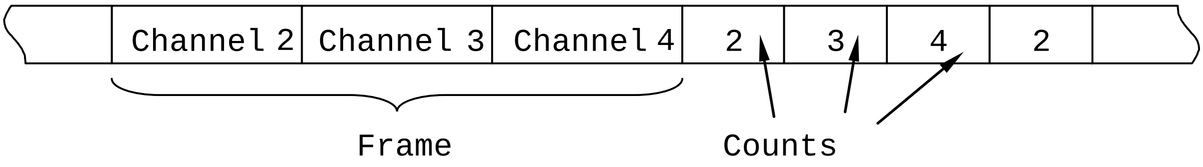

Figure Fig. 4.14. Sequence of the data to be received (channels 2 are switched on)

20-bit count

Transmitted as 32-bit word.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	-	-	-	-	-	-	0	C	N		D19	D18	D17	D16

D0 – D19

20-bit code of ADC.

N

Channel number.

C

Data counters. Set to 1 for every 15th word.

24-bit count

Count is transmitted by two sequentially located 32-bit words in the following order: HIGH, LOW.

HIGH:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	V	D23	D22	D21	D20	D19	D18	D17	D16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	-	-	-	-	-	-	1	0	N		C			

LOW:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	-	-	-	-	-	-	1	1	N		C			

D0 – D23

24-bit code of ADC.

N

Channel number.

C

Data count for the module 15 (count in a circle from 0 to 14). The value of the counter is the same for both count parts.

V

Sign of channel's input path overload.