Multichannel data-acquisition systems

LTR Crate System

Starting operating the LTR Crate System
Software issues

Revision 1.0.4
May 2017

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

DAQ SYSTEMS DESIGN, MANUFACTURING & DISTRIBUTION

Author of the manual:
Alexey Borisov

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

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

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

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

Table 1: Current document revisions

Revision	**Date**	**Description**
1.0.0	May 29, 2016	The document first revision
1.0.1	July 27, 2016	Information on FreeBSD and extra references to ltr_cross_sdk added where necessary.
1.0.2	August 04, 2016	A reference to section *C#*, wherein we can find information on classes used was made in the **LabView** chapter.
1.0.3	April 10, 2017	A reference to OPC-server added in the finalized software list.
1.0.4	May 19, 2017	Information on using the **NuGet** pack added for developing programs in *C/C++* for **Microsoft Visual Studio** (section 4.3).

Contents

1 What this document is about

Within this document we regard general issues of software (SW) providing references to more detailed documents. This document is intended to provide a general idea of the necessary and existing SW, both for the system end-users and software developers. Upon reading this document you can refer for more detailed data to document ltr_cross_sdk.pdf. Also, for successful working with LTR crates and modules read the LTR User Manual (http://en.lcard.ru/download/ltr.pdf).

2 Software provided

Normal operation of LTR crates is based on crate and module control and data processing and collection performed by upper level software which is run on a personal computer (PC). This software is specifically described in this document. Normally, crate software does not provide for autonomous crate operation (without a PC). If necessary, customized firmware can be created for autonomous operation of LTR-EU and LTR-CEU crates which is discussed in detail in section 4.9. The software provided by L-Card is free and in most cases open-source which means that source codes of application programs are provided (see section 4.8).

2.1 Which operating systems are compatible?

For operation of crate controlling software, a PC with one of compatible operating systems (OS) is required. The following OS are supported:

1. All Windows versions starting from Windows XP.
2. Modern Linux distributions. Some distributions come as assembled packages.
3. FreeBSD, starting from 9.3 or higher versions
4. QNX4, QNX6 real-time OS

   Software installation for Windows is described in section 2.2. Installation for other OS is described in ltr_cross_sdk.pdf.

   Note: In this case, system requirements are not specified as they are determined first of all by specific upper level software, data flows processed, etc. In the simplest case, system requirements correspond to the requirements of the OS used and, of course, necessary interface (USB or Ethernet) must be installed on the PC.

2.2 Which software must be installed for operating in Windows OS?

1. USB driver. This driver can be found in the lcomp library (http://en.lcard.ru/download/lcomp.exe), therefore, it is necessary to install this library for crate operation via USB.
2. Ltrd service (http://en.lcard.ru/download/ltrd-setup.exe). After the installation, this service starts automatically when the PC is activated and runs in the background

mode "invisibly" for the user. It is necessary for crate operation as all operational software runs through this service. The purposes for which it is used are described in section 2.4.

3. LTR Manager (http://en.lcard.ru/download/ltrmanager_setup.exe) is an auxiliary program with graphical interface that shows information about the connected crates and the statistics of their operation, displays the ltrd service log, allows the user to set-up the crates for operation via Ethernet, manage crate connections via Ethernet, update crate firmware, and perform other service actions.

4. Set of libraries for working with LTR modules (http://en.lcard.ru/download/ltrdll.exe). This set of libraries must be installed for programs which use the installed system version of libraries. Some programs (for example, LGraph2) install a local copy of these libraries during their installation, so you may not need to install libraries additionally for them. Also, files necessary for users to develop their programs on the PC are installed with the libraries.

Detailed description of this software (including ltrd, LTR Manager, and auxiliary service applications) can be found in ltr_cross_sdk.pdf.

Note: Earlier, the functions of ltrd and LTR Manager was performed by LtrServer application described in section 2.5. Currently, this application is not supported. Users of LtrServer are recommended to switch over to ltrd service which application interface is compatible with LtrServer, that means that applications working with LtrServer should work with ltrd too, unless they are using some rare functionalities of LtrServer. If you face any problems when switching over, contact the support team (see section 5)

2.3 What finalized software is provided for working with LTR modules?

L-Card provides the following finalized software options:

1. LGraph2 (http://en.lcard.ru/download/lgraph2.zip) is a logging visualizer application for installation on most of L-Card devices. It should be noted that the application supports not all LTR modules and not all of their functionalities. Detailed information about this application can be found in LGraph2 User Manual http://en.lcard.ru/download/lgraph2_help.pdf)

2. L-Card OPC-Server (http://en.lcard.ru/download/lcard-opc.pdf). The program allows using LTR crate system in SCADA systems and other software packages with OPC support.

3. UTS (http://en.lcard.ru/download/uts.zip) is a test program used for acknowledgment with the module and checking its functionalities. This program supports all possible software settings for most modules.

4. Specialized software applications for modules - for some modules, own specialized demo programs can be used instead of (or in addition to) UTS. For example, ltr210-osc (http://en.lcard.ru/download/ltr210-osc-install.exe) which is used for LTR210

module. You can always found the complete list of software applications in section "Software" on each module's page.

5. Metrological software for modules verification
   LTR (http://en.lcard.ru/download/ltr_metr_setup.exe). The metrological software includes multimeters for ADC modules and programs for setting standard signals for DAC modules.

6. At the client's order, L-Card can develop specialized finalized software for a specific task, for which the client may send a request to the service team's email (see the "Contacts" section on our web-site: http://en.lcard.ru/contacts)

2.4 What is the ltrd service needed?

In contrast to other L-Card modules and boards, one LTR crate can enclose up to 16 modules. In this case, data from all modules enclosed in the same crate are transmitted as a common stream via one communication channel. So that user programs can work independently with each module (you can work in different programs for different modules in the same crate), you need a program that parses the data stream from the crate to the modules and distributes them among the clients. This is the task which ltrd service performs. Therefore, ltrd establishes a connection with the crate, and the application software (client), in turn, establishes a connection with ltrd indicating the crate and the number of the slot in the crate in which the required module is located. Ltrd receives data from the crate, determines which module they correspond to, and sends them to the necessary clients and receives data from them, then combines the data into a common stream for transmission, and transfers the data stream to the crate.

Important! Ltrd service is necessary both for working with crate via USB or Ethernet.

Important! Despite that only one module can be enclosed in a single-unit crate, the protocol for working with this module is similar to the protocol used for multi-unit crates. Therefore, lrtd service must run even when working with a single-unit crate. A single-unit crate is still a crate but not an individual module!

Lrtd running as a background program provides the following additional benefits:

1. Since lrtd ensures connection with the crate, and the application software does not work directly with the crate, the application software works in the same way both with different types of crates and with crates connected via different interfaces. This means that software written when working with a crate of a particular type via specific interface can automatically work with other types of crates connected via different interfaces (if the user does not introduce forced software restrictions).

2. As the application software connects to ltrd through sockets, the application software (if it is written correspondingly) and ltrd can work, if desired, on different computers connected via a network. This possibility can be used, for example, for network control of crates connected to a remote computer via USB.

3. ltrd includes commands to control IP-addresses for crates connection via Ethernet.

4. ltrd provides for collection of statistics to monitor crate status.

5. ltrd maintains a log in which information on the service operation, change in the crate connection status, occurrence of errors during crate operation, etc. is recorded.

2.5 LtrServer and its difference as compared to ltrd

Prior to the launch of ltrd, LtrServer was used to perform the same functions but in the form of a user program with graphical interface (the functions of which were then transferred to LTR Manager), which had to be manually started to work with LTR crates.

Important! Currently, the program still can be used but it is no longer supported by L-Card. Users are recommended to switch over to ltrd.

Important! When using LtrServer for Windows 8 and higher versions, it is necessary to select the Windows XP compatibility mode in the program properties.

Below in this section, main differences between LtrServer and ltrd are described which can be useful for users who worked with LtrServer before.

ltrd was created as a cross platform (working not only with Windows, but already with Linux and QNX) to replace LtrServer. In addition to the possibility of running in different operating systems, there are the following key differences in the operation of these programs:

1. ltrd for Windows was originally developed as a service, which allows the user to start it with the system (which is set by the installation wizard), even in server systems without user authentification. This allows the user to work without starting any additional programs manually. LtrServer is realized as an ordinary graphical program for Windows, which can be minimized to a tray.

2. Ltrd itself does not provide a graphical interface. To implement the interface functionality (similar to LtrServer), LTR Manager is used which connects to ltrd and provides information about connected crates and the ability to control them, similar to LtrServer.

   For this, LTR Manager is not required to run for user software operation. There is also command line utility ltrctl for performing similar actions.

3. ltrd together with LTR Manager provide a convenient way to change the interface settings of a crate (USB or Ethernet, as well as to set-up the IP address for Ethernet) using a graphical interface. In addition, LTR Manager allows viewing current settings without changing them. When using LtrServer, to change the settings it is necessary to close the program, after which special command line configuration programs must be used which can be downloaded from the site at: http://en.lcard.ru/download/ltr030config.zip. To learn more about using these programs refer to readme.txt file contained in the said archive.

4. LtrServer does not recognize the crate operating mode when it is connected via USB but is configured to work via Ethernet. In such situation, depending on the program

version, the crate can be seen as empty without any express instructions or be not visible at all in LtrServer, while when working through ltrd and LTR Manager it is always visible in the "Only setup" mode and it is possible to change the settings of the crate.

5. ltrd together with LTR Manager provide more detailed statistics and information on connected crates than LtrServer.

6. ltrd does not support some rarely used functionalities of LtrServer, see more in ltr_cross_sdk.pdf.

3 Operating the crate via the Ethernet interface

LTR-EU and LTR-CEU crates can be connected not only via USB interface but also via Ethernet interface using TCP/IP protocol. Unlike USB, working via a network requires additional settings, that is addressed in this chapter.

3.1 Setting the crate configuration for operating via Ethernet

LTR crate can simultaneously work with modules only via one interface. Respectively, a crate is always set-up for working either via USB or via Ethernet. In this case, changing LTR-EU crate settings is always performed via USB interface (LTR-CEU crate allows changing the settings via Ethernet).

If a crate is set-up for working via Ethernet, it still will work as a USB device too, however, only interface set-up and firmware update functions are available via USB, and working with modules is impossible. In this case, this crate will be visible in LTR Manager without modules, and its operating mode indicated in the crate settings will be "Setup Only".

If the crate is at the same time connected via Ethernet too, the crate will be displayed twice with one entry corresponding to the operating Ethernet connection and the other to the USB connection used for setting-up.

Crate settings can be changed in LTR Manager. For this, the user just has to right-click on the crate and select the "Crate Settings" menu item (this item is only available for entries corresponding to LTR-EU crates connected via USB or to LTR-CEU crates). A window will appear displaying the current crate settings which can be changed to the required ones.

For Ethernet interface configuration described above, the following parameters must be specified which depend on the network to which the crate is connected:

- Crate IP address which is 4 digits from 0 to 255 (for example, 192.168.1.2). The IP address selection procedure is described in section 3.2.
- Subnet mask. Subnet mask determines which part of IP address is attributed to the network and which one is the address within the network. For local networks, the most commonly used mask is 255.255.255.0 in which the first 3 digits are for the network address and the last digit is for the device address within the network.
- Gateway IP address. Not used if the crate is connected to the local network (crate and PC in the same network). Working with a crate via Internet is a separate issue requiring consideration of a large number of cases which is not addressed in this document.

Important! New settings become effective only after the crate reboot. LTR-CEU crates support the program reboot command (which can be executed at once by LTR Manager). Reboot of LTR-EU crates is performed manually by crate de-energizing or using the reset button.

For LTR-CEU crates, changing settings can also be protected with a simple password. If the password is not set, then when you change the settings, you can enter nothing in the request and immediately click "Ok". This is done so that you can protect the crate from changing settings remotely, since LTR-CEU supports changing them not only via USB. In this case, if the password is forgotten, you can change the settings via the USB interface by entering the serial number of the crate as the password.

3.2 Choosing an IP-address for the crate

When assigning an IP address, the following requirements must be considered:

- It must be unique within the network (i.e. there should not be another device with the same address, in particular, the crate address and the PC address must be different).
- It must belong to the same network as the address of the PC on which ltrd service is running (i.e. the beginning digits corresponding to the network address must coincide).

When connecting to a ready network, it must be taken into account that the crate does not support protocols for automatic address assigning (DHCP, link-local). Accordingly, if IP addresses are assigned automatically in your network, the network administrator must allocate a certain range of addresses for the manual (static) assignment and assign an address from this range to the crate.

For direct connecting to a PC, you must set-up both the crate address and the PC address (or the address of a specific PC interface) to which the crate is connected. As mentioned above, the crate and PC addresses must be different but being within the same network. For example, if the crate address is 192.168.1.2, then the PC address can be 192.168.1.1 (the mask will be 255.255.255.0 both for the PC and the crate).

3.3 Establishing a connection with the crate

Unlike crates connected via USB, ltrd does not establish connection with crates connected via Ethernet without manual setting-up. Below are described the general actions to perform in LTR Manager for crate connection. For more details refer to the manual ltr_cross_sdk.pdf. To establish connection, perform the following actions:

- Add the crate IP address to the address list of LTR Manager.
- Double click on the added entry (or right-click on the record in the context menu) to start the procedure for establishing a network connection with the crate (the record status must change to "Connecting ...").
- When the connection is complete, the status will change to "Connected", and the crate will appear in the list of connected crates.
- After that, the crate can be operated in the same way as a crate connected via a USB interface.

Note: If the crate IP address entry added is set to "Auto", then when ltrd is started it will try to connect to this crate automatically at this address.

4 Developing user software

4.1 Developing PC software

L-Card provides a set of *C-based* libraries for developing user software (.dll for Windows or .so for Linux or FreeBSD). For each module, an individual ltrXXXapi library is provided (where XXX is the module number) with a particular set of functions. A special ltrapi library is provided for controlling ltrd and crates operation.

For each library, own programmer's manual (ltrXXXapi.pdf) is provided which can be downloaded from the site in the "Documentation" section on each module page.

In addition to *C-based* programs, L-Card also provides wrappers for working with modules in other languages and environments. At the moment, *Delphi*, *C#* and *LabView*are supported.

Programming examples can be downloaded from the page "Software for Developer". Some of them are presented with the source codes: ltr_cross_sdk (https://bitbucket.org/lcard/ltr_cross_sdk/downloads/ltr_cross_sdk_src.zip).

4.2 Using libraries when writing programs in C/C++

To work with LTR modules in your program, you need to connect the ltrXXXapi dynamic library (where XXX is the module number) to the project for each module as well as the general ltrapi library if its functions are used.

If a program is written in *C/C++* in Microsoft Visual Studio, NuGet package can be used as described in section 4.3 which performs all actions to connect the libraries automatically.

When a different environment (or no environment) is used, the following actions must be performed to connect a library:

- Run the header file ltr/include/ltrXXXapi.h for each module used, in which all constants, types and functions available for working with this module are defined.
- For OS Windows:
  - Make sure that all the necessary files ltrXXXapi.dll and ltrapi.dll are installed either into the directory with the program or into the directory of the PATH environment variable (the installer sets % WINDIR%/system32).
  - Connect files ltrXXXapi.lib or libltrXXXapi.a to the project (and also ltrapi.lib or libltrapi.a, if ltrapi functions are used) for the necessary compiler:
    * *Microsoft Visual C++* 32-bit compiler — from ltr/lib/msvc
    * *Microsoft Visual C++* 64-bit compiler — from ltr/lib/msvc64
    * *Borland C++/Borland C++ Builder/Embarcadero C++ Builder* 32-bit compiler — from ltr/lib/borland
    * *Embarcadero C++ Builder* 64-bit compiler — from ltr/lib/borland64

    * *MinGW* 32-bit compiler — from ltr/lib/mingw
    * *MinGW* 64-bit compiler — from ltr/lib/mingw64

- For OS Linux
  - Make sure that libltrXXXapi.so, libltrXXXapi.so.* (where * is the version), libltrapi.so, libltrapi.so.* files are installed either into one of the system directories for the libraries, or into the directory specified through the LD_LIBRARY_PATH environment variable or otherwise (assembled packages install libraries into the standard directory /usr/lib)
  - Connect the library to the project. When GCC is called from the command line, this can be done with lltrXXXapi key.

4.3 Use of NuGet package when writing programs in C/C++ for Microsoft Visual Studio

NuGet (https://www.nuget.org) is a package manager for Microsoft Visual Studio environment. Starting from Visual Studio 2012, it comes pre-installed with the environment by default. For using ltrapi libraries in a project in *C/C++* , lcard.ltr.ltrapi package (https://www.nuget.org/packages/lcard.ltr.ltrapi) is available in the NuGet repository.

When using NuGet package, it is no need to install ltrdll with the installer as all necessary files are included in the package.

When this package is included into the project, all necessary files (.h, .lib, .dll) will automatically be downloaded to the project, and the project will be configured so that the necessary paths to the header files will be included, the necessary .lib files (32-bit or 64-bit depending on the configuration used) will be connected, and all .dll files of the desired bit depth will be copied to the output directory of the project.

The user has only to include the necessary header files (ltr/include/ltrXXXapi.h) where all definitions, types and functions are declared.

NuGet also allows the user to track the release of new package versions and update them.

In the latest versions of Visual Studio all these functions are built in the environment and performed from it. For example, to connect a package in Visual Studio 2015, the user has only to right-click on the project in the "Solution Explorer" and select the "NuGet Package Management...” in the menu. In the window that opens the user has to select the overview and enter ltrapi in the search field, then select lcard.ltr.ltrapi package and click "Install".

4.4 Using libraries when writing programs in Delphi

For writing programs in *Delphi*, L-Card provides a set of .pas files in which the same structures and functions are declared which are used in *C-based* libraries. The user must enable the following modules through "uses" in the program: ltrapi, ltrapitypes, ltrapidefine and ltrXXXapi for each used module. In this case, the program will use the same .dll files as a *C-based* program. There are two versions of .pas files:

- In ltr/include/pascal, the old version of the files is installed which are left only for compatibility with old projects. These files require a manual alignment of

structures in the environment; these files can not be used in modern *Delphi versions* and can contain some inaccuracies.

- In ltr/include/pascal2, the new versions of the files is installed. These files can be used both in *Delphi 7*and in modern versions of *Embarcadero Delphi XE* (32-bit and 64-bit applications). These files have little incompatibilities with the old version (for example, AnsiChar but not Char type is used for characters, since in modern versions Char is a unicode character occupying two bytes, not one). This version requires only small changes and is recommended for use.

It should be noted that there is also a change in the description of functions for old modules (with preservation of old versions). For more correspondence to the language, the following changes are made:

- Use of typed variables "var" or "out" instead of pointers.
- Use of "string" type (a standard type for strings in *Delphi*) for passing strings to release the user of manual conversions.
- Use of open array parameter for passing arrays which means that both static and dynamic (with length pre-set with the use of *SetLength()*) arrays can be passed in these functions.

The new modules contain descriptions accounting for the above changes. For old modules, the new version of the functions will be added as possible, and examples will be updated.

4.5 Using libraries when writing programs in C#

For writing programs in *C#* , special wrapper library ltrModulesNet.dll is provided (installed in ltr/bin/ltrModulesNet) which must be connected to the created project. It uses functions from .dll in *C*. This means that both ltrModulesNet.dll and other used ltrXXXapi.dll libraries are necessary for program functioning.

All classes in this library are declared within ltrModulesNet space (can be enabled through using ltrModulesNet).

All functions, structures and constants are combined within the corresponding classes:

- Definitions of modules for the new api version are contained in ltrXXXapi classes (without underscore). These classes are intended for using both in *C#-based* programs and in *LabView* programs. Access to the descriptor fields is realized there through the properties of the class. For some functions, a version with more convenient types of parameters is implemented. Currently, these classes are available not for all modules.
- ltrcrate class is separated for access to ltrapi functions attributed to the same crate.
- ltrsrvcon class is separated for access to ltrapi functions attributed to the control connection with ltrd.
- All definitions from ltrapi are contained in _LTRNative class.
- For the old api version, two classes are used: _ltrXXXapi and a separate class ltrXXXapiLabView intended for using in *LabView*the only difference of which is that all fields of the module descriptor are available through that class members. Some parameters are described in a way requiring manual conversion. It is recommended to use the new ltrXXXapi classes (if any) if they exist for the given module.

There are two options of working with functions:

- Through static class methods. They exactly replicate the prototype and name of *C-based* functions and also take the structure corresponding to the module descriptor as the first parameter.
- Through the own class methods. In this case, an object of this class is created that contains the module descriptor structure inside it. Accordingly, this structure is not passed to class methods (since all fields of the class instance are available in the method). Names of the methods are similar to the *C-based* functions but without LTRXXX_ prefix (as it is clear by the class to which this module belongs, to which module the function belongs). The module descriptor fields must change either within the structure inside the class (for underscore classes) or through properties (for new classes without underscore). For new classes, this option is more preferable, since it allows the user to make additional necessary conversions to standard *C#* types (for example, for strings or arrays of structures).

A new version of classes with examples of their use is available only for a limited set of modules. The list of such modules will be extended as possible.

4.6 Using libraries when writing programs in LabView

As *LabView* supports working with controlled libraries NetFramework, ltrModulesNet.dll library can be used for writing programs in *LabView*.

Descriptions of classes in ltrModulesNet.dll which can be used in *LabView* are presented in section 4.5.

Important! Only 8.0 and higher *LabView* versions are supported as the previous versions had significant productivity problems despite the support of NetFramework libraries.

There is special Connectivity -> .Net panel for working with .Net classes in *LabView* . For this, the following blocks must be used:

- Constructor Node — creates an object. It must be created for each LTR module that will work as well as for each control connection to the crate or ltrd. During the creation, *LabView* will offer to select a library and a class (ltrModulesNet.dll and necessary class must be selected: ltrXXXapi for the given module (if available) or ltrXXXapiLabView (if not available). One of the outputs of this block is a reference to an object that is used as an input for other blocks for working with the module. Also, with the use of the designer tool, objects of the structures necessary for working with the module are created.
- Close Reference — closes and deletes the object. It must be called for each created object when the operation is complete.
- Invoke Node — function (class method) call. When working with an object, a reference and input parameters are fed to the input, and output parameters and an updated reference (which must be used for blocks that will be called after the current one) are fed to the output. The input reference also determines which object's methods are used (after the entry link is placed, the class name appears on the top line, and when you click on the second one, its method will be suggested for selection). For functions which do not work with

a particular object (these functions are static and are marked with [S] at the beginning), the user does not need to provide a reference to the input. However, they still belong to the class that must be selected by right-clicking on the block and then Select Class/ .Net.

- Property Node — used for installation and obtaining the object properties. Through the properties, structure fields (for example, module settings or module information) are set or read. The user can set several properties in one block by expanding it downward. Also, using properties, the user can specify constants from enumerations (which can be more understandable than simply giving the input numbers). In this case, you must select the enumeration class, and each value will have its own property.

There is some specifics about passing arrays as output parameters. For effectiveness reasons, the library functions do not allocate data arrays within themselves, but use the transferred arrays to store the results. Therefore, such parameters are at the same time both input and output parameters in *LabView* . At the input, the user must send an array of a size sufficient to store the results (while the data itself does not matter) and at the output the same array will be returned already containing the results of the function execution.

4.7 Creating 64-bit programs in Windows software

In 64-bit Windows OS version, programs can be executed both compiled by a 32-bit and a 64-bit compiler (the latter are called native 64-bit applications), therefore, many programs for Windows exist only in 32-bit version. The 64-bit compiler is generally used for programs that work with large bulk of data since the 64-bit program uses more than 4 GB virtual space.

As this feature can be very useful for data collection systems processing large bulk of data, the ltrdll.exe installer, starting from version 1.28.0, installs additional 64-bit versions of the libraries, i.e. provides the possibility to create native 64-bit applications. At the same time, the files location is changed as compared to versions 1.27.x and lower (for more details, see the readme.txt file among the installed files).

Important!: For 64-bit libraries installation, the program for creating the installer was changed. Therefore, if version 1.27.x or lower was installed before the installation, you must first delete it and then install the new version.

When installed on 64-bit Windows version, the ltrdll installer places both 32-bit and 64-bit libraries versions into the corresponding system directories. In this case, Windows/system32 directory refers to one of these directories, depending on the bit depth of the application itself which refers to the specified path. For 32-bit application, 32-bit libraries are stored in Windows/system32, 64-bit libraries in Windows/Sysnative, and for 64-bit application, 64-bit libraries are stored in Windows/system32, and Windows/Sysnative does not exist. At the same time, 32-bit libraries are always stored in Windows/SysWOW64 which always exists irrespectively of the bit depth of the application.

When the application is downloaded, if system libraries are used they are searched for using paths from the PATH environment variable, among which there is Windows/system32. Since this directory refers to different locations depending on the bit depth of the application being launched, the library of the required bit depth is selected

from Windows/system32 automatically. If the libraries are distributed with the program, it must be ensured that the bit depth of the assembled application and the libraries in the same directory be the same.

The only difference when writing programs in *C/C++* is the need to attach a lib-file (or .a) in accordance with the bit depth of the compiler used.

For programs in *Delphi* , you only need to specify for which platform the project will be assembled (win32 or win64), and the assembled program will use the library of that bit depth for which the program was compiled.

Programs in *C#* (or any other using NetFramework) are compiled into machine code when executed. Once created, the program can be executed both in 32-bit version and 64-bit version of the NetFramework virtual machine (in the project, if necessary, it can be specified explicitly for what bit depth of NetFramework the program is intended for). Therefore, if the bit depth is not explicitly specified, the same program in 32-bit Windows version will be executed using 32-bit version of the libraries, and in 64-bit version using 64-bit version. For ltrModulesNet.dll library, the bit depth is determined by the bit depth of the application that uses the library.

Accordingly, in a *LabView*project using the .Net library ltrModulesNet.dll, the bit depth of the library used is determined by the bit depth of the used *LabView*environment.

4.8 Where do we get the source codes for PC software

L-Card provides source codes for most of its programs. For some projects, access to open source repositories of the version control system on bitbucket is provided (for more details, see http://en.lcard.ru/download/lcard_bitbucket_repos.pdf).

- Source codes for ltrapi libraries, ltrd service (daemon), LTR Manager and all other utilities included in ltr_cross_sdk can be downloaded from the project page on bitbucket: https://bitbucket.org/lcard/ltr_cross_sdk. An archive containing all source codes can be downloaded using the link: https://bitbucket.org/lcard/ltr_cross_sdk/ downloads/ltr_cross_sdk_src.zip.

4.9 Creating a user version of LTR-EU crates firmware

For some tasks, the user can be unsatisfied with the crate firmware. As an example, the following reasons can be listed:

- A need to have a stand-alone device that starts the collection and executes processing without control from the PC
- A need to ensure minimum and certain delays from changing signals at the system inputs to the response at its outputs, for which it is necessary to make decisions within the crate
- A need to implement own protocols for exchange with the PC
- A need to control other devices over the network or RS-485 directly from the crate
- etc.

For such cases, the user is provided with an opportunity to change the firmware of Blackfin signal processor installed in LTR-EU crates.

Also, the user can order that L-Card develops such firmware. For this, just contact the support team.

Important! It should be noted that the development of firmware for built-in systems is very different from the development of PC software and requires appropriate skills.

Important! At the moment, L-Card does not provide a manual on modification of crate processor firmware, so the programmer has to receive necessary information by analyzing the provided source firmware codes. At the same time, our support team is ready to answer any general questions.

There are two versions of LTR-EU crates firmware for which L-Card provides source codes to the user for further modification:

1. FreeRTOS-based firmware with the use of TCP/IP lwip stack. This firmware is assembled with the use of VisualDSP environment provided by Analog Devices. Source codes of this firmware can be downloaded from the web-site of L-Card at: http://en.lcard.ru/download/ltr_source_1_0_0_1.zip. This firmware is finalized and will not be updated in the future.
2. Currently, RTEMS-based firmware is being developed with the use of TCP/IP stack from this RTOS transferred from FreeBSD. This version is developed to solve some problems of the old version when working via TCP/IP and provide a convenient API for users to implement data processing within the crate (similar to API for PC). This firmware is under development and not all crate features are implemented. API functions are implemented only for a limited number of modules. In case of any questions related to this firmware, contact the support team (see the "Contacts" section on our web-site: http://en.lcard.ru/contacts).

5 In case problems arise

If you face any problems while working with crates and LTR modules, do the following:

1. Make sure that you have read and understood: this document, necessary sections of the User Manual, documentation for the used software (if available) or the relevant programmer's guides when writing your own programs.
2. Check if the latest SW versions are installed. The latest versions can be downloaded from L-Card site in the "Software" section related to the needed module.
3. If you use your own software or software developed by any companies other than L-Card, test the module operation using software provided by L-Card (see section 2.3).
4. If the problem is still unsolved, describe it in detail (!), having created a topic in the support forum, or send a request to the support team's email (see the "Contacts" section on the site: http://en.lcard.ru/contacts).