

# Начиная работать с крейтовой системой LTR. Вопросы по программному обеспечению.

Борисов Алексей

Август 2014 г.

## Содержание

<b>1</b>	<b>О чем этот документ</b>	<b>1</b>
<b>2</b>	<b>Предоставляемое программное обеспечение</b>	<b>1</b>
2.1	Какие операционные системы поддерживаются	2
2.2	Какое ПО необходимо установить для работы под ОС Windows	2
2.3	Какое существует законченное ПО для работы с модулями LTR	2
2.4	Для чего нужен ltrd/LtrServer	3
2.5	Разница между ltrd и LtrServer	3
<b>3</b>	<b>Работа с крейтом по интерфейсу Ethernet</b>	<b>4</b>
3.1	Настройка крейта на работу по Ethernet	4
3.2	Выбор IP-адреса для крейта	5
3.3	Установление соединения с крейтом	5
<b>4</b>	<b>Разработка пользовательского программного обеспечения</b>	<b>5</b>
4.1	Разработка ПО для ПК	5
4.2	Использование библиотек при написании программ на языках C/C++	6
4.3	Использование библиотек при написании программ на Delphi	6
4.4	Использование библиотек при написании программ на языке C#	7
4.5	Использование библиотек при написании программ на LabView	7
4.6	Создание 64-битных программ под ОС Windows	8
4.7	Где взять исходные коды ПО для ПК	9
4.8	Создание пользовательской версии прошивки крейтов LTR-EU	9
<b>5</b>	<b>При возникновении проблем</b>	<b>10</b>

## 1 О чем этот документ

В данном документе рассматриваются общие вопросы по программному обеспечению (ПО) с ссылками на более подробные документы. Этот документ предназначен, чтобы дать общее представление о необходимом и существующем ПО как для конечных пользователей системы, так и для программистов. Также для успешной работы с крейтами и модулями LTR необходимо изучить руководство пользователя LTR (<http://www.lcard.ru/download/ltr.pdf>).

## 2 Предоставляемое программное обеспечение

Штатная работа крейтов LTR предполагает, что управление крейтами, модулями, сбором и обработкой данных осуществляет ПО верхнего уровня, запущенное на персональном компьютере (ПК). О данном ПО главным образом и пойдет речь в данном документе. Штатный вариант ПО крейта не предполагает автономной работы крейта (без ПК), однако при желании прошивка процессора крейтов LTR-EU может быть изменена пользователем для решения подобных задач, о чем отдельно рассказано в [разделе 4.8](#). ПО, предоставляемое “Л Кард”, бесплатное и в большинстве случаев полностью открытое, т.е. предоставляются исходные коды программ (см. [раздел 4.7](#)).

## 2.1 Какие операционные системы поддерживаются

Для работы ПО, управляющего крейтами, необходим ПК с одной из поддерживаемых операционных систем (ОС). Поддерживаются следующие ОС:

1. Все версии Windows, начиная с Windows XP.

**Важно!:** При использовании программы **LtrServer** для Windows 8 и выше необходимо установить в свойствах программы режим совместимости с Windows XP

2. Современные дистрибутивы Linux. Для некоторых дистрибутивов распространяются собранные пакеты (подробнее см. в [ltr\\_cross\\_sdk.pdf](#))
3. ОС реального времени QNX4, QNX6

**Примечание:** Системные требования в данном случае не указаны, так как они определяются в первую самим ПО верхнего уровня, обрабатываемыми потоками данных и т.п. В простейшем случае системные требования соответствуют требованиям используемой ОС, а также конечно необходимо наличие у ПК требуемого интерфейса (USB или Ethernet).

## 2.2 Какое ПО необходимо установить для работы под ОС Windows

1. Драйвер USB. Драйвер входит в состав библиотеки **lcomp** (<http://www.lcard.ru/download/lcomp.exe>), поэтому для работы с крейтами по USB необходимо установить данную библиотеку.
2. Служба (service) **ltrd** ([https://bitbucket.org/lcard/ltr\\_cross\\_sdk/downloads](https://bitbucket.org/lcard/ltr_cross_sdk/downloads)) или программа **LtrServer** ([http://www.lcard.ru/download/ltrserver\\_setup.exe](http://www.lcard.ru/download/ltrserver_setup.exe)). Одно из этих приложений должно быть запущено во время работы с крейтами, так как все штатное ПО (за исключением некоторых сервисных старых программ) работает через данные программы. О том, для чего нужны данные программы и какая между ними разница, описано в [разделе 2.4](#) и [разделе 2.5](#) соответственно. При работе через **ltrd** также может быть очень полезна и программа **LtrManager**.
3. Набор библиотек для работы с модулями LTR (<http://www.lcard.ru/download/ltrdll.exe>). Установка этого набора библиотек необходима для программ, которые используют установленную системную версию библиотек. Часть программ (например **LGraph2**) во время установки устанавливают локальную копию данных библиотек, поэтому для них отдельно устанавливать библиотеки может быть не нужно. Также с библиотеками устанавливаются файлы, необходимые для разработки пользователями своих программ.

## 2.3 Какое существует законченное ПО для работы с модулями LTR

“Л Кард” предоставляет следующие варианты законченного ПО (все ПО предоставляется бесплатно и доступно для скачивания с сайта):

1. **LGraph2** (<http://www.lcard.ru/download/lgraph2.zip>) — программа самописец-визуализатор для большинства устройств “Л Кард”. Следует учитывать, что программа поддерживает не все модули LTR и может поддерживать не все их возможности. Подробную информацию о данной программе можно найти в руководстве пользователя для **LGraph2** ([http://www.lcard.ru/download/lgraph2\\_help.pdf](http://www.lcard.ru/download/lgraph2_help.pdf))
2. Программа **UTS** (<http://www.lcard.ru/download/uts.zip>) — тестовая программа, предназначена для ознакомления и проверки возможностей модуля. Для большинства модулей поддерживает все возможные программные настройки.
3. Специализированные программы для модулей — для некоторых модулей вместо (или в дополнение) **UTS** могут использоваться свои специализированные демонстрационные программы. Например, **ltr210-osc** (<http://www.lcard.ru/download/ltr210-osc-install.exe>) для модуля LTR210. Полный список ПО можно всегда посмотреть в разделе “Программное обеспечение” на странице каждого модуля.

**Примечание:** Платное ПО для работы с LTR предоставляют наши партнеры: ACTest (<http://www.actech.ru>) и Powergraph ([www.powergraph.ru](http://www.powergraph.ru))

## 2.4 Для чего нужен ltrd/LtrServer

В отличие от других модулей и плат “Л Кард” , в одном крейте LTR может находиться до 16 модулей. При этом данные от всех модулей одного крейта передаются общим потоком по одному каналу связи. Для того, чтобы пользовательские программы могли работать с каждым модулем независимо (с разными модулями в одном крейте можно работать из разных программ), необходима программа, которая бы разбирала поток данных от крейта. Именно для этой задачи и созданы **LtrServer** и **ltrd**. Таким образом, **ltrd/LtrServer** устанавливает связь с самим крейтом, а прикладное ПО (клиент) устанавливает связь уже с **ltrd/LtrServer**, указывая крейт и номер слота в крейте, в котором находится модуль, с которым необходимо работать. **ltrd/LtrServer** принимает данные от крейта и передает нужные данные нужным клиентам, а также принимает от клиентов данные, объединяет в общий поток для передачи и передает в крейт.

**Важно!** **ltrd/LtrServer** необходимы как при работе с крейтом по интерфейсу USB, так и Ethernet.

**Важно!** Несмотря на то, что в одноместном крейте может быть только один модуль, протокол работы с этим модулем аналогичен протоколу для многоместных крейтов. Поэтому даже при работе с одноместным крейтом все равно необходим запущенный экземпляр **ltrd/LtrServer**. Одноместный крейт это все равно крейт, а не отдельный модуль!

Дополнительно введение фоновой программы **ltrd/LtrServer** дает следующие преимущества:

1. Так как связь с крейтом обеспечивает **ltrd/LtrServer**, а прикладное ПО не работает с крейтом напрямую, то прикладное ПО работает одинаково как с разными типами крейтов, так и с крейтами, подключенными по разным интерфейсам. Т.е. ПО, которое было написано при работе с определенным типом крейта по определенному интерфейсу, автоматически может работать и с другими типами крейтов, подключенными по другому интерфейсу (если не делать искусственных ограничений в ПО).
2. Так как соединение с **ltrd/LtrServer** прикладным ПО выполняется через сокет, то при желании (если ПО написано соответствующим образом) прикладное ПО и **ltrd/LtrServer** могут работать на разных машинах, соединенных по сети. Это можно, например, использовать для управления по сети крейтами, подключенными по USB к удаленной машине.
3. В **ltrd/LtrServer** включены команды управления IP-адресами для подключения крейтов по Ethernet.
4. В **ltrd/LtrServer** реализован сбор статистики для мониторинга за состоянием крейтов.

## 2.5 Разница между ltrd и LtrServer

**ltrd** был создан как кроссплатформенная (работающая не только под Windows, но также на текущий момент и под Linux и QNX) замена программы **LtrServer**. Помимо возможности запуска под разными операционными системами, существуют следующие ключевые различия в работе этих программ:

1. **ltrd** под Windows изначально создан как служба, что позволяет запускать его вместе с системой (на что и настраивает систему установщик), причем даже на серверных системах без входа пользователя. Это позволяет работать без ручного запуска каких-либо дополнительных программ. **LtrServer** сделан как обычная графическая программа Windows, которая может сворачиваться в трей. При необходимости ее тоже можно запустить как службу (см. архив [http://www.lcard.ru/download/ltrserver\\_as\\_service.zip](http://www.lcard.ru/download/ltrserver_as_service.zip)), однако то, что графический интерфейс не отделен от службы накладывает свои ограничения.
2. Сама программа **ltrd** не предоставляет графический интерфейс. Для реализации возможностей интерфейса (аналогичных **LtrServer**) используются программа **LtrManager**, которая подключается к **ltrd** и предоставляет информацию о подключенных крейтах и возможность управлять ими, аналогично **LtrServer**. При этом для работы пользовательского ПО запуск **LtrManager** не требуется. Есть также утилита командной строки **ltrctl** для выполнения сходных действий.
3. **ltrd** вместе с **LtrManager** предоставляют удобный способ смены настроек интерфейса крейта (USB или Ethernet, а также настройка IP-адреса для Ethernet) через графический интерфейс. При этом **LtrManager** позволяет также посмотреть текущие настройки, не изменяя их.

4. **ltrd** вместе с **LtrManager** предоставляют более подробную статистику и информацию по подключенным крейтам, нежели **LtrServer**.
5. **ltrd** не поддерживает часть редко используемых возможностей **LtrServer**, подробнее см. в документе [ltr\\_cross\\_sdk.pdf](#).

**Примечание:** В дальнейшем возможна реализация новых возможностей в **ltrd**, которые не будут поддержаны в **LtrServer**

## 3 Работа с крейтом по интерфейсу Ethernet

Крейты LTR-EU предоставляют возможность подключения не только по интерфейсу USB, но и по интерфейсу Ethernet с использованием протокола TCP/IP. В отличие от USB работа по сети требует дополнительных настроек, чему посвящена данная глава.

### 3.1 Настройка крейта на работу по Ethernet

Одновременно крейт LTR позволяет работать со вставленными модулями только по одному интерфейсу. Соответственно, крейт всегда настроен на работу либо по USB, либо по Ethernet. При этом изменение настроек крейта всегда выполняется по USB-интерфейсу.

Если крейт настроен на работу по Ethernet, то он все равно видится и как USB-устройство, однако по USB доступны только функции по настройке интерфейса, но невозможна работа с модулями. В таком случае при работе через **ltrd** данный крейт будет виден в **LtrManager** без модулей с указанием режимом работы: “Только настройка”. Если при этом связь с крейтом установлена по Ethernet, то крейт будет отображаться два раза — одна запись соответствует рабочему подключению по Ethernet, а другая — подключению по USB для настройки. При такой ситуации в случае работы через **LtrServer** в зависимости от версии программы крейт может быть виден как пустой, либо вообще не быть видимым в **LtrServer**.

Настройка крейта осуществляется следующим образом:

- При работе через **ltrd** настройки можно изменить напрямую через **LtrManager**. Для этого достаточно нажать правой кнопкой мыши по крейту и выбрать пункт меню “Настройки крейта” (пункт доступен только для записей, соответствующих крейтам LTR-EU, подключенным по USB). Появится окно, в котором будут отображены текущие настройки крейта, которые можно изменить на требуемые.
- При работе через **LtrServer**, в связи с тем, что сама программа **LtrServer** такие возможности не предоставляет, для смены настроек необходимо выйти из программы. После чего необходимо воспользоваться специальными программами командной строки для настройки, которые можно скачать с сайта по адресу: <http://lcard.ru/download/ltr030config.zip>. Подробнее об использовании этих программ написано в **readme.txt** из указанного архива.

При описанной выше настройке интерфейса Ethernet должны быть указаны следующие параметры, которые зависят от сети в которую подключен крейт:

- IP-адрес крейта. 4 числа от 0 до 255 (например, 192.168.1.2). Выбор адреса описан в [разделе 3.2](#).
- Маска подсети. Определяет, какая часть IP-адреса отвечает за сеть, а какая за адрес внутри сети. Для локальных сетей чаще всего используется маска 255.255.255.0, что означает, что первые 3 цифры отвечают за адрес сети, а последняя — за адрес устройства внутри сети.
- IP-адрес шлюза (gateway). Если крейт подключен в локальную сеть (крейт и ПК в одной сети), то не используется. Вопрос работы с крейтом через Интернет — отдельный вопрос, требующий рассмотрения большого количества случаев, и в данном документе не рассматривается.

**Важно!:** Новые настройки вступают в силу только после перезагрузки крейта и сохраняются до следующего их изменения.

## 3.2 Выбор IP-адреса для крейта

При назначении IP-адреса нужно учитывать следующие требования к нему:

- Он должен быть уникальный внутри сети (т.е. не должно быть другого устройства с таким же адресом, в частности адрес крейта и ПК должны быть разные).
- Он должен принадлежать той же сети, что и адрес ПК, на котором запущен **ltrd/LtrServer** (Т.е. числа в начале, соответствующие адресу сети, должны совпадать).

При подключении в уже готовую сеть надо учитывать, что крейт не поддерживает протоколы для автоматического получения адреса (DHCP, link-local). Соответственно, если в Вашей сети адреса назначаются автоматически, то необходимо, чтобы администратор сети выделил некоторый диапазон адресов для ручного (статического) назначения и назначить крейту адрес из данного диапазона.

При подключении к ПК напрямую необходимо настроить как адрес крейта, так и адрес ПК (или конкретного интерфейса ПК), к которому подключен крейт. Как упоминалось выше, адрес крейта и ПК должны быть разные, но внутри одной сети. Например, если адрес крейта 192.168.1.2, то адрес ПК может быть 192.168.1.1 (маска и у ПК, и у крейта — 255.255.255.0).

## 3.3 Установление соединения с крейтом

В отличие от крейтов, подключенных по USB, **ltrd/LtrServer** не устанавливает связь с крейтами в сети без ручной настройки. Ниже описаны общие шаги, которые нужны для подключения при работе через **LtrServer** или **ltrd** с **LtrManager**. Более подробно Вы можете прочитать в руководстве по **LtrServer** ([ltrserver.pdf](#) из архива [http://www.lcard.ru/download/ltr\\_sw.zip](http://www.lcard.ru/download/ltr_sw.zip)) или в [ltr\\_cross\\_sdk.pdf](#) соответственно.

Для установления связи необходимо:

- Добавить IP-адрес крейта в список адресов в программе **LtrServer** или **LtrManager**.
- Двойным нажатием на добавленной записи (или через контекстное меню по нажатию на записи правой кнопкой мыши) запустить процедуру установления соединения с крейтом по сети (статус записи должен измениться на “Подключение...”).
- При завершении подключения статус должен измениться на “Подключен”, а крейт появиться в списке подключенных крейтов.
- После этого с крейтом можно работать также, как с крейтом, подключенным по интерфейсу USB.

**Примечание:** Если у добавленной записи с IP-адресом крейта установить флажок “Авто”, то при запуске **ltrd/LtrServer** данные программы попытаются автоматически установить соединения с крейтом по данному адресу.

# 4 Разработка пользовательского программного обеспечения

## 4.1 Разработка ПО для ПК

Для разработки пользовательского ПО верхнего уровня “Л Кард” предоставляет набор библиотек, написанных на языке *C* (.dll для Windows или .so для Linux). Для каждого модуля предоставляется своя библиотека **ltrXXXapi** (где XXX — номер модуля) со своим набором функций. Для управления работой **ltrd/LtrServer**, а также для управления работой крейтов, предоставляется специальная библиотека **ltrapi**.

Для каждой библиотеки предоставляется свое руководство программиста (**ltrXXXapi.pdf**), которое можно скачать с сайта в разделе “Документация” на странице каждого модуля.

Помимо использования в программах на *C*, “Л Кард” предоставляет также “обертки” для возможности работы с модулями с использованием других языков и сред. На данный момент поддерживаются *Delphi*, *C#* и *LabView*.

Примеры программирования пользователь может скачать со странички “ПО для разработчика”, а также часть примеров идет вместе с исходными кодами **ltr\_cross\_sdk** ([https://bitbucket.org/lcard/ltr\\_cross\\_sdk/downloads/ltr\\_cross\\_sdk\\_src.zip](https://bitbucket.org/lcard/ltr_cross_sdk/downloads/ltr_cross_sdk_src.zip)).

## 4.2 Использование библиотек при написании программ на языках C/C++

Для работы с модулями LTR из своей программы необходимо подключить к проекту динамическую библиотеку **ltrXXXapi** (где XXX - номер модуля) для каждого модуля, а также общую библиотеку **ltrapi**, если используются ее функции.

При написании программы на языке C/C++ для этого необходимо:

- Включить заголовочный файл **ltr/include/ltrXXXapi.h** для каждого используемого модуля, в котором определены все доступные константы, типы и функции для работы с данным модулем.
- Для ОС Windows:
  - Убедиться, что все нужные файлы **ltrXXXapi.dll** и **ltrapi.dll** установлены либо в директорию с программой, либо в директорию из переменной окружения **PATH** (установщик устанавливает в **%WINDIR%/system32**)
  - Подключить к проекту файлы **ltrXXXapi.lib** или **ltrXXXapi.a** (а также **ltrapi.lib** или **ltrapi.a** при использовании функций из **ltrapi**) для нужного компилятора:
    - \* *Microsoft Visual C++* 32-битный компилятор — из **ltr/lib/msvc**
    - \* *Microsoft Visual C++* 64-битный компилятор — из **ltr/lib/msvc64**
    - \* *Borland C++/Borland C++ Builder/Embarcadero C++ Builder* 32-битный компилятор — из **ltr/lib/borland**
    - \* *Embarcadero C++ Builder* 64-битный компилятор — из **ltr/lib/borland64**
    - \* *MinGW* 32-битный компилятор — из **ltr/lib/mingw**
    - \* *MinGW* 64-битный компилятор — из **ltr/lib/mingw64**
- Для ОС Linux
  - Убедиться, что файлы **libltrXXXapi.so**, **libltrXXXapi.so.\*** (где \* - версия), **libltrapi.so**, **libltrapi.so.\*** установлены либо в одну из системных директорий для библиотек, либо в директорию, заданную через переменную окружения **LD\_LIBRARY\_PATH** или другим способом (собранные пакеты устанавливают библиотеки в стандартную директорию **/usr/lib**)
  - Подключить библиотеку к проекту. При вызове GCC из командной строки это можно сделать с помощью ключа **-ltrXXXapi**.

## 4.3 Использование библиотек при написании программ на Delphi

Для написания программ на *Delphi* “Л Кард” предоставляет набор .pas файлов, в которых объявлены те же структуры и функции, что используются в библиотеках на C. В программе нужно включить через uses нужные модули: **ltrapi**, **ltrapitypes**, **ltrapidefine** и **ltrXXXapi** для каждого используемого модуля. При этом программа будет использовать те же .dll файлы, что и программа на C. При этом есть две версии .pas файлов:

- В **ltr/include/pascal** устанавливается старая версия файлов, которые оставлены только для совместимости со старыми проектами. Эти файлы требуют ручной настройки выравнивания структур в среде, их нельзя использовать в современных версиях *Delphi* и могут содержать некоторые неточности.
- В **ltr/include/pascal2** устанавливается новая версия файлов. Эти файлы можно использовать как в *Delphi 7*, так и в современных версиях *Embarcadero Delphi XE*, как для 32-битных, так и 64-битных приложениях. Эти файлы имеют небольшие несовместимости со старой версией (например для символов используется тип **AnsiChar**, а не **Char**, т.к. в современных версиях **Char** — юникодный символ, занимающий два байта, а не один). Эта версия требует лишь небольших изменений и рекомендуются для использования.

Следует иметь в виду, что также ведется изменение описания функций для старых модулей (с сохранением старых вариантов). Для большего соответствия языку выполняются следующие изменения:

- Отказ от указателей (**Pointer**) в пользу типизированных переменных **var** или **out**.
- Использование типа **string** (стандартный тип для строк в *Delphi*) для передачи строк для избавления пользователя от ручных преобразований

- Использование для передачи массивов в качестве параметра открытого массива (open array parameter), что означает, что в эти функции можно передать как статический массив, так и динамический (установив предварительно его длину с помощью *SetLength()*)

Новые модули содержат описания с учетом указанных выше изменений, для старых новый вариант функций будет добавляться по мере возможности и обновляться примеры.

#### 4.4 Использование библиотек при написании программ на языке C#

Для написания программ на C# предоставляется специальная библиотека-обертка **ltrModulesNet.dll** (устанавливается в **ltr/bin/ltrModulesNet**), которую надо подключить к создаваемому проекту. Она использует уже функции из .dll на языке C. Т.е. для работы программы нужна как **ltrModulesNet.dll**, так и остальные используемые библиотеки **ltrXXXapi.dll**.

Все классы в данной библиотеке объявлены внутри пространства **ltrModulesNet** (можно включить через `using ltrModulesNet`).

Все функции, структуры и константы объединены внутри соответствующих классов:

- Все определения из **ltrapi** находятся в классе `_LTRNative`
- Определения модулей для старого варианта **api** содержатся в классах `_ltrXXXapi`. При этом также существует аналогичный класс `ltrXXXapiLabView`, предназначенный для использования в *LabView*, отличающийся только тем, что все поля описателя модуля доступны через члены самого класса.
- Определения модулей для нового варианта **api** содержатся в классах `ltrXXXapi` (без подчеркивания). Эти классы предназначены как для использования в программах на C#, так и в программах на *LabView*. При этом в них доступ к полям описателя сделан через свойства класса. Для некоторых функций сделан вариант с более удобными типами параметров.

Существует два варианта работы с функциями:

- Через статические методы класса. Они в точности повторяют прототип и название функций на C и также принимают первым параметром структуру, соответствующую описателю модуля.
- Через методы самого класса. В этом случае создается объект данного класса, который содержит внутри себя структуру описателя модуля. Соответственно, в методы класса эта структура не передается (т.к. в методе доступны все поля экземпляра класса). Названия методов аналогичны функциям на C, но без префикса `LTRXXX_` (так как к какому модулю относится функция, понятно по классу, к которому относится этот модуль). Поля описателя модуля должны меняться либо в структуре внутри класса (для классов с подчеркиванием), либо через свойства (для новых классов без подчеркивания). Для новых классов этот вариант более предпочтителен, так как позволяет сделать дополнительно нужные преобразования в стандартные типы C# (например для строк или массивов структур).

Новый вариант классов с примерами их использования есть только для ограниченного набора модулей. По возможности список таких модулей будет расширяться.

#### 4.5 Использование библиотек при написании программ на LabView

Так как *LabView* поддерживает работу с управляемыми библиотеками **NetFramework**, то для создания программ на *LabView* можно использовать библиотеку **ltrModulesNet.dll**.

**Важно!:** Поддерживаются только версии *LabView* начиная с 8.0 (и выше), так как в предыдущих версиях хотя и была поддержка библиотек **NetFramework**, но в ней были существенные проблемы с производительностью.

Для работы с классами **.Net** в *LabView* есть специальная панель **Connectivity** -> **.Net**.

При этом необходимо использовать следующие блоки:

- **Constructor Node** — создает объект. Должен быть создан для каждого модуля **ltr**, с которым будет идти работа, а также для каждого управляющего соединения с крейтом или **ltrd/LtrServer**. При создании *LabView* предложит выбрать библиотеку и класс (нужно выбрать **ltrModulesNet.dll** и нежный класс: `ltrXXXapi` для данного модуля, если есть, или `ltrXXXapiLabView`, если нету). Одним из выходов этого блока является ссылка на объект,

которая используются как вход для остальных блоков для работы с модулем. Также с помощью конструктора создаются объекты структур, необходимых при работе с модулем.

- **Close Reference** — закрывает и удаляет объект. Должен вызываться для каждого созданного объекта по завершению работы.
- **Invoke Node** — вызов функции (метода класса). При работе с объектом на вход подается ссылка и входные параметры, а на выходе — выходные параметры и обновленная ссылка (которая должна использоваться для блоков, которые будут вызваны после текущего). Входная ссылка и определяет, методы какого объекта используются (после заведения ссылки на вход имя класса появится в верхней строке, а при нажатии на вторую будет предложен на выбор его метод). Для функций, которые не работают с конкретным объектом (эти функции статические и при выборе помечены [S] в начале) ссылку на вход подавать не обязательно. Однако они все равно принадлежат классу, который надо выбрать нажав правой кнопкой на блок и далее **Select Class/.Net**.
- **Property Node** — используется для установки или получения свойств объекта. Через свойства устанавливаются или считываются поля структур (например, настройки модуля или информация о модуле). Можно устанавливать несколько свойств одним блоком, расширив его вниз. Также с помощью свойств можно задавать константы из перечислений (что может быть более понятно, чем просто подавать на вход числа) — в этом случае надо выбрать класс перечисления, а каждому значению соответствует свое свойство.

## 4.6 Создание 64-битных программ под ОС Windows

На 64-битной версии ОС Windows могут выполняться программы, как собранные 32-битным, так и 64-битным компилятором (последние называют нативными 64-битными приложениями), поэтому многие программы для Windows существуют только в 32-битном варианте. 64-битный компилятор используют как правило для программ, работающих с большим количеством данных, так как процесс 64-битной программы имеет виртуальное пространство больше 4 Гбайт.

Так как эта возможность может быть очень нужна для систем сбора данных при большом количестве обрабатываемой информации, то начиная с версии 1.28.0 установщик **ltdll.exe** устанавливает дополнительно 64-битные версии библиотек, т.е. предоставляет возможность создания нативных 64-битных приложений. При этом изменено расположение файлов по сравнению с версиями 1.27.x и ниже (подробнее см. файл **readme.txt** среди устанавливаемых файлов).

**Важно!:** Для установки 64-битных библиотек была изменена программа для создания установщика. Поэтому, если до установки была установлена версия 1.27.x или ниже, то необходимо сперва ее удалить, а потом уже поставить новую.

При установке на 64-битной версии Windows установщик **ltdll** в соответствующие системные директории ставит и 32-битную версию библиотек, так и 64-битную. При этом директория **Windows/system32** указывает на одну из этих директорий в зависимости от разрядности самого приложения, которое обращается по указанному пути. Для 32-битного приложения в **Windows/system32** находятся 32-битные библиотеки, а 64-битные в **Windows/Sysnative**, а для 64-битного в **Windows/system32** находятся 64-битные, а **Windows/Sysnative** не существует. При этом в **Windows/SysWOW64** всегда лежат 32-битные библиотеки и она всегда существует независимо от разрядности приложения.

При загрузке приложения, если используются системные библиотеки, то они ищутся по путям из переменной окружения **PATH**, среди которых есть **Windows/system32**. Так как эта директория ссылается на разные места в зависимости от разрядности запускаемого приложения, то выбор библиотеки нужной разрядности из **Windows/system32** происходит автоматически. Если библиотеки распространяются вместе с программой, то нужно следить, чтобы разрядность собранного приложения и библиотек в одной директории совпадала.

Единственным отличием при написании программ на C/C++ является необходимость подключить lib-файл (или .a) в соответствии с разрядностью используемого компилятора.

Для программ на Delphi необходимо только указать, для какой платформы будет собираться проект (win32 или win64) и собранная программа будет использовать библиотеку той разрядности, для которой программа была скомпилирована.

Программы на языке C# (или на любом другом, использующем NetFramework) компилируются в машинный код при выполнении. При этом один раз созданная программа может выполняться как на

32-битной версии, так и на 64-битной версии виртуальной машины NetFramework (в проекте при необходимости можно указать явно для какой разрядности NetFramework предназначена программа). Таким образом, если разрядность не указана явно, то одна и та же программа в 32-битной версии Windows будет выполняться с использованием 32-битной версии библиотек, а в 64-битной — с использованием 64-битной. Для библиотеки `ltrModulesNet.dll` разрядность определяется разрядностью использующего его приложения.

Соответственно, в проекте на *LabView*, который использует .Net библиотеку `ltrModulesNet.dll`, разрядность используемой библиотеки определяется разрядностью используемой среды *LabView*.

#### 4.7 Где взять исходные коды ПО для ПК

Для большинства своих программ “Л Кард” предоставляет исходные коды пользователям. Для части проектов предоставляется доступ к открытым репозиториям системы контроля версий на [bitbucket](https://bitbucket.org/lcard/ltr_cross_sdk) (подробнее об этом см. в документе: [http://www.lcard.ru/download/lcard\\_bitbucket\\_repos.pdf](http://www.lcard.ru/download/lcard_bitbucket_repos.pdf)).

- Исходные коды библиотек `ltrapi`, службы (демона) `ltrd`, программы `LtrManager`, а также всех остальных утилит, входящих в состав `ltr_cross_sdk`, можно скачать со странички проекта на [bitbucket](https://bitbucket.org/lcard/ltr_cross_sdk): [https://bitbucket.org/lcard/ltr\\_cross\\_sdk](https://bitbucket.org/lcard/ltr_cross_sdk). Архив со всеми исходными кодами доступен по адресу [https://bitbucket.org/lcard/ltr\\_cross\\_sdk/downloads/ltr\\_cross\\_sdk\\_src.zip](https://bitbucket.org/lcard/ltr_cross_sdk/downloads/ltr_cross_sdk_src.zip).
- Исходные коды программы `LtrServer` доступны в виде архива `ltrserver_src_XXXX.zip` со странички “Программное обеспечение” любого модуля LTR.

#### 4.8 Создание пользовательской версии прошивки крейтов LTR-EU

Для некоторых задач пользователя может не устраивать штатная прошивка крейта. В качестве примера, можно привести следующие причины:

- Необходимость иметь автономное устройство, запускающее сбор и выполняющее обработку без управления с ПК
- Необходимость обеспечить минимальные и определенные задержки от изменения сигналов на входах системы до реакции на ее выходах, для чего необходимо выполнять принятие решений внутри крейта
- Необходимость реализации своих протоколов для обмена с ПК
- Необходимость управления другими устройствами по сети или RS-485 непосредственно из крейта
- и др.

Для подобных случаев пользователю предоставляется возможность изменения прошивки сигнального процессора Blackfin, установленного в крейтах LTR-EU.

**Важно!:** Следует учитывать, что разработка прошивки для встраиваемых систем сильно отличается от разработки ПО для ПК и требует соответствующих навыков.

**Важно!:** На данный момент “Л Кард” не предоставляет руководства по модификации прошивки процессора крейта, поэтому необходимую информацию программист должен быть готов получить, анализируя предоставляемые исходные коды прошивок. При этом наши специалисты готовы проконсультировать по интересующим вопросам.

Существует две версии прошивки крейтов LTR-EU, для которых “Л Кард” предоставляет пользователю исходные коды для дальнейшей модификации:

1. Прошивка на основе ОСРВ [FreeRTOS](#) с использованием стека TCP/IP [lwip](#). Данная прошивка собирается с помощью среды [VisualDSP](#) от [Analog Devices](#). Исходные коды данной прошивки можно скачать с сайта “Л Кард” по адресу: [http://www.lcard.ru/download/ltr\\_source\\_1\\_0\\_0\\_1.zip](http://www.lcard.ru/download/ltr_source_1_0_0_1.zip). Это законченная прошивка, но она не будет развиваться в дальнейшем.
2. В настоящее время разрабатывается прошивка на основе ОСРВ [RTEMS](#) с использованием стека TCP/IP из данной ОС, перенесенного из [FreeBSD](#). Данная версия призвана решить некоторые проблемы старой версии при работе по TCP/IP, а также предоставить удобный API для

реализации пользователями обработки данных внутри крейта (сходный с API для ПК). Данная прошивка находится в стадии разработки, не все возможности крейта реализованы, а API функции реализованы только для ограниченного числа модулей. По вопросам о данной прошивке просьба обращаться по почте технической поддержки (см. раздел “Контакты” сайта: <http://www.lcard.ru/contact>).

## 5 При возникновении проблем

При возникновении проблем при работе с крейтами и модулями LTR, необходимо сделать следующее:

1. Убедиться, что были прочитаны и осмыслены: данный документ, необходимые разделы [руководства пользователя](#), а также документация на используемое ПО (при наличии) или соответствующие руководства программиста при написании собственных программ.
2. Проверить, последние ли версии ПО установлены. Последние версии Вы можете скачать с сайта в разделе “Программное обеспечение” страницы сайта “Л Кард” , посвященной интересующему Вас модулю.
3. Если Вы используете свое ПО или ПО, разработанное не “Л Кард” , а другими фирмами, то необходимо проверить работу модуля на предоставляемом “Л Кард” ПО (см. [раздел 2.3](#)).
4. Посмотреть на форуме технической поддержки (<http://www.lcard.ru/forums/viewforum.php?id=1>), не обсуждалась ли уже Ваша проблема.
5. Если проблема остается, то необходимо **максимально(!)** подробно описать ее, создав тему на форме технической поддержки, либо написав на почту технической поддержки (см. раздел “Контакты” сайта: <http://www.lcard.ru/contact>).