Базовая библиотека работы с крейтом LTR

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.10

Март 2010 г.

Авторы руководства:
Кодоркин А.В.
Емельянов А.С.

ЗАО "Л-КАРД"
117105, г. Москва, Варшавское ш., д. 5, корп. 4, стр. 2

тел.: (095) 785-95-25
факс: (095) 785-95-14

Адреса в Интернет:
http://www.lcard.ru/
ftp://ftp.lcard.ru/pub

E-Mail:
Отдел продаж: *sale@lcard.ru*
Техническая поддержка: *support@lcard.ru*
Отдел кадров: *job@lcard.ru*
Общие вопросы: *lcard@lcard.ru*

Представители в регионах:
Украина: HOLIT Data Systems, *http://www.holit.com.ua/*, (044) 241-6754
Санкт-Петербург: Autex Spb Ltd., *http://www.autex.spb.ru/*, (812) 567-7202
Новосибирск: Сектор-Т, *http://www.sector-t.ru/*, (383-2) 396-592
Екатеринбург: Аск, *http://www.ask.ru/*, 71-4444
Казань: ООО 'Шатл', *shuttle@kai.ru*, (8432) 38-1600

Крейтовая система *LTR*

История ревизий настоящего документа.

Ревизия	Дата	Примечания по внесенным изменениям
1.0.0	23.01.2006	Первая доступная для пользователя ревизия.
1.0.1	23.04.2006	Добавлено описание меток времени.
1.0.2	25.07.2006	Добавлена функция LTR_SetServerProcessPriority
1.0.3	02.04.2007	Отмечены особенности многопотоковой работы
1.0.4	23.04.2007	Добавлено более подробное описание предупреждения LTR_WARNING_MODULE_IN_USE, и изменены примеры
1.0.5	04.05.2008	Добавлено описание функции LTR_GetCrateRawData
1.0.6	04.09.2008	Добавлены примеры для функции LTR_GetCrateRawData
1.0.7	05.09.2008	Обновлено описание LTR_GetCrateRawData и флагов структуры TLTR.
1.0.8	03.09.2009	Проведена корректура текста, добавлено описание функции LTR_GetCrateInfo, дополнен список констант и структур, добавлено описание функций синхронизации и коммутации служебных сигналов крейт-контроллеров LTR-EU.
1.0.9	11.01.2010	Добавлены описания новых функций API – управление сервером и соединениями с IP-крейтами из прикладной программы. Добавлен параграф "соображения сетевой безопасности".
1.0.10	31.03.2010	Добавлено описание функции LTR_GetServerVersion

На CD-ROM, входящий в комплект поставки, всегда записывается последняя ревизия данного документа. Кроме того, последнюю ревизию Вы сможете найти в разделе *библиотека файлов* на нашем сайте.

L-Card оставляет за собой право обновлять документацию без уведомления пользователей об изменениях.

Оглавление

1. О чем этот документ 7
2. Общая идеология программного интерфейса для работы с крейтом LTR 8
3. Ltrapi.dll - базовая библиотека для работы с крейтом LTR 9
3.1. Использование ltrapi.dll 9
3.2. Общий подход к работе с интерфейсными функциями библиотеки ltrapi.dll 9
3.3. Простой пример 10
3.4. Простой пример 2 10
3.5. Соображения сетевой безопасности 12
4. Описание функций, структур и констант библиотеки ltrapi.dll 13
4.1. Константы 13
4.2. Структуры 15
4.2.1. Структура TLTR 15
4.2.2. Структура TCRATE_INFO 17
4.2.3. Структура TLTR_CONFIG 17
4.2.4. Структура TIPCRATE_ENTRY 17
4.2.5. Метка времени 17
4.3. Функции 18
4.3.1. Основные функции 18
4.3.1.1. Инициализация полей структуры 18
4.3.1.2. Открытие соединения с модулем 19
4.3.1.3. Разрыв соединения с модулем 19
4.3.1.4. Состояние соединения с модулем 19
4.3.1.5. Прием данных от модуля 20
4.3.1.6. Передача данных модулю 21
4.3.2. Функции информационного характера 21
4.3.2.1. Список крейтов, подключенных к серверу 21
4.3.2.2. Список модулей крейта 22
4.3.2.3. Информация о типе и способе подключения крейта 22
4.3.3. Функции синхронизации и коммутации служебных сигналов LTR-EU 23
4.3.3.1. Управление коммутацией сигналов 23
4.3.3.2. Генерация метки "СТАРТ" 24
4.3.3.3. Начало генерации меток "СЕКУНДА" 25
4.3.3.4. Прекращение генерации меток "СЕКУНДА" 25
4.3.4. Функции вспомогательного характера 26
4.3.4.1. Преобразование кода ошибки в текстовое сообщение 26
4.3.4.2. Получение сырых данных на входе сервера (мгновенных значений) 27
4.3.5. Функции управления LTR-сервером 28
4.3.5.1. Общие замечания 28
4.3.5.2. Чтение приоритета сервера 29
4.3.5.3. Изменение приоритета сервера 29
4.3.5.4. Получение списка IP-крейтов 30
4.3.5.5. Добавление IP-крейта 31
4.3.5.6. Удаление IP-крейта 31
4.3.5.7. Подключение IP-крейта 32
4.3.5.8. Отключение IP-крейта 32
4.3.5.9. Подключение всех IP-крейтов, имеющих признак "авто" 33
4.3.5.10. Отключение всех IP-крейтов 33
4.3.5.11. Изменение режимов подключения IP-крейта 34
4.3.5.12. Чтение режима поиска IP-крейтов в локальной сети 34
4.3.5.13. Установка режима поиска IP-крейтов в локальной сети 35
4.3.5.14. Чтение уровня журнализации 35

4.3.5.15. Установка уровня журнализации....36
4.3.5.16. Перезапуск LTR-сервера....36
4.3.5.17. Останов LTR-сервера....37
4.3.5.18. Чтение номера версии LTR-сервера....37

1. О чем этот документ.

Настоящий документ – руководство программиста. Здесь рассматривается общая идеология построения программного обеспечения для работы с крейтом **LTR** и достаточно подробно описывается интерфейс базовой dll-библиотеки ***ltrapi.dll***.

В настоящем документе не рассматриваются какие-либо вопросы, касающиеся подключения сигналов, параметров и принципов функционирования аппаратной части. Эти вопросы затронуты в документе *Крейтовая система LTR. Руководство пользователя*. Кроме того, этот документ не содержит описаний библиотек для работы с конкретным типом модулей; эти описания вынесены в отдельные документы.

2. Общая идеология программного интерфейса для работы с крейтом LTR.

С точки зрения пользовательского программного обеспечения все ***физические модули***, установленные в крейт, и сам крейт-контроллер представляются как независимые друг от друга ***логические модули***. Это достигается использованием клиент-серверной идеологии. В качестве сервера выступает программа *ltrserver.exe*, в качестве клиента – пользовательское ПО.

Сервер обеспечивает прием (передачу) и буферизацию потока данных от крейта, анализ и разделение (смешивание) данных от разных ***физических модулей***, а также обмен данными с подключенными клиентами. В результате того, что механизмы *приема-анализа-разделения* данных от крейта и *анализ-смешивание-передача* данных крейту практически полностью скрыты внутри сервера, клиент работает с каждым модулем как с отдельно стоящей единицей - ***логическим модулем.***

Все взаимодействия клиента с сервером сводятся к вызовам функций, входящих в состав штатной библиотеки.

Штатная библиотека организованна в виде нескольких dll-библиотек, каждая из которых предоставляет набор логически связанных функций, реализующих взаимодействие с конкретным типом ***логических модулей***.

Примечание: Далее везде в тексте под ***модулем*** подразумевается ***логический модуль,*** если иное не оговорено специально.

3. *Ltrapi.dll* - базовая библиотека для работы с крейтом LTR.

Библиотека ***ltrapi.dll*** предоставляет набор базовых функций для работы с модулями. В функциях библиотеки не делается каких-либо предположений о типе модуля. Таким образом, весь протокол обмена, характерный для заданного типа модуля, реализуется в библиотеке более высокого уровня, которая использует ***ltrapi.dll*** как базовую библиотеку для доступа к модулю.

ltrapi.dll написана с использованием языка программирования **Borland C++** и поставляется вместе с исходными текстами.

Внимание: Функции библиотеки, строго говоря, не обеспечивают "потокобезопасную" (*thread-safe*) работу. Поэтому, во избежание недоразумений, в многопоточных приложениях пользователь должен сам организовывать, если необходимо, корректную синхронизацию вызовов интерфейсных функций в различных потоках (используя, например, критические секции, мьютексы и т.д.).

Но (!) рекомендуется создать для работы с каждым модулем отдельный поток. Например, пусть имеется 10 модулей LTR11, в таком случае целесообразно создать 10 потоков, в каждом открыть, инициалазировать модуль, запустить сбор данных, принять данные и закрыть соотвествующий модуль по окончании работы

3.1. Использование ltrapi.dll.

Для получения возможности вызова интерфейсных функций библиотеки ***ltrapi.dll*** из Вашего приложения необходимо следующее:

- создать проект в какой-либо из сред разработки;
- поместить в папку проекта или в папку, описанную в переменной окружения **PATH,** файл ***ltrapi.dll***.
- добавить в проект информацию о способе вызова интерфейсных функций dll-библиотеки и используемых типах данных. В различных средах разработки последовательность действии и приложенные усилия могут несколько отличаться:

  Borland C++/Borland C++ Builder :
  - подключить к проекту файлы ***LTR\LIB\BORLAND\ltrapi.lib, LTR\INCLUDE\ltrapi.h, LTR\INCLUDE\ltrapitypes.h*** и ***LTR\INCLUDE\ltrapidefine.h***;

  Microsoft Visual C++ :
  - подключить к проекту файлы ***LTR\LIB\MSVC\ltrapi.lib, LTR\INCLUDE\ltrapi.h, LTR\INCLUDE\ltrapitypes.h*** и ***LTR\INCLUDE\ltrapidefine.h***;

  Другие среды разработки :
  - следует обратиться к соответствующей документации на средство разработки.

- создать и добавить в проект файл с исходным текстом будущей программы;
- после этого Вы можете писать свою программу, вызывая соответствующие интерфейсные функции dll-библиотеки.

3.2. Общий подход к работе с интерфейсными функциями библиотеки ltrapi.dll.

Для взаимодействия с каким-либо модулем необходимо выполнить следующие действия:

- создать экземпляр структуры *TLTR* и проинициализировать его, вызвав функцию *LTR_Init()*
- выбрать модуль, с которым вы хотите работать задав серийный номер крейта и номер модуля (поля *csn* и *cc*)
- задать, при необходимости, сетевой адрес компьютера и TCP порт, на которые настроен LTR-сервер (поля *saddr* и *sport*)
- установить соединение с модулем, произведя вызов функции *LTR_Open()*
- осуществлять обмен данными с модулем:
- отсылать/принимать данные модуля, производя вызов функций *LTR_Send()*/*LTR_Recv()*

- при использовании канала управления (*CC_CONTROL*) можно получить список крейтов, подключенных к LTR-серверу, произведя вызов функции *LTR_GetCrates()*
- при использовании канала управления (*CC_CONTROL*) можно получить список модулей, подключенных к выбранному крейту, произведя вызов функции *LTR_GetCrateModules()*
- разорвать соединение с модулем, произведя вызов функции *LTR_Close()*

3.3. Простой пример.

```
//-----------------------------------------------------------------------------
// В данном примере осуществляется чтение списка установленных в крейт модулей
//-----------------------------------------------------------------------------
#include <stdio.h>
#include "ltr\\include\\ltrapi.h"
#include "ltr\\include\\ltrapidefine.h"
int main(void)
    {
    INT res;
    TLTR ltr;
    // инициализируем поля структуры значениями по умолчанию
    res = LTR_Init(&ltr);
    if (res == LTR_OK)
        {
        // используем сетевой адрес и порт ltr-сервера по умолчанию
        // используем серийный номер крейта по умолчанию - первый найденный
        // задаем управляющий канал
        ltr.cc = CC_CONTROL;
        // устанавливаем соединение
        res = LTR_Open(&ltr);
        if (res == LTR_WARNING_MODULE_IN_USE)
            {
            printf(">> Внимание ! Модуль уже открыт и кто - то с ним работает \n");
            res = LTR_OK;
            }
        if (res == LTR_OK)
            {
            WORD mid[MODULE_MAX];
            // получаем список модулей крейта
            res = LTR_GetCrateModules(&ltr, mid);
            if (res == LTR_OK)
                {
                printf(">> список установленных модулей:\n");
                for (int i = 0; i < MODULE_MAX; i++)
                    {
                    printf(">> slot%0.2d  mid = %d\n", i + 1, mid[i] & 0xFF);
                    }
                }
            // разрываем соединение
            LTR_Close(&ltr);
            }
        }
    // выводим сообщение об ошибке
    if (res != LTR_OK)
        printf(">> %s\n", LTR_GetErrorString(res));
    }
//-----------------------------------------------------------------------------
```

3.4. Простой пример 2.

Поскольку для работы с несколькими модулями одновременно рекомендуется использовать потоки, то небольшой пример на использование потоков :

```
#include "stdafx.h"
#include "ltr\\include\\ltrapi.h"
#include "ltr\\include\\ltrapidefine.h"


void ModuleOpenAndWork(void* Args)
      {
      INT res = LTR_OK;
      // используем сетевой адрес и порт ltr-сервера по умолчанию

      TLTR ltr = *(TLTR *)Args;
      // используем серийный номер крейта по умолчанию - первый найденный.
      // задаем управляющий канал
      ltr.cc = CC_CONTROL;
      // устанавливаем соединение
      res = LTR_Open(&ltr);
      if (res == LTR_WARNING_MODULE_IN_USE)
            {
            printf(">>Warning: Module already opened elsewhere \n");
            res = LTR_OK;
            }

      int ThreadNumber = (int)ltr.internal;
      if (res == LTR_OK)
            {
            WORD mid[MODULE_MAX];
            // получаем список модулей крейта
            res = LTR_GetCrateModules(&ltr, mid);
            if (res == LTR_OK)
                  {
                  printf(">> NumberInstalled Modules for threadNumber %d :\n",
                        ThreadNumber);
                  for (int i = 0; i < MODULE_MAX; i++)
                        {
                        printf(">> slot%0.2d  mid=%d  >> threadNumber %d \n",
                              i + 1, mid[i] & 0xFF, ThreadNumber);
                        }
                  }
            // разрываем соединение
            LTR_Close(&ltr);
            }

      if (res != LTR_OK)
            printf("Module Number %d, error %s\n",
                  ThreadNumber, LTR_GetErrorString(res));
      }

int _tmain(int argc, _TCHAR* argv[])
      {
      const int NUM_MODULE = 3;

      INT res;
      TLTR ltr[NUM_MODULE];
      HANDLE ThreadHandle[NUM_MODULE];

      // инициализируем поля структуры значениями по умолчанию
      for (int i = 0; i < NUM_MODULE; i++)
            {
            res = LTR_Init(&ltr[i]);
```

```
        if (res == LTR_OK)
              {
              ThreadHandle[i] = CreateThread(NULL,0,
                    (LPTHREAD_START_ROUTINE)ModuleOpenAndWork,
                    &ltr[i], 0, NULL);
              }
        }

  for (int i = 0; i < NUM_MODULE; i++)
        {
        WaitForSingleObject(ThreadHandle[i], INFINITE);
        CloseHandle(ThreadHandle[i]);
        }

  // выводим сообщение об ошибке
  if (res != LTR_OK)
        printf(">> %s\n", LTR_GetErrorString(res));

  return 0;
  }
```

3.5. Соображения сетевой безопасности.

В существующем ПО крейтовой системы LTR не имеется собственной системы разграничения доступа. Если необходимо использовать систему в сети с прямым подключением к интернету (на реальных IP-адресах), то настоятельно рекомендуется использовать соответственно настроенный межсетевой экран (firewall) или маршрутизатор. Это касается и связи между LTR-сервером и приложениями, и крейтов LTR-EU, подключаемых по TCP/IP.

Порты, используемые системой LTR:

Порт	Сервер	Клиент	Назначение
11110/tcp	LTR-EU (ip)	ltrserver	данные крейт-контроллера
11111/tcp*	ltrserver	приложение пользователя (ltrapi)	связь с приложением
11112/udp	LTR-EU (ip)	ltrserver	обнаружение крейтов в ЛВС
11113/tcp	LTR-EU (ip)	ltrserver	команды крейт-контроллера

(*) Программа ltrserver по умолчанию слушает свой серверный порт на всех интерфейсах (адрес привязки 0.0.0.0). При необходимости можно указать определенный адрес (например, 127.0.0.1) и другой порт, добавив параметры listen_addr и listen_port в файле конфигурации сервера (секция [LTR_SERVER_CONFIG]).

4. Описание функций, структур и констант библиотеки *ltrapi.dll*.

В настоящем разделе приведены достаточно подробные описания констант, структур и интерфейсных функций, входящих в состав библиотеки ***ltrapi.dll***.

Примечание: Рекомендованную последовательность вызовов интерфейсных функций см. *Общий подход к работе с интерфейсными функциями dll-библиотеки*.

4.1. Константы

Константа	Значение	Описание
Константы, используемые для задания логического модуля (поле cc структуры *TLTR*)		
CC_CONTROL	0	Выбирает для связи управляющий канал сервера и крейт-контроллера.
CC_MODULE1 ... CC_MODULE16	1...16	Задает канал для связи с модулем, находящимся в слоте 1...16
CC_USERDATA	18	Задает канал для работы с особым виртуальным модулем, соответствующим каналу пользовательских данных (при использовании модифицированного пользователем встроенного ПО крейтов LTR-EU)
CC_RAW_DATA_FLAG	0x4000	Флаг, инициирующий сбор "сырых" данных с крейта, доступных с помощью функции *LTR_GetCrateRawData()*
CC_DEBUG_FLAG	0x8000	Флаг, используемый в отладочных целях.
Остальные константы, используемые при заполнении и анализе полей структуры *TLTR*		
SADDR_LOCAL	0x7F000001	Локальный сетевой адрес (127.0.0.1)
FLAG_RBUF_OVF	1	Флаг переполнения буфера клиента
FLAG_RFULL_DATA	2	Флаг получения полных данных крейта в функции *LTR_GetCrateRawData()*.
FLAG_SIGNAL_TSTAMP	0x100	Флаг получения синхрометки.
CSN_SERVER_CONTROL	"#SERVER_CONTROL"	Фиктивный серийный номер, который можно использовать для *управления программой ltrserver*, даже если не подключен ни один крейт.
Константы, используемые для определения типа модуля		
MID_EMPTY	0	Модуль не установлен
MID_LTR11	0x0B0B	LTR11
MID_LTR22	0x1616	LTR22
MID_LTR27	0x1B1B	LTR27
MID_LTR43	0x2B2B	LTR43
MID_LTR51	0x3333	LTR51
MID_LTR212	0xD4D4	LTR212
MID_...		и т.д. для всех существующих модулей
Константы, используемые в стркутуре TCRATE_INFO		
CRATE_TYPE_UNKNOWN	0	Неизвестный тип крейта (ошибка).
CRATE_TYPE_LTR010	10	Тип подключенного крейта — LTR-U-8 или LTR-U-16.
CRATE_TYPE_LTR021	21	Тип подключенного крейта — LTR-U-1.

Константа	Значение	Описание
CRATE_TYPE_LTR030	30	Тип подключенного крейта — LTR-EU-8 или LTR-EU-16.
CRATE_TYPE_LTR031	31	Тип подключенного крейта — LTR-EU-2.
CRATE_TYPE_ BOOTLOADER	99	Обнаружен только загрузчик. В нормальном режиме такое значение не возникает.
CRATE_IFACE_ UNKNOWN	0	Неизвестный тип подключения крейта (ошибка).
CRATE_IFACE_USB	1	Крейт подключен к LTR-серверу по USB.
CRATE_IFACE_TCPIP	2	Крейт подключен к LTR-серверу по TCP/IP.
Константы, используемые в структуре TIPCRATE_ENTRY (состояние IP-крейта)		
CRATE_IP_STATUS_ OFFLINE	0	Соединение с крейтом не установлено.
CRATE_IP_STATUS_ CONNECTING	1	Крейт находится в процессе установления соединения.
CRATE_IP_STATUS_ ONLINE	2	Соединение с крейтом установлено.
CRATE_IP_STATUS_ ERROR	3	Некорректное состояние соединения (ошибка).
Битовые маски режимов IP-крейтов		
CRATE_IP_FLAG_ AUTOCONNECT	0x00000001	Автоматически подключать крейт при запуске сервера.
Коды ошибок		
LTR_OK	0	Выполнено без ошибок.
LTR_ERROR_ UNKNOWN	-1	Неизвестная ошибка.
LTR_ERROR_ PARAMETRS	-2	Ошибка входных параметров.
LTR_ERROR_ MEMORY_ALLOC	-3	Ошибка динамического выделения памяти.
LTR_ERROR_ OPEN_CHANNEL	-4	Ошибка открытия канала обмена с сервером.
LTR_ERROR_ OPEN_SOCKET	-5	Ошибка открытия TCP сокета.
LTR_ERROR_ CHANNEL_CLOSED	-6	Ошибка. Канал обмена с сервером не создан.
LTR_ERROR_SEND	-7	Ошибка передачи данных.
LTR_ERROR_RECV	-8	Ошибка приема данных.
LTR_ERROR_EXECUTE	-9	Ошибка при выполнения команды крейт-контроллером.
LTR_WARNING_ MODULE_IN_USE	-10	Предупреждение: канал с сервером для этого модуля уже создан (то есть модуль уже открыт), может быть выдано как ответ на функцию *LTR_Open()*. При этом предупреждении канал с сервером создается, с модулем можно работать дальше, и обязательно завершить сессию функцией *LTR_Close()*.
LTR_ERROR_NOT_ CTRL_CHANNEL	-11	Ошибка: вызванная функция может работать только с каналом CC_CONTROL, а открыт другой канал.
Остальные константы		
CRATE_MAX	16	Максимальное количество крейтов, которое может обслужить один LTR-сервер
MODULE_MAX	16	Максимальное количество слотов в одном крейте.

Кроме того, имеется набор констант-перечислений (enum) для управления крейт-контроллерами LTR-EU в части синхронизации (т.е. стартовых и секундных меток) и конфигурации специальных внутренних сигналов процессора Blackfin.

Программирование внутренних сигналов Blackfin в данном руководстве не рассматривается. Оно может потребоваться только при модификации пользователем встроенного ПО (прошивки) крейт-контроллера.

Константа	Значение	Описание
enum en_LTR_UserIoCfg Настройки внутренних сигналов на выводах процессора крейт-контроллера		
LTR_USERIO_DEFAULT	0	Значение в нормальном режиме работы.
LTR_USERIO_DIGOUT	0	(спец.) Сигнал – выход Blackfin.
LTR_USERIO_DIGIN1	1	(спец.) Сигнал – вход Blackfin и соединен с DIGIN1.
LTR_USERIO_DIGIN2	2	(спец.) Сигнал – вход Blackfin и соединен с DIGIN2.
enum en_LTR_DigOutCfg Настройки коммутации выходов DIGOUT1 и DIGOUT2		
LTR_DIGOUT_DEFAULT LTR_DIGOUT_CONST0	0	На выходе постоянный уровень логического "0".
LTR_DIGOUT_CONST1	1	На выходе постоянный уровень логической "1".
LTR_DIGOUT_USERIO0	2	(спец.) Выход подключен к зарезервированной линии 0.
LTR_DIGOUT_USERIO1	3	(спец.) Выход подключен к зарезервированной линии 1.
LTR_DIGOUT_DIGIN1	4	Выход подключен ко входу DIGIN1.
LTR_DIGOUT_DIGIN2	5	Выход подключен ко входу DIGIN2.
LTR_DIGOUT_START	6	На выход подаются метки "СТАРТ".
LTR_DIGOUT_SECOND	7	На выход подаются метки "СЕКУНДА".
enum en_LTR_MarkMode Настройки режимов генерации меток "СТАРТ" и "СЕКУНДА"		
LTR_MARK_OFF	0	Метка отключена
LTR_MARK_EXT_ DIGIN1_RISE	1	Внешняя генерация: метка фиксируется по фронту сигнала DIGIN1
LTR_MARK_EXT_ DIGIN1_FALL	2	Внешняя генерация: метка фиксируется по спаду сигнала DIGIN1
LTR_MARK_EXT_ DIGIN2_RISE	3	Внешняя генерация: метка фиксируется по фронту сигнала DIGIN2
LTR_MARK_EXT_ DIGIN2_FALL	4	Внешняя генерация: метка фиксируется по спаду сигнала DIGIN2
LTR_MARK_INTERNAL	5	Внутренняя генерация: метка формируется крейт-контроллером.

4.2. Структуры

4.2.1. Структура TLTR

Структура *TLTR* – основная структура, содержащая всю необходимую информацию о конфигурации и состоянии канала связи с выбранным модулем. Эта структура используется во всех библиотечных функциях обмена с модулем.

Определение структуры находится в файле `ltrapi.h` и представлено ниже:

```
typedef struct {                    //
 DWORD saddr;                       // сетевой адрес сервера
 WORD  sport;                       // сетевой порт сервера
 BYTE  csn[16];                     // серийный номер крейта
```

```
 WORD  cc;                          // номер модуля крейта
 DWORD flags;                       // флаги состояния связи с модулем
 DWORD tmark;                       // последняя принятая метка времени
 LPVOID internal;                   // указатель на канал связи с модулем
} TLTR;                             //
```

Перед началом работы с модулем необходимо создать экземпляр данной структуры и проинициализировать поля значениями по умолчанию, вызвав функцию *LTR_Init()*.

Назва-ние поля	Значение по умолчанию	Назначение и допустимые значения поля
saddr	SADDR_LOCAL	Позволяет задать сетевой адрес LTR-сервера. Сетевой адрес LTR-сервера – это упакованный в 32-х битное host-endian беззнаковое целое IP-адрес компьютера, на котором запущен LTR-сервер. **Пример:** Если ip-адрес компьютера “a.b.c.d”, то поле saddr должно принять значение (a<<24)|(b<<16)|(c<<8)|(d<<0). **Примечание:** Если LTR-сервер запущен на том же компьютере, что и пользовательская программа, то в качестве сетевого адреса удобно использовать локальный сетевой адрес 127.0.0.1 (SADDR_LOCAL).
sport	11111	Позволяет задать сетевой порт LTR-сервера. Сетевой порт LTR-сервера – это 16-битное беззнаковое целое (host-endian), указывающее TCP порт, на который настроен LTR-сервер.
csn	{0}	Позволяет задать серийный номер LTR-крейта. Серийный номер крейта – это строка длиной до SERIAL_NUMBER_SIZE (16) символов. Если дина серийного номера меньше, то строка должна заканчиваться нулем. **Примечание 1:** Если при вызове функции *LTR_Open()* в качестве серийного номера указать пустую строку, то будет произведена попытка установить соединение с первым найденным LTR-крейтом. В случае успеха на выходе данное поле будет содержать серийный номер крейта, с которым установлено соединение. **Примечание 2**: В версиях LTR-сервера начиная с 1.5.2.1 можно установить специальное соединение "без крейта" для управления работой LTR-сервера (в т.ч. если к системе не подключен ни один крейт). Для этого в поле csn следует записать константу CSN_SERVER_CONTROL, а поле cc должно иметь значение CC_CONTROL.
cc	*CC_CONTROL*	Позволяет задать номер модуля LTR-крейта. Номер модуля – это 16-ти битовое беззнаковое целое, однозначно идентифицирующее ***логический модуль*** выбранного LTR-крейта. Допустимые значения: *CC_CONTROL*, *CC_MODULE1*, *CC_MODULE2*,.., *CC_MODULE16, CC_USERDATA*
flags	0	Содержит флаги состояния канала связи с модулем. FLAG_RBUF_OVF – флаг переполнения буфера в LTR-сервере. Сигнализирует о том что пользовательское(клиентское) приложение не успевает забирать данные принимаемые от модуля. Флаг обновляется при вызове функции *LTR_Recv()*. FLAG_RFULL_DATA – служебный флаг, используется в функции *LTR_GetCrateRawData()*;
tmark	0	Содержит последнюю принятую от LTR-сервера *метку времени*.
internal	NULL	Содержит указатель на канал связи с модулем.

		Поле зарезервировано для внутреннего использования.

4.2.2. Структура TCRATE_INFO

Структура TCRATE_INFO заполняется функцией *LTR_GetCrateInfo()* при передаче ей указателя на структуру *TLTR* с открытым каналом CC_CONTROL. Эта структура содержит информацию о модели и способе подключения крейта.

Определение структуры находится в файле `ltrapitypes.h` и представлено ниже:

```
typedef struct {
 BYTE  CrateType;          // Тип крейта (одна из констант CRATE_TYPE_...)
 BYTE  CrateInterface;     // Тип подключения крейта (одна из констант CRATE_IFACE_...)
} TCRATE_INFO;
```

4.2.3. Структура TLTR_CONFIG

Данная структура используется только для крейтов LTR-EU с процессором Blackfin и предназначена для коммутации цифровых входов и выходов разъема синхронизации крейта "SYNC".

Определение структуры находится в файле `ltrapi.h` и представлено ниже:

```
typedef struct {
 WORD userio[4];     // Настройки сигналов процессора (одна из констант LTR_USERIO_...)
 WORD digout[2];     // Настройки выходов DIGOUTx (одна из констант LTR_DIGOUT_...)
 WORD digout_en;     // Разрешение работы DIGOUT1 и DIGOUT2 на выход.
} TLTR_CONFIG;
```

Элементы массива userio[] задают коммутацию специальных сигналов процессора Blackfin и при нормальной работе ***должны быть равны LTR_USERIO_DEFAULT***.

Элементы массива digout[0], digout[1] задают коммутацию линий разъема "SYNC" DIGOUT1 и DIGOUT2 соответственно.

Ненулевое значение поля digout_en разрешает работу линий DIGOUT1 и DIGOUT2 как активных выходов (одновременно включаются или выключаются оба выхода). Если digout_en = 0, то обе линии DIGOUT переходят в высокоимпедансное состояние.

4.2.4. Структура TIPCRATE_ENTRY

Массив стркутур TIPCRATE_ENTRY заполняется функцией *LTR_GetListOfIPCrates()* при передаче ей указателя на структуру *TLTR* с открытым каналом CC_CONTROL. Эта структура содержит информацию об адресе, режиме подключения и состоянии крейта, подключение к которому осуществляется по TCP/IP.

Определение структуры находится в файле `ltrapitypes.h` и представлено ниже:

```
typedef struct {
 DWORD ip_addr;           // IP адрес (host-endian, 0x01020304 = 1.2.3.4)
 DWORD flags;             // флаги режимов (сумма констант CRATE_IP_FLAG_...)
 CHAR serial_number[16];  // серийный номер (если крейт подключен)
 BYTE is_dynamic;         // 0 = крейт задан пользователем, 1 = найден автоматически
 BYTE status;             // состояние (одна из констант CRATE_IP_STATUS_...)
} TIPCRATE_ENTRY;
```

4.2.5. Метка времени.

Метка времени представляет собой 32-х разрядное беззнаковое целое, служащее для синхронизации данных, собранных различными модулями. (см. также *LTR_Recv()*)

Биты	Описание

0..15	Счетчик секундных меток, инкрементируемый LTR-сервером каждый раз при приеме от модуля LTR41, LTR42, LTR43 или от крейт-контроллера слова, содержащего метку "СЕКУНДА".
16..31	Счетчик стартовых меток, инкрементируемый LTR-сервером каждый раз при приеме от модуля LTR41, LTR42, LTR43 или от крейт-контроллера слова, содержащего метку "СТАРТ".

4.3. Функции

Все интерфейсные функции библиотеки ***ltrapi.dll***, кроме функции *LTR_GetErrorString()*, в качестве первого параметра принимают указатель на экземпляр структуры *TLTR*.

Также все интерфейсные функции имеют один и тот же тип возвращаемого значения – INT. Возвращаемое значение сообщает о результате выполнения функции. Отрицательные значения сигнализируют о возникновении *ошибки*. Значение 0 (LTR_OK) соответствует успешному завершению функции (исключение составляют функции *LTR_Recv()* и *LTR_Send()*). Положительные значения определены только для функций *LTR_Recv()* и *LTR_Send()* и определяют количество принятых или переданных данных.

4.3.1. Основные функции

Функции этой подгруппы применимы ко всем типам модулей.

4.3.1.1. Инициализация полей структуры

Формат: `INT LTR_Init(TLTR *ltr)`

Назначение: Инициализирует поля структуры значениями по умолчанию.

Эту функцию необходимо вызвать однократно для каждого созданного экземпляра структуры *TLTR*, прежде чем будут вызваны остальные функции библиотеки.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Возвращаемое значение:

- *Код ошибки*.

4.3.1.2. Открытие соединения с модулем

Формат: `INT LTR_Open(TLTR *ltr)`

Назначение: Открывает соединение с модулем.

Эту функцию необходимо вызвать перед началом обмена данными с модулем. Выбор модуля осуществляется полями структуры *TLTR*. Если переданный экземпляр структуры указывает на открытое соединение с модулем, то это соединение будет автоматически разорвано и будет произведена попытка вновь установить соединение с заданным модулем.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Примечания:

Если функция вернула предупреждение LTR_WARNING_MODULE_IN_USE, то это значит, что к данному модулю уже имеются подключения. При этом канал обмена данными создается успешно, с ним можно работать, либо считать данную ситуацию ошибкой, на усмотрение программиста. В последнем случае, тем не менее, необходимо закрыть канал функцией *LTR_Close()*.

Возвращаемое значение:

- *Код ошибки*.

4.3.1.3. Разрыв соединения с модулем

Формат: `INT LTR_Close(TLTR *ltr)`

Назначение: Прекращает соединение с модулем.

Эту функцию необходимо вызвать после окончания обмена данными с модулем для корректного завершения соединения и освобождения ресурсов системы, выделенных при открытии соединения.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Возвращаемое значение:

- *Код ошибки*.

4.3.1.4. Состояние соединения с модулем

Формат: `INT LTR_IsOpened(TLTR *ltr)`

Назначение: Определяет состояние соединения с модулем.

Функция позволяет определить текущее состояние соединения с модулем. Если ltr соответствует открытому дескриптору, то возвращается LTR_OK.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Возвращаемое значение:

- *Код ошибки*.

4.3.1.5. Прием данных от модуля

Формат: `INT LTR_Recv(TLTR *ltr, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)`

Назначение: Производит прием данных от модуля.

Функция осуществляет прием данных от модуля. Формат принятых данных зависит от типа модуля и в этом документе не рассматривается.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- data – указатель на массив типа DWORD[size], в который будут помещены принятые от модуля данные.
- tmark – указатель на массив типа DWORD[size], в который будут помещены *метки времени*, соответствующие принятым данным. Таким образом, каждому элементу массива data[i] соответствует элемент массива tmark[i], содержащий метку времени. Если необходимости во временных метках нет, в качестве параметра tmark можно передать значение NULL.
- size – количество двойных слов (DWORD), которое следует принять от модуля.
- timeout – интервал времени в миллисекундах, в течение которого следует ожидать приема запрошенного количества двойных слов. Если в течение указанного интервала времени данные от модуля получены не будут, то произойдет выход из функции.

Возвращаемое значение:

- Отрицательные значения следует интерпретировать как *коды ошибок*. Неотрицательные значения следует обрабатывать как количество реально принятых от модуля двойных слов за отведенный интервал времени.

4.3.1.6. Передача данных модулю

Формат: `INT LTR_Send(TLTR *ltr, DWORD *data, DWORD size, DWORD timeout)`

Назначение: Производит передачу данных модулю.

Функция осуществляет передачу данных модулю. Формат передаваемых данных зависит от типа модуля и в этом документе не рассматривается.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- data – указатель на массив типа DWORD[size], содержащий данные, предназначенные для передачи модулю.
- size – количество двойных слов (DWORD), которое следует передать модулю.
- timeout – интервал времени в миллисекундах, в течение которого следует ожидать завершения передачи запрошенного количества двойных слов. Если в течение указанного интервала времени данные модулю переданы не будут, то произойдет выход из функции.

Возвращаемое значение:

- Отрицательные значения следует интерпретировать как *коды ошибок*. Неотрицательные значения следует обрабатывать как количество реально переданных модулю двойных слов за отведенный интервал времени.

4.3.2. Функции информационного характера

Функции этой подгруппы применимы только для управляющего канала, т.е. для структуры *TLTR* с открытым каналом *CC_CONTROL*.

4.3.2.1. Список крейтов, подключенных к серверу

Формат: `INT LTR_GetCrates(TLTR *ltr, BYTE *csn)`

Назначение: Получает список крейтов, подключенных к LTR-серверу.

Функция возвращает список крейтов, подключенных к серверу. Распознавание крейтов сервером происходит только по их серийным номерам, поэтому возвращаемый функцией список представляет собой список серийных номеров подключенных крейтов.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- csn – указатель на массив типа BYTE[CRATE_MAX][SERIAL_NUMBER_SIZE], который будет заполнен серийными номерами подключенных к серверу крейтов.

Возвращаемое значение:

- *Код ошибки*

4.3.2.2. Список модулей крейта

Формат: `INT LTR_GetCrateModules(TLTR *ltr, WORD *mid)`

Назначение: Получает список модулей крейта.

Функция возвращает список установленных в крейт модулей. Каждый модуль имеет двухбайтовый *идентификатор*, однозначно определяющий его тип.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- mid – указатель на массив типа WORD[MODULE_MAX], который будет заполнен *идентификаторами* установленных в крейт модулей.

Возвращаемое значение:

- *Код ошибки*

4.3.2.3. Информация о типе и способе подключения крейта

Формат: `INT LTR_GetCrateInfo(TLTR *ltr, TCRATE_INFO *CrateInfo)`

Назначение: Получает информацию о типе крейта, с которым установлено соединение, и способе его подключения к LTR-серверу.

Функция запрашивает тип крейта и способ подключения и заполняет структуру *CrateInfo.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- CrateInfo – указатель на экземпляр структуры *TCRATE_INFO*, который при успешном выходе из функции будет заполнен соответствующими *значениями*.

Возвращаемое значение:

- *Код ошибки*

4.3.3. Функции синхронизации и коммутации служебных сигналов LTR-EU

Данная подгруппа объединяет функции, позволяющие задействовать расширенные возможности крейт-контроллеров серий LTR-EU (с процессором Blackfin) в части генерации меток "СТАРТ" и "СЕКУНДА" и управлять коммутацией входов и выходов разъема синхронизации "SYNC", а также специальных внутренних сигналов процессора Blackfin, которые могут потребоваться при использовании модифицированных пользователем версий встроенного ПО крейт-контроллера.

Функции этой подгруппы применимы только для управляющего канала, т.е. для структуры *TLTR* с открытым каналом *CC_CONTROL*.

Для крейтов серий LTR-U эти функции не поддерживаются, и вызывать их не следует. При необходимости узнать тип крейта можно с помощью функции *LTR_GetCrateInfo()*.

Примечание: средства генерации меток "СТАРТ" и "СЕКУНДА" также присутствуют в модулях LTR-41, LTR-42, LTR-43. Во избежание дублирования меток и искажения привязки к реальному времени при задействовании средств синхронизации крейт-контроллера недопустимо одновременно включать аналогичные функции модулей. *Это особенно важно учитывать при модернизации существующего программного обеспечения.*

4.3.3.1. Управление коммутацией сигналов

Формат: `INT LTR_Config(TLTR *ltr, const TLTR_CONFIG *conf)`

Назначение: Устанавливает конфигурацию синхросигналов и внутренних линий процессора в соответствии с полями структуры *conf.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*, соответствующей открытому каналу CC_CONTROL крейта LTR-EU.
- conf – указатель на экземпляр структуры *TLTR_CONFIG*.

Примечания:

Эту функцию необходимо вызвать перед началом работы с остальными функциями данной группы, чтобы задать необходимое состояние крейт-контроллера, в т.ч. если используется только внутренняя синхронизация.

Подробное описание назначения полей структуры приведено в 4.2.3.

Возвращаемое значение:

- *Код ошибки*

4.3.3.2. Генерация метки "СТАРТ"

Формат: `INT LTR_MakeStartMark(TLTR *ltr, enum en_LTR_MarkMode mode)`

Назначение: Управляет генерацией меток "СТАРТ" по внутреннему или внешнему событию..

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*, соответствующей открытому каналу CC_CONTROL крейта LTR-EU.
- mode – одно из значений *enum en_LTR_MarkMode*, определяющее режим работы.

Примечания:

Эта функция должна использоваться одним из двух следующих способов в зависимости от того, внешние ли используются метки "СТАРТ" или внутренние.

Для *внутренней однократной* генерации метки необходимо вызвать функцию с параметром mode = LTR_MARK_INTERNAL в тот момент, когда требуется метка. Непосредственно после получения команды крейт-контроллер сформирует одну метку "СТАРТ", отправит ее LTR-серверу и, если задана трансляция меток на выходы разъема "SYNC", сформирует соответствующий импульс.

Для *внешней* генерации меток необходимо предварительно вызвать функцию с параметром mode = LTR_MARK_EXT_DIGINx_RISE или LTR_MARK_EXT_DIGINx_FALL (x = 1 или 2), тем самым разрешается генерация меток "СТАРТ" по фронту или спаду соответствующего сигнала на входе разъема "SYNC". Отключить внешнюю генерацию меток можно вызовом этой же функции с параметром mode = LTR_MARK_OFF.

Возвращаемое значение:

- *Код ошибки*

4.3.3.3. Начало генерации меток "СЕКУНДА"

Формат: `INT LTR_StartSecondMark(TLTR *ltr, enum en_LTR_MarkMode mode)`

Назначение: Запускает генерацию меток "СЕКУНДА" по внутреннему или внешнему событию.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*, соответствующей открытому каналу CC_CONTROL крейта LTR-EU.
- mode – одно из значений *enum en_LTR_MarkMode*, определяющее режим работы.

Примечания:

Для *внутренней* генерации меток необходимо вызвать функцию с параметром mode = LTR_MARK_INTERNAL. После получения команды крейт-контроллер начинает формировать метки "СЕКУНДА" с периодичностью 1 с, отправляя их LTR-серверу и, если задана трансляция меток на выходы разъема "SYNC", формируя соответствующие импульсы.

Для *внешней* генерации меток необходимо вызвать функцию с параметром mode = LTR_MARK_EXT_DIGINx_RISE или LTR_MARK_EXT_DIGINx_FALL (x = 1 или 2), тем самым разрешается генерация меток "СЕКУНДА" по фронту или спаду соответствующего сигнала на входе разъема "SYNC".

Значение mode = LTR_MARK_OFF останавливает генерацию секундных меток и имеет тот же смысл, что и вызов функции *LTR_StopSecondMark()*.

Возвращаемое значение:

- *Код ошибки*

4.3.3.4. Прекращение генерации меток "СЕКУНДА"

Формат: `INT LTR_StopSecondMark(TLTR *ltr)`

Назначение: Останавливает генерацию меток "СЕКУНДА" по внутреннему или внешнему событию.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*, соответствующей открытому каналу CC_CONTROL крейта LTR-EU.

Примечания:

Вызов этой функции отменяет действие любых предшествовавших вызовов функции *LTR_StartSecondMark()* и запрещает генерацию крейт-контроллером секундных меток (внешних или внутренних).

Возвращаемое значение:

- *Код ошибки*

4.3.4. Функции вспомогательного характера

4.3.4.1. Преобразование кода ошибки в текстовое сообщение

Формат: `LPCSTR LTR_GetErrorString(INT error)`
Назначение: Получает сообщение об ошибке в текстовом виде по числовому коду. Функция возвращает строку, содержащую сообщение об ошибке, соответствующее переданному в функцию коду ошибки.
Передаваемые параметры: • error – код ошибки .
Возвращаемое значение: • указатель на строковую константу, содержащую сообщение об ошибке.

4.3.4.2. Получение сырых данных на входе сервера (мгновенных значений)

Формат: `INT LTR_GetCrateRawData(TLTR *ltr, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)`

Назначение: Осуществляет прием данных напрямую от крейта, без обработки сервером.

Функция осуществляет прием последних поступивших данных непосредственно от крейта, без обработки сервером. При этом перемешаны данные от модулей и временные метки.

Может использоваться для получения мгновенных значений.

Размер запрашиваемых данных (size) – не более 1024*1024 значений.

Варианты использования функции:

1. Получение последних мгновенных данных из крейта:
   `size > 0`, в **`ltr->flags`** отсутствует флаг **`FLAG_RFULL_DATA`**.. При этом переполнение внутренних буферов крейта не сигнализируется флагом **`FLAG_RBUF_OVF`**.
2. Получение размера сырых данных, накопившихся в сервере:
   `size = 0, FLAG_RFULL_DATA` не имеет значения. В таком случае возвращаемое функцией значение указывает размер накопленных данных. Однако на практике для получения данных рекомендуется использовать цикл с таймаутом.
3. Получение полных данных из крейта:
   `size > 0`, в **`ltr->flags`** присутствует флаг **`FLAG_RFULL_DATA.`**

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR* с открытым каналом *CC_CONTROL*.
- data – указатель на массив типа DWORD[size], в который будут помещены принятые от крейта данные.
- tmark – указатель на массив типа DWORD[size], в который будут помещены *метки времени*, соответствующие принятым данным. Таким образом, каждому элементу массива data[i] соответствует элемент массива tmark[i], содержащий метку времени. Если необходимости во временных метках нет, в качестве параметра tmark можно передать значение NULL.
- size – количество двойных слов (DWORD), которое следует принять от крейта.
- timeout – интервал времени в миллисекундах, в течение которого следует ожидать приема запрошенного количества двойных слов. Если в течение указанного интервала времени данные от крейта получены не будут, то произойдет выход из функции.

Возвращаемое значение:

- *Код ошибки*
- При (size = 0) – размер накопленных данных.

Пример получения последних мгновенных данных:

```
int i, nslot;
INT res, data_size;
TLTR module;
TLTR11 hltr11;
DWORD data[NumRawData];
double VoltsData[NumRawData];

LTR_Init(&module);
module.cc = CC_CONTROL;
LTR_Open(&module);
LTR11_Init(&hltr11);
nslot = 10;  //номер слота модуля LTR11
res = LTR11_Open(&hltr11, SADDR_DEFAULT, SPORT_DEFAULT, "", nslot);
if (res == LTR_OK)
      {
      printf("LTR11 Open OK\n");
      res = LTR11_GetConfig(&hltr11);
      if (res == LTR_OK)
            {
            printf("LTR11 getconfig OK\n");
            hltr11.InpMode = LTR11_INPMODE_INT;
            hltr11.LChQnt = 16;
            for(i = 0; i < 16; i++)
                  {
                  hltr11.LChTbl[i] = (BYTE)i;
                  }
            hltr11.ADCMode = LTR11_ADCMODE_ACQ;
            hltr11.ADCRate.divider = 36;
            hltr11.ADCRate.prescaler = 8;
            res = LTR11_SetADC(&hltr11);
            if (res == LTR_OK)
                  {
                  printf("LTR11 setadc OK\n");
                  LTR11_Start(&hltr11);
                  }
            }
      }
while (!kbhit())
      { // принимаем и отображаем
      LTR_GetCrateRawData(&module, data, NULL, NumRawData, 1000);
      data_size = NumRawData;
      res = LTR11_ProcessData(&hltr11, data, VoltsData, &data_size, TRUE, TRUE);
      if (res==0)
            printf(" res=%00004d %10.4f %10.4f%10.4f\n",
                  res, VoltsData[0], VoltsData[1], VoltsData[2]);
      }
LTR11_Stop(&hltr11);
LTR11_Close(&hltr11);
LTR_Close(&module);
```

4.3.5. Функции управления LTR-сервером

4.3.5.1. Общие замечания

Функции этой подгруппы позволяют программно управлять работой программы ltrserver аналогично (и параллельно) командам, доступным через графический интерфейс (в т.ч. конфигурировать, подключать, отключать крейты по TCP/IP, перезапускать LTR-сервер и т.д.).

Все функции этой подгруппы применимы только для управляющего канала, т.е. для структуры *TLTR* с открытым каналом *CC_CONTROL*.

Соединение без крейта: Если в системе еще нет ни одного подключенного крейта, то установить соединение с "первым найденным" крейтом (с пустым полем csn структуры *TLTR*) не удастся. Чтобы соединиться с управляющим каналом LTR-сервера в режиме "без крейта", следует использовать специальный фиктивный серийный номер CSN_SERVER_CONTROL. Например:

```
LTR_Init(&ltr);
strncpy(ltr.csn, CSN_SERVER_CONTROL, sizeof(ltr.csn));
ltr.cc = CC_CONTROL;
ltr_result = LTR_Open(&ltr);
```

Однако этот режим не является обязательным. Функции управления сервером доступны и для обычного соединения ("с крейтом"), единственное непременное условие – номер канала должен быть *CC_CONTROL*.

4.3.5.2. Чтение приоритета сервера

Формат: `INT LTR_GetServerProcessPriority(TLTR *ltr, DWORD *Priority)`

Назначение: Получает информацию о текущем приоритете процесса LTR-сервера в системе.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- Priority – указатель на переменную, принимающую считанное значение. Значение *Priority соответствует Win32 API :

`IDLE_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, REALTIME_PRIORITY_CLASS.`

Возвращаемое значение:

- *Код ошибки*

4.3.5.3. Изменение приоритета сервера

Формат: `INT LTR_SetServerProcessPriority(TLTR *ltr, DWORD Priority)`

Назначение: Устанавливает новый приоритет процесса LTR-сервера в системе. Повышение приоритета полезно при постоянной загрузке компьютера посторонними программами, когда может возникнуть переполнение буфера крейта из-за того, что сервер не успел считать информацию. Особенно это критично для одноместного крейта LTR-21.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- Priority – приоритет согласно Win32 API :

`IDLE_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, REALTIME_PRIORITY_CLASS.`

Рекомендуется использовать значения: для повышенного приоритета `HIGH_PRIORITY_CLASS`, для нормального – `NORMAL_PRIORITY_CLASS.`

Возвращаемое значение:

- *Код ошибки*

4.3.5.4. Получение списка IP-крейтов

Формат: `INT LTR_GetListOfIPCrates(TLTR *ltr, DWORD max_entries, DWORD ip_net, DWORD ip_mask, DWORD *entries_found, DWORD *entries_returned, TIPCRATE_ENTRY *info_array)`

Назначение: Получает выборку из списка известных LTR-серверу IP-адресов крейтов (как активных, т.е. подключенных в настоящий момент, так и неактивных).

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- max_entries – максимальное количество информационных записей, которое может принять массив info_array.
- ip_net, ip_mask – адрес и маска IP-сети (host endian), задающие фильтр для отбора записей. Учитываются только IP-адреса крейтов, попадающие в указанную сеть. Например: ip_net = 0xC0A80100 (соответствует "192.168.1.0"), ip_mask = 0xFFFFFF00 (соответствует "255.255.255.0") – поиск только крейтов с IP-адресами 192.168.1.x.
- entries_found – указатель на переменную, принимающую количество найденных крейтов (всего), удовлетворяющих заданной сетевой маске. Это значение может быть больше, чем max_entries.
- entries_returned – указатель на переменную, принимающую на выходе количество записей в массиве info_array. В норме это значение равно min(max_entries, entries_found).
- info_array – указатель на массив структур *TIPCRATE_ENTRY* (длиной не менее max_entries), в который будут помещены IP-адреса и флаги состояния найденных крейтов. Допустимо сочетание info_array = NULL, max_entries = 0.

Примечания:

Данная функция может использоваться в различных целях: для проверки наличия и определения состояния крейта по его IP-адресу (ip_net равно IP-адресу, ip_mask = 0xFFFFFFFF), для получения списка всех известных крейтов (ip_net = ip_mask = 0) и т.д.

Возвращаемое значение:

- *Код ошибки*

4.3.5.5. Добавление IP-крейта

Формат: `INT LTR_AddIPCrate(TLTR *ltr, DWORD ip_addr, DWORD flags, BOOL permanent)`

Назначение: Добавляет IP-крейт в список крейтов, известных серверу (список отображается в интерфейсе программы ltrserver). Для того, чтобы установить соединение с крейтом, он должен быть сначала внесен в этот список.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- ip_addr – IP-адрес добавляемого крейта (host endian, 0x01020304 = "1.2.3.4").
- flags – флаги режимов крейта (сумма констант *CRATE_IP_FLAG_...*).
- permanent – если TRUE, то адрес сохраняется в файле конфигурации LTR-сервера.

Примечания:

Если крейт с указанным адресом уже есть в списке, команда завершается с ошибкой. Для проверки наличия крейта можно использовать функцию *LTR_GetListOfIPCrates()*.

Возвращаемое значение:

- *Код ошибки*

4.3.5.6. Удаление IP-крейта

Формат: `INT LTR_DeleteIPCrate(TLTR *ltr, DWORD ip_addr, BOOL permanent)`

Назначение: Удаляет крейт из списка известных серверу. С крейтом не должно быть установлено соединение.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- ip_addr – IP-адрес удаляемого крейта (host endian, 0x01020304 = "1.2.3.4").
- permanent – если TRUE, то адрес удаляется также из файла конфигурации LTR-сервера.

Примечания:

Удалить можно как "статический" адрес (заданный пользователем или добавленный программно), так и адрес крейта, найденного сервером автоматически в локальной сети (если с ним не установлено соединение). В последнем случае следует учитывать, что при включенном автоматическом поиске, если крейт присутствует в сети, он будет повторно обнаружен сервером и добавлен в список снова.

Если крейт в данный момент занят (подключен), либо адреса нет в списке, то команда завершается с ошибкой. Для проверки наличия крейта в списке и его состояния можно использовать функцию *LTR_GetListOfIPCrates()*.

Возвращаемое значение:

- *Код ошибки*

4.3.5.7. Подключение IP-крейта

Формат: `INT LTR_ConnectIPCrate(TLTR *ltr, DWORD ip_addr)`

Назначение: Инициирует установление соединения с крейтом по TCP/IP.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- ip_addr – IP-адрес крейта (host endian, 0x01020304 = "1.2.3.4").

Примечания:

Адрес крейта должен присутствовать в списке IP-крейтов, известных LTR-серверу, т.е. должен быть заранее быть внесен в этот список (любым способом – через файл конфигурации, через графический интерфейс, функцией *LTR_AddIPCrate()* или автоматически обнаружен в локальной сети).

Если крейт уже подключен или уже находится в процессе установления соединения, то возвращается LTR_OK.

Успешное завершение этой функции означает лишь, что **отправлена команда** на установление соединения с крейтом. Процедура подключения занимает некоторое время. Результат операции следует проверить позже с помощью функций *LTR_GetCrates()* или *LTR_GetListOfIPCrates()*. Например, можно организовать циклический опрос сервера с тайм-аутом.

Возвращаемое значение:

- *Код ошибки*

4.3.5.8. Отключение IP-крейта

Формат: `INT LTR_DisconnectIPCrate(TLTR *ltr, DWORD ip_addr)`

Назначение: Прекращает соединение с крейтом по TCP/IP.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- ip_addr – IP-адрес крейта (host endian, 0x01020304 = "1.2.3.4").

Примечания:

Адрес крейта должен присутствовать в списке IP-крейтов, известных LTR-серверу.

Если крейт не был подключен, то возвращается LTR_OK.

Успешное завершение этой функции означает лишь, что **отправлена команда** на прекращение соединения с крейтом. Процедура отключения занимает некоторое время. Результат операции, если требуется, можно проверить позже с помощью функций *LTR_GetCrates()* или *LTR_GetListOfIPCrates()*. Например, можно организовать циклический опрос сервера с тайм-аутом.

Возвращаемое значение:

- *Код ошибки*

4.3.5.9. Подключение всех IP-крейтов, имеющих признак "авто"

Формат: `INT LTR_ConnectAllAutoIPCrates(TLTR *ltr)`

Назначение: Инициирует установление соединения по TCP/IP со всеми известными крейтами, у которых установлен флаг автоматического подключения при запуске сервера (CRATE_IP_FLAG_AUTOCONNECT)

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Примечания:

Крейты, с которыми уже установлено соединение, остаются подключенными.

Для крейтов, находящихся в процессе установления соединения, производится сброс попытки подключения, и процесс подключения начинается заново.

Успешное завершение этой функции означает лишь, что **отправлена команда** на установление соединения с крейтами (их, в частности, может быть 0 штук). Процедура подключения занимает некоторое время. Результат операции следует проверить позже с помощью функций *LTR_GetCrates()* или *LTR_GetListOfIPCrates()*. Например, можно организовать циклический опрос сервера с тайм-аутом.

Возвращаемое значение:

- *Код ошибки*

4.3.5.10. Отключение всех IP-крейтов

Формат: `INT LTR_DisconnectAllIPCrates(TLTR *ltr)`

Назначение: Прекращает соединение со всеми известными крейтами, подключенными по TCP/IP.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Примечания:

Отключаются все подключенные по TCP/IP крейты (заданные пользователем, найденные автоматически, имеющие флаг автоматического подключения при запуске сервера и не имеющие его, и т.д.). Для крейтов, находящихся в процессе установления соединения, производится сброс попытки подключения.

Успешное завершение этой функции означает лишь, что **отправлена команда** на прекращение соединения с крейтами. Процедура отключения занимает некоторое время. Результат операции, если требуется, можно проверить позже с помощью функций *LTR_GetCrates()* или *LTR_GetListOfIPCrates()*. Например, можно организовать циклический опрос сервера с тайм-аутом.

Возвращаемое значение:

- *Код ошибки*

4.3.5.11. Изменение режимов подключения IP-крейта

Формат: `INT LTR_SetIPCrateFlags(TLTR *ltr, DWORD ip_addr, DWORD new_flags, BOOL permanent)`

Назначение: Устанавливает новое значение флагов, опередеяющих режимы подключения крейта по TCP/IP.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- ip_addr – IP-адрес крейта (host endian, 0x01020304 = "1.2.3.4").
- new_flags – флаги режимов крейта (сумма констант *CRATE_IP_FLAG_...*).
- permanent – если TRUE, то адрес с новыми флагами сохраняется в файле конфигурации LTR-сервера.

Примечания:

Адрес крейта должен присутствовать в списке IP-крейтов, известных LTR-серверу.

На момент написания данной редакции определен только один флаг CRATE_IP_FLAG_AUTOCONNECT, наличие (ненулевое состояние) которого означает, что с крейтом должно автоматически устанавливаться соединение при запуске (перезапуске) сервера.

Возвращаемое значение:

- *Код ошибки*

4.3.5.12. Чтение режима поиска IP-крейтов в локальной сети

Формат: `INT LTR_GetIPCrateDiscoveryMode(TLTR *ltr, BOOL *enabled, BOOL *autoconnect)`

Назначение: Получает информацию о текущем состоянии механизма автоматического обнаружения крейтов в локальной сети.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- enabled – указатель на переменную, принимающую флаг разрешения автоматического обнаружения крейтов в локальной сети.
- autoconnect – указатель на переменную, принимающую флаг автоматического подключения обнаруженных крейтов.

Примечания:

Подробное описание приведено в примечании к функции *LTR_SetIPCrateDiscoveryMode()*.

Возвращаемое значение:

- *Код ошибки*

4.3.5.13. Установка режима поиска IP-крейтов в локальной сети

Формат: `INT LTR_SetIPCrateDiscoveryMode(TLTR *ltr, BOOL enabled, BOOL autoconnect, BOOL permanent)`

Назначение: Изменяет состояние механизма автоматического обнаружения крейтов в локальной сети.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- enabled – флаг разрешения автоматического обнаружения крейтов в локальной сети.
- autoconnect – флаг автоматического подключения обнаруженных крейтов.
- permanent – если TRUE, то изменения сохраняются в файле конфигурации LTR-сервера.

Примечания:

Автоматическое обнаружение IP-крейтов в локальной сети осуществляется посредством периодической широковещательной посылки по протоколу UDP при наличии соответствующей функции во встроенном ПО крейт-контроллера. Поиск может работать только в пределах локального сегмента сети (проходит через коммутаторы, но не через маршрутизаторы).

Если enabled = FALSE, то автоматическое обнаружение крейтов отключено.

Если enabled = TRUE, autoconnect = FALSE, то обнаруженные крейты, IP-адресов которых не было в списке известных крейтов, добавляются в него с установленным в "0" флагом CRATE_IP_FLAG_AUTOCONNECT.

Если enabled = TRUE, autoconnect = TRUE, то обнаруженные крейты, IP-адресов которых не было в списке известных крейтов, добавляются в него с установленным в "1" флагом CRATE_IP_FLAG_AUTOCONNECT, после чего немедленно инициируется установление соединения.

Возвращаемое значение:

- *Код ошибки*

4.3.5.14. Чтение уровня журнализации

Формат: `INT LTR_GetLogLevel(TLTR *ltr, INT *level)`

Назначение: Получает информацию о текущем уровне журнализации работы LTR-сервера.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- level – указатель на переменную, принимающую значение уровня журнализации.

Примечания:

Список уровней журнализации приведен в описании функции *LTR_SetLogLevel()*.

Возвращаемое значение:

- *Код ошибки*

4.3.5.15. Установка уровня журнализации

Формат: `INT LTR_SetLogLevel(TLTR *ltr, INT level, BOOL permanent)`

Назначение: Устанавливает уровень журнализации работы LTR-сервера.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- level – новый уровень журнализации:
  - 0 Критические (запуск, останов, фатальные ошибки);
  - 1 Ошибки;
  - 2 Предупреждения;
  - 3 Информационные сообщения;
  - 4 Подробные информационные сообщения;
  - 5 Отладочные сообщения (минимум);
  - 6 Отладочные сообщения (средний уровень);
  - 7 Отладочные сообщения (максимум).

  В журнал записываются события с уровнем важности, численно меньшим (т.е. важнее) или равным текущему установленному уровню журнализации.
- permanent – если TRUE, то изменения сохраняются в файле конфигурации LTR-сервера.

Возвращаемое значение:

- *Код ошибки*

4.3.5.16. Перезапуск LTR-сервера

Формат: `INT LTR_ServerRestart(TLTR *ltr)`

Назначение: Инициирует полный сброс и повторную инициализацию программы ltrserver.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Примечания:

Функция аналогична соответствующей команде, доступной через графический интерфейс программы. По этой команде LTR-сервер останавливает все свои операции, разрывает все соединения с крейтами и клиентскими приложениями (в том числе с приложением, вызвавшим эту функцию), перечитывает с диска файл конфигурации и запускается заново.

После успешного выполнения данной функции связь с сервером теряется, поэтому после ее вызова соединение следует закрыть, вызвав функцию *LTR_Close()*.

Соединение с сервером после перезапуска осуществляется обычным образом (как новое). Перед этим рекомендуется выдержать паузу не менее 1 с.

Возвращаемое значение:

- *Код ошибки*

4.3.5.17. Останов LTR-сервера

Формат: `INT LTR_ServerShutdown(TLTR *ltr)`

Назначение: Завершает процесс ltrserver.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.

Примечания:

Функция аналогична закрытию приложения ltrserver штатными средствами (LTR-сервер посылает себе системное сообщение WM_CLOSE).

После успешного выполнения данной функции связь с сервером теряется, поэтому после ее вызова соединение следует закрыть, вызвав функцию *LTR_Close()*.

Возвращаемое значение:

- *Код ошибки*

4.3.5.18. Чтение номера версии LTR-сервера

Формат: `INT LTR_GetServerVersion(TLTR *ltr, DWORD *version)`

Назначение: Возвращает номер версии программы ltrserver.

Передаваемые параметры:

- ltr – указатель на экземпляр структуры *TLTR*.
- version– указатель на переменную, принимающую номер версии.

Примечания:

Номер версии представляет собой беззнаковое 32-битное целое число, каждый байт которого соответствует одной части номера версии, отделяемой в текстовой записи точкой, причем старший байт соответствует главному номеру. Например, 0x01050202 соответствует версии 1.5.2.2.

Функция доступна в ltrserver версии 1.5.2.2 и выше.

Возвращаемое значение:

- *Код ошибки*