Библиотека пользовательского интерфейса модуля LTR43

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.5

Сентябрь 2012г.

Автор руководства:
Милованов А.Н., Борисов А.В.

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

тел.: (495) 785-95-25
факс: (495) 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 Sistems, *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	06.04.2006	Первая доступная для пользователя ревизия
1.0.1	29.06.2006	Выполнен потоковый ввод из портов цифрового ввода-вывода
1.0.2	18.10.2006	Исправлены некоторые ошибки в этом Руководстве
1.0.3	22.11.2006	Произведены изменения в формате структуры описания модуля и именах полей в целях унификации программного интерфейса всех модулей
1.0.4	21.03.2007	Введено описание функции *LTR43_IsOpened()*
1.0.5	18.09.2012	Добавлено описание новых возможностей для обмена по RS485 в прошивке AVR 1.6. Добавлено описание кодов команд протокола обмена с модулем.

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 Основные принципы работы с библиотекой пользовательского интерфейса модуля LTR43.

Модуль LTR43 предназначен для цифрового асинхронного ввода-вывода TTL-сигналов, для синхронизации сбора данных в одном крейте или в многокрейтовой системе по внутренним или внешним синхросигналам, а также для реализации обмена данными по интерфейсу RS485. Подробно о структуре модуля и всех его возможностях написано в документе «Крейтовая система LTR. Руководство пользователя». В данном Руководстве речь пойдет о программировании модуля посредством вызова функций, содержащихся в библиотеке пользовательского интерфейса.

На плате модуля установлен микроконтроллер **AVR Atmega 8515**, осуществляющий общее управление модулем, контролирующий обмен информацией с крейт-контроллером, с портами ввода вывода модуля и обмен данными по интерфейсу RS485. Управляющая программа микроконтроллера записывается в его Флэш-память при изготовлении модуля. В этой памяти также содержится идентификационная информация (серийный номер, версия управляющей программы микроконтроллера, дата ее создания, имя модуля). С точки зрения программного обеспечения связь с микроконтроллером модуля и обмен информацией с ним осуществляются при помощи библиотеки пользовательских функций.

Последовательность применения этих функций сходна с использованием аналогичных функций в программном обеспечении других модулей системы LTR. В общих чертах эта последовательность выглядит следующим образом:

- Инициализация интерфейсного канала связи с модулем, установка настроек по умолчанию;
- Установка параметров работы (конфигурация портов ввода-вывода, интерфейса RS485, меток) и загрузка их в память микроконтроллера AVR ATmega 8515;
- Получение данных с входных цифровых линий ввода-вывода и отправка данных на выходные линии. Обмен данным по интерфейсу RS485. Управление генерацией меток.
- Закрытие интерфейсного канала связи с модулем.

2 Библиотека интерфейса пользователя модуля LTR43, общие сведения.

Библиотека функций пользовательского интерфейса содержит следующие основные компоненты: **структуру описания модуля** и **набор функций для связи и работы с модулем**.

2.1 Структура описания модуля.

Структура описания модуля (тип **TLTR43**) предназначена для инициализации, хранения и изменения данных о конфигурации модуля в рамках создаваемой программы. При помощи полей этой структуры задается конфигурация портов ввода-вывода, определяются параметры интерфейса RS485 и типы синхрометок. Перед вызовом функции конфигурации модуля

LTR43_Config() следует обязательно заполнить поля структуры описания модуля. В противном случае модуль может выдавать неверные данные.

2.2 Функции пользовательской библиотеки (общие сведения).

Функции библиотеки пользователя **ltr43api.dll** предназначены для конфигурирования, программирования, сбора данных и диагностики модуля.

2.2.1 Классификация функций пользовательской библиотеки.

Функции, входящие в состав пользовательской библиотеки модуля LTR43, можно разделить на следующие группы:

- Функции инициализации и открытия;
- Функции конфигурации модуля;
- Функции сбора данных;
- Функция обмена данных по интерфейсу RS485.
- Функция закрытия;
- Вспомогательные функции.

К функциям инициализации и открытия относятся функции *LTR43_Init()* и *LTR43_Open()*. Их следует вызывать в первую очередь. Они предназначены для того, чтобы заполнить поля структуры описания модуля значениями по умолчанию, открыть интерфейсный канал связи с модулем, и выполнить необходимые проверки. Без вызова этих функций дальнейшая работа с модулем невозможна.

Функция конфигурирования – это *LTR43_Config()*. Эта функция передает все конфигурационные параметры, записанные в полях структуры описания модуля, в оперативную память микроконтроллера модуля. После выполнения указанных функций устройство готово к сбору данных.

Функции сбора данных: *LTR43_ReadPort()*, *LTR43_StartStreamRead()*, *LTR43_StopStreamRead()*, *LTR43_Recv()*, *LTR43_WritePort()*, *LTR43_WriteArray()*. Они предназначены для обмена данными с портом ввода-вывода модуля, а также для проверки правильности полученных данных.

Функция для работы с интерфейсом RS-485: *LTR43_RS485_Exchange()*. Эта функция осуществляет обмен пакетами данных с другими устройствами по интерфейсу RS-485.

Функции управления метками:

LTR43_StartSecondMark(),
LTR43_StopSecondMark(),
LTR43_MakeStartMark(),

При помощи этих функций производится запуск и остановка генерации секундной метки, а также формирование метки «СТАРТ».

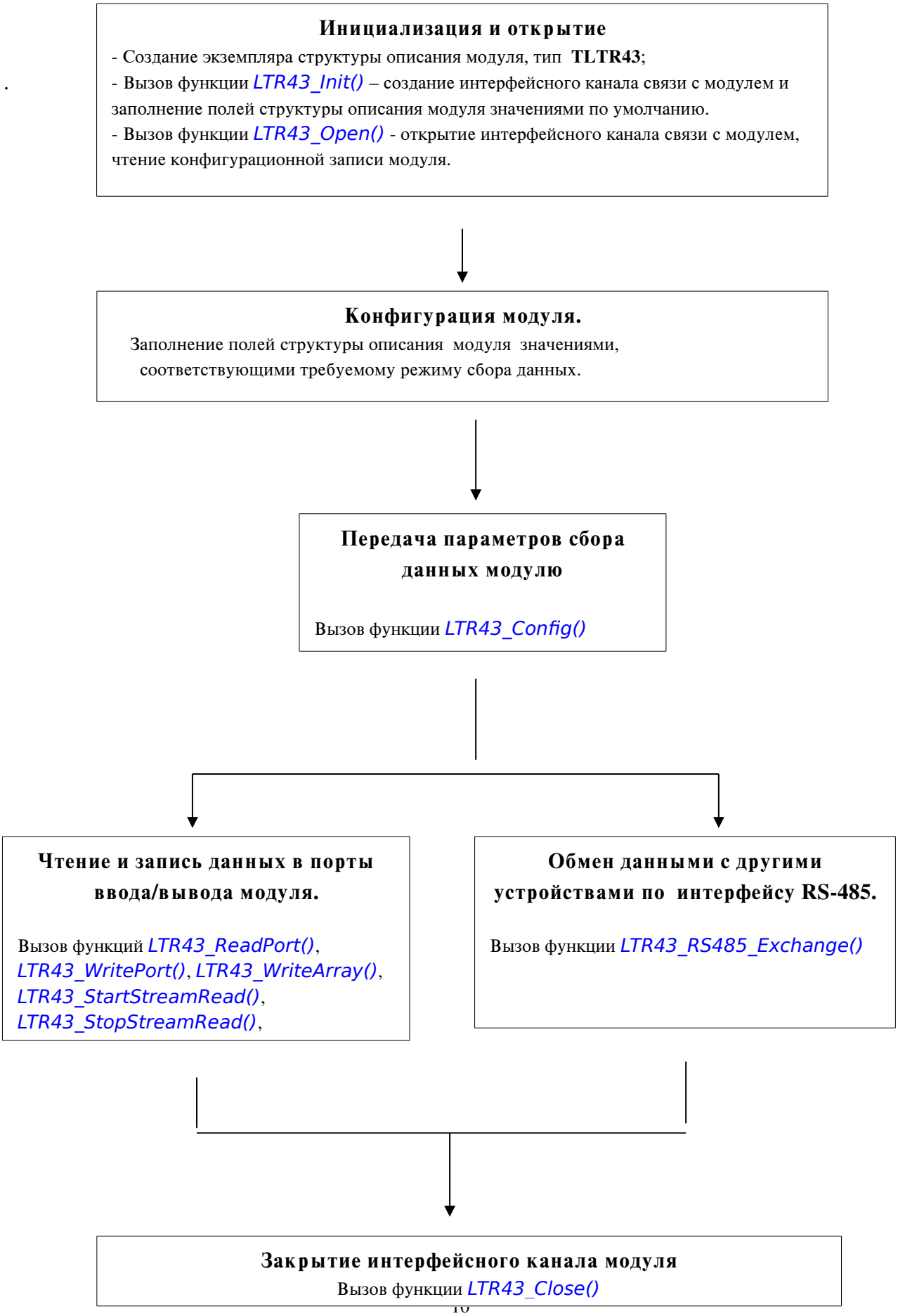
Вспомогательные функции: *LTR43_GetErrorString()* возвращает строку с описанием ошибки, соответствующую коду ошибки; *LTR43_ReadEEPROM()* выполняет

чтение информации из указанной ячейки пользовательского ППЗУ, *LTR43_WriteEEPROM()* выполняет запись информации в указанную ячейку пользовательского ППЗУ.

Функция закрытия: *LTR43_Close()*. Вызывается при окончании работы с модулем с целью закрытия интерфейсного канала. Для корректного завершения работы с модулем следует обязательно вызывать данную функцию.

2.2.2 Типичная последовательность написания программы.

Рис. 1



При программировании модуля следует придерживаться указанной выше схемы.
Далее будут подробно рассмотрены компоненты пользовательской библиотеки **ltr43api.dll.**

3 Подробное описание библиотеки ltr43api.

Для получения возможности вызова интерфейсных функций библиотеки ***ltr43api.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 **ltr43api.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\ltr43api.lib** и **LTR\INCLUDE\ltr43api.h**;

  Microsoft Visual C++ :
  - подключить к проекту файлы **LTR\LIB\MSVC\ltr43api.lib** и **LTR\INCLUDE\ltr43api.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 можно писать свою программу, вызывая соответствующие интерфейсные функции dll-библиотеки.

3.1 Структура описания модуля.

Поля данной структуры содержат информацию о направлениях линий портов ввода-вывода модуля, режиме работы интерфейса RS485 и о конфигурации меток. Также в эту структуру считывается конфигурационная запись модуля при вызове функции **LTR43_Open().** Определение структуры приводится ниже:

```
typedef struct
{
    INT size;   // размер структуры
    TLTR Channel;
    double StreamReadRate;
    struct
    {
        INT Port1; // направление линий ввода/вывода группы 1 (IO1-IO8)
        INT Port2; // направление линий ввода/вывода группы 2 (IO9-IO16)
        INT Port3; // направление линий ввода/вывода группы 3 (IO17-IO24)
        INT Port4; // направление линий ввода/вывода группы 4 (IO25-IO32)
    } IO_Ports;

    struct
    {
        INT FrameSize;      // Кол-во бит в кадре
        INT Baud;          // Скорость обмена в бодах
        INT StopBit;      // Кол-во стоп-бит
        INT Parity;          // Включение бита четности
        INT SendTimeoutMultiplier; // Множитель таймаута отправки
        INT ReceiveTimeoutMultiplier; // Множитель таймаута приема подтверждения
    } RS485; // Структура для конфигурации RS485

    struct
    {
        INT SecondMark_Mode; //Тип сек. метки 0 - внутр., 1-внутр.+выход, 2-внешняя
        INT StartMark_Mode; //Тип метки «СТАРТ» 0 - внутр., 1-внутр.+выход, 2-внешняя
    } Marks;  // Структура для работы с временными метками
```

```
    TINFO_LTR43 ModuleInfo;
} TLTR43, *PTLTR43; // Структура описания модуля
```

Далее приводится описание полей этой структуры

Таблица 1

Поле		Тип	Описание
size		**INT**	Объем памяти, выделяемой под структуру. Заполняется функцией *LTR43_Init()* - пользователю самому заполнять не нужно.
Channel		**TLTR**	Подструктура, представляющая собой описание интерфейсного канала связи с модулем.
StreamReadRate		**double**	Частота выдачи данных при потоковом вводе в Герцах
IO_Ports			Подструктура, определяющая направления линий портов ввода/вывода
Поля структуры	**Port1**	**INT**	Определяет направление линий порта ввода/вывода 1. «0» - вход, «1» - выход.
	Port2	**INT**	Определяет направление линий порта ввода/вывода 2. «0» - вход, «1» - выход.
	Port3	**INT**	Определяет направление линий порта ввода/вывода 3. «0» - вход, «1» - выход.
	Port4	**INT**	Определяет направление линий порта ввода/вывода 4. «0» - вход, «1» - выход.
RS485			Подструктура, определяющая параметры обмена данными по интерфейсу RS-485.
Поля структуры	**FrameSize**	**INT**	Размер кадра при передаче и приеме по интерфейсу RS485. Допустимое значение – от 5 до 9, что соответствует количеству 5-9 значащих бит в кадре.
	Baud	**INT**	Скорость обмена в бодах (бит/с). Возможны значения от 2400 до 115200.
	StopBit	**INT**	Количество стоп-бит в отправляемом слове. «0» - 1 стоп-бит. «1» - 2 стоп-бита.
	Parity	**INT**	Определяет режим проверки четности при обмене данными (выполняется аппаратно микроконтроллером модуля). «0» - проверка четности отключена, «1» - «четная»

			проверка. «2» – «нечетная» проверка. Если проверка четности включена, то к отправляемым кадрам микроконтроллером добавляется бит четности, а в принимаемых кадрах он считывается и анализируется.
	SendTimeoutMultiplier	**INT**	Множитель таймаута отправляемых слов
	ReceiveTimeoutMultiplier	**INT**	Множитель таймаута принимаемых подтверждений
Marks			Подструктура, определяющая режим формирования временнЫх меток
Поля структуры	**SecondMark_Mode**	**INT**	Тип секундной метки. «0» - внутренняя, «1» - внутренняя с трансляцией на выход, «2» - внешняя.
	StartMark_Mode	**INT**	Тип метки «СТАРТ». «0» - внутренняя, «1» - внутренняя с трансляцией на выход, «2» - внешняя.
ModuleInfo			Идентификационная информация модуля. Тип – **TINFO_LTR43.** Данный тип представляет собой структуру. Ее описание приводится ниже в этой главе настоящего Руководства

Для задания режима работы модуля пользователю необходимо самостоятельно (вручную) определить поля этой структуры. В конце данного Руководства приводятся примеры задания конфигурации.

Идентификационная информация модуля:

```
/* Информация о модуле */
typedef struct
{
    CHAR Name[16]; //Название модуля "LTR43"
    CHAR Serial[24]; //Серийный номер модуля
    CHAR FirmwareVersion[8];// Версия прошивки AVR
    CHAR FirmwareDate[16];  // Дата создания прошивки AVR
} TINFO_LTR43,*PTINFO_LTR43;
```

3.2 Описание функций библиотеки ltr43.dll.

3.2.1 Функции инициализации и открытия модуля

Формат: INT LTR43_Init(TLTR43 hnd)

Параметр: hnd – указатель на структуру описания модуля (тип **PTLTR43**)

Описание: Выполняет инициализацию интерфейсного канала связи с модулем и заполнение полей структуры описания модуля значениями по умолчанию. Ниже приводятся значения, которыми данная функция инициирует поля указанной структуры:

```
 hnd->size=sizeof(TLTR43);
//По умолчанию все линии устанавливаются на вход
hnd->IO_Ports. Port1=0;
hnd->IO_Ports. Port2=0;
hnd->IO_Ports. Port3=0;
hnd->IO_Ports. Port4=0;

hnd->StreamReadRate=15000; // По умолчанию частота выдачи при потоковом вводе – 15 кГц

 //Параметры RS485 по умолчанию:
hnd->RS485.FrameSize=8;   // 8 бит в кадре
hnd->RS485.Baud=2400;     // Скорость обмена – 2400 бод
hnd->RS485.StopBit=0;     // 1 стоп-бит
hnd->RS485.Parity=0;      // Проверка на четность отключена
//Множители таймаутов:
hnd->RS485.SendTimeoutMultiplier=10;      // Множитель таймаута передачи - 10 мс
hnd->RS485.ReceiveTimeoutMultiplier=10;   // Множитель таймаута приема  -  10 мс

// Режимы синхрометок по умолчанию
 hnd->Marks.SecondMark_Mode=0;   // Режим секундой метки - внутренняя
 hnd->Marks.StartMark_Mode=0;    // Режим метки «СТАРТ»-внутренняя
```

При инициализации все поля идентификационной структуры ModuleInfo будут содержать только «\0»

Возвращаемое значение: код ошибки**.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_Open(PTLTR43 hnd, DWORD net_addr, WORD net_port, const CHAR *crate_sn, INT slot_num)

Параметры:

- **hnd** – указатель на структуру описания модуля (тип **PTLTR43**)
- **net_addr** – сетевой адрес LTR-Сервера в формате 32-х битного беззнакового целого. Если ip-адрес «a.b.c.d», то поле **net_addr = (a<<24)|(b<<16)|(c<<8)|(d<<0).** Например, **net_addr** для адреса 127.0.0.1 будет выглядеть следующим образом: 0x7F000001. Необходимо помнить, что все компоненты адреса должны иметь значение, не превосходящее 255. Если LTR-сервер запущен на том же компьютере что и пользовательская программа, то в качестве сетевого адреса можно использовать константу *SADDR_DEFAULT*.
- **net_port** – сетевой порт LTR-Сервера. Если LTR-сервер запущен на том же компьютере что и пользовательская программа, то в качестве **net_port** можно использовать константу *SPORT_DEFAULT*.
- **crate_sn** – серийный номер **крейта (не модуля!!!).**
- **slot_num** – номер слота, в котором расположен модуль (***нумерация с единицы!***)

Описание: Функция открывает интерфейсный канал связи с модулем, выполняет необходимые проверки, а также считывает из ППЗУ модуля его идентификационную запись. После работы функции в соответствующих полях структуры описания модуля будут находиться: версия управляющей программы AVR, дата создания управляющей программы AVR, имя модуля и серийный номер модуля.

Возвращаемое значение: код ошибки, тип INT**.** Если “0” – функция выполнилась без ошибок. Если возвращаемое значение равно –10, то это предупреждение, что интерфейсный канал уже открыт. Тем не менее функция выполнилась успешно и можно продолжать работу. Однако дальнейшая работа модуля может быть некорректной. Рекомендуется разобраться в причинах предупреждения и закрыть открытый ранее канал.

Формат: INT LTR43_IsOpened(PTLTR43 hnd)

Параметр: **hnd** – указатель на структуру описания модуля, тип TLTR43

Описание: Функция позволяет отслеживать состояние соединения с модулем: если возвращаемый результат отличен от 0, то соединения нет.

Возвращаемое значение: Если “0” – интерфейсный канал связи с модулем создан и открыт. Если значение ненулевое - канал не создан

Формат: INT LTR43_Close(PTLTR43 hnd)
Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR43**
Описание: Выполняет закрытие интерфейсного канала связи с модулем принудительную остановку генерации секундных меток. Эту функцию следует вызывать всегда перед окончанием работы с модулем.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

3.2.2 Функции для конфигурации модуля

Формат: INT LTR43_Config(PTLTR43 hnd)
Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR43**
Описание: функция передачи модулю конфигурационной информации, определенной ранее в полях структуры описания модуля. Информация о направлениях линий портов ввода-вывода, параметрах RS-485, о конфигурации меток загружается в оперативную память микроконтроллера модуля. Перед использованием этой функции поля структуры описания модуля должны быть заполнены требуемыми значениями. Обмен данными с портом ввода-вывода, передачу данных по RS-485, а также генерацию временнЫх меток следует осуществлять *только после выполнения этой функции*.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

3.2.3 Функции ввода-вывода

Формат: INT LTR43_ReadPort(PTLTR43 hnd, DWORD *InputData)
Параметры: hnd – указатель на структуру описания модуля **PTLTR43;** *OutputData - указатель на 32-битное слово, показывающее уровень сигналов на каждой из входных линий
Описание: Выполняет чтение значений сигналов с линий портов ввода-вывода модуля. Для выполнения этой операции необходимо, чтобы интересующие порты были настроены на ВХОД. В противном случае информация о значениях сигналов на линиях этих портов будет неверной. После работы функции значение, адресуемое указателем **OutputData,** содержит информацию о сигналах, считанных со входных линий портов ввода-вывода. Подробнее о формате этого значения см. в *главе 4.4*. настоящего Руководства.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_StartStreamRead(PTLTR43 hnd)
Параметр: hnd – указатель на структуру описания модуля **PTLTR43;**
Описание: Запускает потоковое (непрерывное) чтение значений сигналов с линий портов ввода-вывода модуля. Для выполнения этой операции необходимо, чтобы интересующие порты были настроены на ВХОД. В противном случае информация о значениях сигналов на линиях этих портов будет неверной. Модуль начнет выдавать пакеты данных с частотой, определяемой полем **StreamReadRate** структуры описания модуля. Подробнее о непрерывном чтении данных см. в гл. 4.5 настоящего Руководства.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_Recv(PTLTR43 hnd, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)
Параметр: hnd – указатель на структуру описания модуля **PTLTR43;** ***data –** указатель на массив с входными данными; ***tmark –** указатель на массив полученных секундных меток и меток СТАРТ; **size** – количество слов данных в запрашиваемом массиве; **timeout –** время миллисекундах, в течение которого функция будет ожидать получения заказанного количества слов
Описание: Выполняет получение массива слов из модуля размеров **size** . Полученные слова при выходе из функции содержатс в массиве, адресуемом указателем **data**. Указатель **tmark** адресует массив, содержащий метки (секундные и СТАРТ), если таковые были получены. **Если элементы этого массива не используются в программе, то в качестве значения параметра tmark можно использовать NULL**. Параметр **timeout** определяет время в миллисекундах, в течение которого функция будет ожидать получения заказанного количества слов. Если требуемое количество получено до истечения таймаута, то функция завершается немедленно. Если по истечении таймаута не было получено требуемое количество слов, то функция все равно завершатся. Возвращаемое значение функции – это полученное количество слов от модуля. Если же возвращаемое значение отрицательно, то это свидетельствует об ошибочном завершении. В этом случае следует идентифицировать ошибку функцией *LTR43_GetErrorString()*. Данную функцию целесообразно использовать при потоковом вводе с цифровых линий. *Примечание: Описание этой функции соответствует описанию функции **LTR_Recv()** библиотеки крейт-контроллера ltrapi.dll.*
Возвращаемое значение: Если значение положительное или 0, то оно соответствует количеству слов, принятых от модуля. Если отрицательное, то оно представляет собой код ошибки.

Формат: INT LTR43_ProcessData(PTLTR43 hnd, const DWORD *src, DWORD *dest, DWORD *size)

Параметры:

hnd – указатель на структуру описания модуля **PTLTR43;**

***src –** указатель на массив с исходными словами данных, полученными из модуля. Каждая пара слов содержит соответственно 16 старших и 16 младших бит значения, считанного с портов ввода/вывода.

***dest –** массив сформированных 32-битных слов данных

***size** – указатель на количество слов данных в массиве

Описание: Выполняет проверку целостности данных, поступающих при потоковом чтении из портов ввода/вывода, а также формирует готовые 32-битные слова, соответствующие считанным значениям цифровых линий. О формате слов подробнее см. в описании однократного чтения из порта ввода/вывода, в *главе 4.4*. настоящего Руководства.

При потоковом чтении пользователю необходимо самостоятельно получать слова-данные из модуля при помощи функции *LTR43_Recv()*, а затем передавать полученные данные функции *LTR43_ProcessData()*, после работы которой массив, адресуемый указателем ***dest**, будет содержать сформированные 32-битные слова. Параметр ***size** при входе в функцию указывает на размер исходного массива, при выходе из функции – выходного. Подробнее о непрерывном чтении данных и, в частности, о применении функции *LTR43_ProcessData()* см. гл. 4.5 настоящего Руководства.

Если функция выявляет сбой в последовательности значений счетчика отправляемых слов-данных, то ее выполнение немедленно прерывается и возвращается код ошибки.

Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_StopStreamRead(PTLTR43 hnd)

Параметры: hnd – указатель на структуру описания модуля **PTLTR43;**

Описание: Останавливает потоковое (непрерывное) чтение значений сигналов с линий портов ввода-вывода модуля.

Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_WritePort(PTLTR43 hnd, DWORD OutputData)

Параметры: hnd – указатель на структуру описания модуля **PTLTR43;**

OutputData - 32-битное слово, определяющее уровень сигнала на каждой из выходных линий

Описание: Выполняет однократную запись 32 битного слова данных в порты ввода-вывода модуля. Для того чтобы нужные уровни сигналов появились на соответствующих линиях ввода-вывода, необходимо, чтобы эти линии были настроены на ВЫХОД. В противном случае значение сигнала на этих линиях не изменится. О формате параметра **OutputData** см. подробно в *главе 4.2.* настоящего Руководства.

Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_WriteArray(PTLTR43 hnd, DWORD *OutputArray, BYTE ArraySize)
Параметры: **- hnd –** указатель на структуру описания модуля **PTLTR43;** **- OutArray -** Указатель на массив 32-битных слов данных, подлежащих отправке в порты ввода-вывода модуля. Слова выводятся в порты последовательно, с минимально возможным интервалом времени для системы LTR. Каждое слово в массиве аналогично параметру **OutputData** для функции *LTR43_WritePort()*. **- ArraySize –** Количество элементов массива **OutputArray**. Может принимать значения от 1 до 255.
Описание: Выполняет вывод 32-битных элементов массива **OutputArray** в порты ввода-вывода модуля. Для того чтобы нужные уровни сигналов появились на соответствующих линиях, необходимо, чтобы порты, соответствующие этим линиям, были настроены на ВЫХОД. В противном случае значение сигнала на этих линиях не изменится. Подробнее о выводе массива см. в *главе 4.3.* настоящего Руководства.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

3.2.4 Функции для формирования секундных меток

Формат: INT LTR43_StartSecondMark(PTLTR43 hnd)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR43**
Описание: Выполняет запуск генерации секундной метки.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_StopSecondMark(PTLTR43 hnd)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR43**
Описание: Выполняет остановку генерации секундной метки.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_MakeStartMark(PTLTR43 hnd)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR43**
Описание: Выполняет однократную генерацию метки «СТАРТ».
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

3.2.5 Функции управления интерфейсом RS485

Формат: INT LTR43_RS485_Exchange(PTLTR43 hnd, const SHORT *PackToSend, SHORT *ReceivedPack, INT OutPackSize, INT InPackSize)

Параметры:

- **hnd –** указатель на структуру описания модуля типа **PTLTR43**
- ***PackToSend –** Массив слов на передачу из **OutPackSize** слов. В каждом слове значащими являются от 5 до 9 младших бит (в зависимости от поля RS485.FrameSize описателя модуля);
- ***ReceivedPack –** Массив размером в **InPackSize** слов, в который будет записаны принятые слова по RS485;
- **OutPackSize** – количество слов в отправляемом пакете, диапазон значений - 1..10 (от 0 до 256 при работе с прошивкой 1.6 и выше);
- **InPackSize** – количество слов в принимаемом пакете-подтверждении, диапазон значений- 1..10 (от 0 до 256 при работе с прошивкой 1.6 и выше);

Описание: Выполняет обмен данными по протоколу RS-485. Пакет слов, элементы которого соответствуют элементам массива **PackToSend**[], в количестве, равном **InPackSize** последовательно отправляется по интерфейсу RS485. Затем модуль ожидает ответного пакета-подтверждения. Количество слов в этом пакете должно быть равно **InPackSize**. Полученные слова записываются в массив **ReceivedPack[].** Процесс отправления пакета ограничен по времени программно задаваемым таймаутом (при работе с прошивкой до 1.15).. Если в течение этого таймаута отправить все слова пакета не удалось, работа функции завершается соответствующей ошибкой. Аналогичным образом используется таймаут при приеме входящего пакета. При этом, если за указанный таймаут не будет приняты все **InPackSize** пакетов, то функция вернет ошибку.

Более подробно об обмене данными по интерфейсу RS-485 при помощи модуля LTR43 см. в *главе 6* настоящего Руководства.

Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: **INT LTR43_RS485_ExchangeEx(PTLTR43 hnd, const SHORT *PackToSend, SHORT *ReceivedPack, INT OutPackSize, INT InPackSize, DWORD flags, INT *ReceivedSize)**

Параметры:

- **hnd –** указатель на структуру описания модуля типа **PTLTR43**
- ***PackToSend –** Массив слов на передачу из **OutPackSize** слов. В каждом слове значащими являются от 5 до 9 младших бит (в зависимости от поля **RS485.FrameSize** описателя модуля);
- ***ReceivedPack –** Массив размером в **InPackSize** слов, в который будут записаны принятые слова по RS485;
- **OutPackSize** – количество слов в отправляемом пакете, диапазон значений — 1..10 (от 0 до 256 при работе с прошивкой 1.6 и выше);
- **InPackSize** – Количество слов в принимаемом пакете-подтверждении, диапазон значений- 1..10 (от 0 до 256 при работе с прошивкой 1.6 и выше);
- **flags** — Флаги, определяющие поведение функции.
- **ReceivedSize**— в данной переменной сохраняется реально принятое количество слов.

Описание: Выполняет обмен данными по протоколу RS-485. В отличие от функции *LTR43_RS485_Exchange()*, функция позволяет обработать ситуацию, если в ответе пришло меньше слов, чем **InPackSize**. В этом случае функция не возвращает ошибку, а сохраняет в массив **ReceivedPack** все принятые слова и возвращает в переменной **ReceivedSize** реально принятое количество слов.

В случае, если в N-ом принятом слове будет ошибка четности или кадра, функция вернет ошибку, но сохранит первые N-1 слов в выходной массив и запишет N-1 в **ReceivedSize**.

Так же функция позволяет осуществлять обмен, при котором конец принимаемого пакета определяется по превышению таймаута между принятыми словами. Для этого необходимо передать флаг **LTR43_RS485_FLAGS_USE_INTERVAL_TOUT**.

Внимание: для корректной работы при приеме неполного кадра необходима версия прошивки AVR не ниже чем 1.16. Для предыдущей версии, если по RS485 придет не полный пакет, то контроллер будет ожидать прихода окончания до сброса контроллера.

Возвращаемое значение: код ошибки

Формат: INT LTR43_RS485_SetResponseTout(PTLTR43 hnd, DWORD tout)

Параметры:

- **hnd –** указатель на структуру описания модуля типа **PTLTR43**
- **tout** – постоянная составляющая таймаута на прием ответа по RS485 в милисекундах

Описание: Функция устанавливает постоянную составляющую таймаута на ответ по RS485. Постоянная составляющая может быть установлена только в случае, если версия прошивки контроллера модуля не меньше 1.6 (в противном случае функция вернет ошибку LTR_ERROR_UNSUP_BY_FIRM_VER).

Возвращаемое значение: код ошибки

Формат: INT LTR43_RS485_SetIntervalTout(PTLTR43 hnd, DWORD tout)

Параметры:

- **hnd –** указатель на структуру описания модуля типа **PTLTR43**
- **mks_tout** – максимально допустимый интервал следования между словами по RS485 в микросекундах (от 1750 мкс до 63750 мкс).

Описание: Функция устанавливает максимально допустимый интервал следования между словами по RS485. Превышение этого интервала считается признаком конца кадра, если в функцию *LTR43_RS485_ExchangeEx()* был передан флаг LTR43_RS485_FLAGS_USE_INTERVAL_TOUT. В противном случае данный таймаут не учитывается. В отличие от других таймаутов, данный таймаут устанавливается в микросекундах. Однако реально значение устанавливается с шагом 250 мкс с округлением в большую сторону.

Функция доступна только если версия прошивки контроллера модуля не меньше 1.6 (в противном случае функция вернет ошибку LTR_ERROR_UNSUP_BY_FIRM_VER).

Возвращаемое значение: код ошибки

Формат: INT LTR43_RS485_SetTxActiveInterval(PTLTR43 hnd, DWORD start_of_packet, DWORD end_of_packet)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR43** - **start_of_packet** – интервал времени перед передачей пакета по RS485, в течение которого модуль активирует передатчик до передачи данных (от 0 мкс до 63750 мкс). - **end_of_packet** – интервал времени после завершения передачи пакета по RS485, в течение которого модуль оставляет активным передатчик (от 0 мкс до 63750 мкс).
Описание: Функция устанавливает интервалы времени перед и после передачи пакета данных, в течение которых модуль включает передатчик. Данные интервалы устанавливается в микросекундах. Однако реально значение устанавливается с шагом 250 мкс с округлением в большую сторону. Подробнее см. главу 6.3. По-умолчанию эти интервалы равны 0 мкс. Функция доступна только если версия прошивки контроллера модуля не меньше 1.6 (в противном случае функция вернет ошибку LTR_ERROR_UNSUP_BY_FIRM_VER).
Возвращаемое значение: код ошибки

3.2.6 Функции для работы с ППЗУ модуля

Формат: INT LTR43_ReadEEPROM(PTLTR43 hnd, INT Address, BYTE *val)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR43;** - **Addres –** адрес ячейки ППЗУ, из которой следует считать байт; - ***val –** указатель на считанное из указанной ячейки значение
Описание: Выполняет чтение байта из ячейки пользовательского ПЗУ с адресом, определяемым параметром **Address**.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR43_WriteEEPROM(PTLTR43 hnd, INT Address, BYTE val)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR43;** - **Addres –** адрес ячейки ППЗУ, из которой следует считать байт; - **val –** байт, который следует записать в ППЗУ
Описание: Выполняет запись байта в ячейку пользовательского ПЗУ с адресом, определяемым параметром **Address**.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: LPCSTR LTR43_GetErrorString(INT Error_Code)
Параметры: - **Error_Code** – код ошибки;
Описание: Возвращает строку, описывающую ошибку, соответствующую коду **Error_Code**
Возвращаемое значение: строка, соответствующая данному коду ошибки

Внимание!!! Следует не путать типы данных: структура описания модуля (тип TLTR43) и ***УКАЗАТЕЛЬ*** на структуру описания модуля (тип PTLTR43).

4 Организация обмена данными с портами ввода-вывода.

4.1 Конфигурация линий ввода-вывода

Каждый из 4-х портов ввода-вывода может быть настроен как на вход, так и на выход. Перед обменом информацией необходимо определить направления линий портов. Если этого не сделать, обмен данными будет некорректным. Определение направлений линий портов ввода-вывода осуществляется при помощи заполнения соответствующих полей структуры описания модуля и последующего вызова функции *LTR43_Config()*.

Структура описания модуля содержит подструктуру **IO_Ports**, определенную следующим образом:

```
struct
{
    INT Port1; // направление линий ввода/вывода группы 1 (IO1-IO8)
    INT Port2; // направление линий ввода/вывода группы 2 (IO9-IO16)
    INT Port3; // направление линий ввода/вывода группы 3 (IO17-IO24)
    INT Port4; // направление линий ввода/вывода группы 4 (IO25-IO32)
} IO_Ports;
```

В таблице указано соответствие физических линий ввода-вывода портам:

Таблица 1

Группа	Соответствующее поле подструктуры **IO_Ports**	Линии
1	**Port1**	IO1-IO8
2	**Port2**	IO9-IO16
3	**Port3**	IO17-IO24
4	**Port4**	IO25-IO32

Для того чтобы настроить какой-либо порт на ВХОД, нужно присвоить значение «0» соответствующему полю подструктуры **IO_Ports**. Для того чтобы настроить какой-либо порт на ВЫХОД, нужно присвоить значение «1» соответствующему полю подструктуры **IO_Ports**.

Пример конфигурации портов ввода-вывода:

```
TLTR43 conf;                // Объявляем переменную – экземпляр структуры

conf.IO_Ports. Port1=0;     // Порт ввода-вывода 1 настроен на вход
conf.IO_Ports. Port2=1;     // Порт ввода-вывода 2 настроен на выход
conf.IO_Ports. Port3=1;     // Порт ввода-вывода 3 настроен на выход
conf.IO_Ports. Port4=0;     // Порт ввода-вывода 4 настроен на вход
```

После определения полей структуры в соответствии с желаемыми направлениями линий ввода-вывода следует обязательно вызвать функцию *LTR43_Config()* для того чтобы передать эту информацию управляющей программе модуля, которая настроит порты в

соответствии с требуемой конфигурацией. Для нашего примера вызов функции выглядит следующим образом: **LTR43_Config(&conf).**

Напомним, что функция *LTR43_Config()* используется не только для определения направления линий ввода-вывода, но и для конфигурирования иных функций модуля. Это всегда нужно иметь в виду и следить за состоянием полей структуры описания модуля, которые не затрагивались перед данным конкретным применением функции *LTR43_Config()*, потому что их значения тоже буду загружены в память модуля при вызове этой функции.

4.2 Запись 32-битного слова в выходные порты модуля.

Для записи слова данных в порты ввода-вывода модуля LTR43 предназначена функция *LTR43_WritePort()*. Параметр **OutputData** представляет собой 32-битное слово, каждый бит которого определяет значение сигнала для соответствующей линии ввода-вывода. Биты 0..31 параметра **OutputData** соответствуют линиям ввода-вывода IO1..IO32. Значение бита “1” соответствует уровню логической единицы на соответствующей линии, значение “0” – уровню логического нуля.

Например, если все линии ввода-вывода сконфигурированы как выходные, то после вызова функции **LTR43_WritePort(&conf, 0x4518AF03)** на них установятся следующие уровни сигнала:

Таблица 2

Линия	**Уровень сигнала**	**Линия**	**Уровень сигнала**	**Линия**	**Уровень сигнала**	**Линия**	**Уровень сигнала**
IO1	1	**IO 9**	1	**IO 17**	0	**IO 25**	1
IO 2	1	**IO 10**	1	**IO 18**	0	**IO 26**	0
IO 3	0	**IO 11**	1	**IO 19**	0	**IO 27**	1
IO 4	0	**IO 12**	1	**IO 20**	1	**IO 28**	0
IO 5	0	**IO 13**	0	**IO 21**	1	**IO 29**	0
IO 6	0	**IO 14**	1	**IO 22**	0	**IO 30**	0
IO 7	0	**IO 15**	0	**IO 23**	0	**IO 31**	1
IO 8	0	**IO 16**	1	**IO 24**	0	**IO 32**	0

Если какие-либо порты были сконфигурированы на вход, то уровень сигнала на соответствующих им линиях ввода-вывода не изменится независимо от значения параметра **OutputData.**

4.3 Запись массива 32-битных слов в порты модуля.

Для того чтобы последовательно вывести МАССИВ слов в порты ввода-вывода модуля, следует воспользоваться функцией *LTR43_WriteArray()*. Слова, подлежащие выводу, должны быть записаны в массив **OutputArray**[], указатель на который является параметром функции *LTR43_WriteArray()*. Параметр **ArraySize** должен содержать

количество слов, которые необходимо вывести в порты. *Значение этого параметра должно быть не более 255*. Вывод будет производиться в последовательности от слова с младшим индексом массива **OutputArray[]** к слову со старшим индексом. Вывод слов в порты будет произведен с минимально возможным для данной системы интервалом (примерно 85 мкс). Например, чтобы вывести массив слов **Array**, нужно выполнить следующие действия:

- Определить массив. Например,

DWORD Array[5]={ 0x76A1CD54, 0x1C, 0x45CB7A, 0x1, 0xFF259031 }

- Применить функцию **LTR43_WriteArray(&conf, Array, 5)**

При выводе в порты микроконтроллер осуществляет проверку правильности данных, пришедших в модуль из крейт-контроллера. Если обнаружено нарушение целостности данных, микроконтроллер высылает ошибку и функция *LTR43_WritePort()* или *LTR43_WriteArray()* завершается с соответствующей ошибкой. В случае успешного вывода функция возвращает “0”.

4.4 Чтение 32-битного слова из портов ввода-вывода модуля.

Для чтения сигналов с линий портов ввода-вывода, настроенных на вход, предназначена функция *LTR43_ReadPort()*. После выполнения функции параметр-указатель ***InputData** будет адресовать 32-битное значение, считанное из портов модуля. Формат параметра такой же, как и для выходного слова, т.е. каждый бит определяет уровень сигнала соответствующей линии ввода-вывода. Например,

DWORD WordFromPort;

LTR43_ReadPort(&conf, &WordFromPort);

После выполнения функции переменная **WordFromPort** будет содержать считанное значение.

4.5 Потоковое (непрерывное) чтение данных из портов ввода-вывода модуля.

При потоковом вводе считывание значений линий ввода-вывода и отправка слов данных в крейт-контроллер и ПК производится непрерывно с заданной частотой.

Частота выдачи данных при потоковом вводе определяется значением поля **StreamReadRate** структуры описания модуля. Диапазон частот – от 100 Гц до 100 000 Гц. Перед стартом потокового чтение необходимо выполнить функцию *LTR43_Config()*. Следует иметь ввиду, что частота выдачи данных имеет дискретный диапазон значений, поэтому функция *LTR43_Config()* сама определит ближайшее возможное значение к требуемому пользователем и подкорректирует поле **StreamReadRate** структуры описания модуля. Таким образом, после выполнения функции *LTR43_Config()* поле **StreamReadRate** структуры описания модуля будет содержать *фактическую* частоту выдачи данных.

Потоковый ввод запускается функцией *LTR43_StartStreamRead()*.

В случае потокового ввода считывать массивы слов данных из модуля необходимо при помощи функции *LTR43_Recv()*.

Внимание! Процесс потокового ввода данных НЕ ЯВЛЯЕТСЯ абсолютно синхронным! При выполнении процесса возможны колебания периода сбора данных в пределах микросекунды. Выбор частоты выдачи данных определяется как конкретной задачей, так и быстродействием ПК, применяемого для обработки и отображения данных. Если быстродействие ПК недостаточно, то он не будет успевать обрабатывать (отображать, записывать и т.д.) поток данных, собираемых с большой частотой. Поэтому для полноценной работы следует выбирать оптимальную частоту выдачи, определяемую задачами обработки данных и быстродействием конкретного ПК.

Как говорилось выше, для старта непрерывного чтения данных с портов ввода/вывода, необходимо применить функцию *LTR43_StartStreamRead()*. Затем при помощи функции *LTR43_Recv()* следует циклически считывать массивы данных, поступающие из модуля.

Получив порцию данных, необходимо вызвать функцию *LTR43_ProcessData()* для проверки целостности потока данных и для формирования конечных 32-битных значений. Так как порты ввода/вывода в сумме содержат 32 линии ввода-вывода, то каждое считанное значение передается **двумя** словами данных. Это связано с тем, что, в соответствии с интерфейсом системы LTR, каждое слово может передать только 16 бит данных. Таким образом, функция *LTR43_Recv()* должна считывать **удвоенное** число слов-данных из модуля по отношению к желаемому количеству 32-битных слов. Каждая пара сэмплов содержит 16 старших и 16 младших бит считанного из портов значения, причем элементы с четными индексами содержат старшие 16 бит этого значения, а с нечетными – младшие 16 бит (индексация с нуля) . Указатель на считанный массив данных должен передаваться функции *LTR43_ProcessData()* (параметр ***src**). При вызове этой функции параметр ***size** должен указывать на размер именно этого, **исходного,** массива.

После выполнения функции *LTR43_ProcessData()* будет сформирован выходной массив, содержащий готовые 32-битные данные. Этот массив адресуется параметром ***dest**. Размер этого массива **вдвое меньше**, чем размер исходного. Поэтому функция *LTR43_ProcessData()* сама изменит значение, адресуемое параметром ***size**, и после работы функции этот параметр будет указывать на число, равное размеру выходного массива, т.е. **половине** размера исходного массива. Другими словами, после работы функции значение, адресуемое ***size**, уменьшается вдвое.

Одновременно с формированием выходного массива функция выполняет проверку целостности данных в исходном массиве путем проверки 8-битного последовательного счетчика слов. Если последовательность значений счетчика нарушена, функция немедленно завершается с ошибкой.

Рассмотрим пример последовательности действий при потоковом вводе данных из портов ввода/вывода модуля. Предполагается, что экземпляр структуры описания модуля **conf** уже создан, а также иные параметры модуля уже сконфигурированы.

```
/* Определяем массив, в который будем считывать «сырые» данные из модуля: */
   DWORD SourceArray[1000];

/* Определяем массив, в который будут записаны готовые 32-битные данные  */

   DWORD DestArray[500];

   DWORD size // определяет размер как входного, так и впоследствии выходного массивов

   /*   Задание частоты выдачи данных. В нашем примере 10 кГц */
   conf.StreamReadRate=10000;

   /*  Вызываем функцию конфигурации модуля   */
   err=LTR43_Config(&conf);
    if(err) return (err);

   /* Стартуем потоковый сбор данных с частотой выдачи слов 10 кГц. */

   err=LTR43_StartStreamRead(&conf)
    if(err) return (err);

   size=1000;
   /************************************************************
     Если требуется, то условие цикла
   ************************************************************/

   /* Получаем порцию из 1000 слов данных в массив SourceArray[]. Таймаут – 2 с */
   LTR43_Recv(&hltr43, SourceArray, NULL, size, 2000);

   /*Далее вызываем функцию LTR43_ProcessData() */

   err=LTR43_ProcessData(&conf, SourceArray, DestArray, &size);
    if(err)
     {
       LTR43_StopStreamRead((&conf);
       return(err);
    }

   /* После работы функции массив DestArray[] будет содержать готовые 32-битные слова,
а значение переменной size будет равно 500 */

   /************************************************************
```

```
 Обработка, отображение данных.
 Если требуется – выход из цикла
*********************************************************/
/* Останавливаем потоковое чтение из портов ввода-вывода */
err=LTR43_StopStreamRead((&conf);
 if(err) return(err);

/* Продолжение работы с модулем */
```

5 Формирование меток.

Модуль LTR43 позволяет генерировать специальные сигналы - временнЫе метки, - которые могут выводиться на определенные линии выходного разъема модуля с целью синхронизации работы всей крейтовой системы LTR, а также нескольких систем. Кроме того, модуль высылает в крейт-контроллер специальные пакеты-команды временных меток. Подробнее о метках см. в главе 8.3.3 документа «Крейтовая система LTR. Руководство пользователя».

5.1 Конфигурация секундных меток.

Секундная метка генерируются с периодом 1 с и может быть сконфигурирована как **внутренняя**, **внутренняя с трансляцией на выход** и **внешняя**.

Внутренняя секундная метка НЕ ВЫВОДИТСЯ на линию 2 «СЕКУНДНАЯ МЕТКА» разъема модуля. Но ежесекундно модулем в крейт-контроллер передается слово-команда специального формата, индицирующее, что пришла секундная метка. Каждая такая команда обрабатывается сервером, и счетчик секундных меток сервера инкрементируется В модуль эти команды ***не поступают***.

Внутренняя секундная метка с трансляцией на выход аналогична внутренней (см. предыдущий абзац), но, кроме посылки каждую секунду в крейт контроллер специальных команд, происходит подача импульса на линию 2 «СЕКУНДНАЯ МЕТКА» выходного разъема модуля. Этот сигнал может использоваться для синхронизации работы других модулей крейтовой системы или нескольких крейтовых систем.

Внешняя секундная метка считывается с линии 2 «СЕКУНДНАЯ МЕТКА» разъема модуля. Собственная генерация секундных меток в этом случае модулем не производится. По приходу каждого импульса на указанной линии аппаратура модуля высылает в крейт-контроллер специальную команду, определяющую секундную метку. Счетчик секундных меток при этом инкрементируется аналогичным образом.

Для выполнения конфигурации секундной метки необходимо должным образом заполнить соответствующие поля структуры описания модуля перед вызовом функции *LTR43_Config()*. Тип секундной метки определяется полем **Marks.SecondMark_Mode** структуры описания модуля.

Соответствие значения поля режимам генерации секундной метки:

- LTR43_MARK_MODE_INTERNAL (0) – внутренняя;
- LTR43_MARK_MODE_MASTER (1) — внутренняя с трансляцией на выход;
- LTR43_MARK_MODE_EXTERNAL (2) — внешняя;

Например, для конфигурации секундной метки как «Внутренняя с трансляцией на выход» нужно выполнить следующие действия:

```
TLTR43 conf;

conf.Marks.SecondMark_Mode=1;
LTR43_Config(&conf);
```

Напомним, что функция *LTR43_Config()* используется не только для определения меток, но и для конфигурирования иных функций модуля. Это всегда нужно иметь в виду и следить за состоянием полей структуры описания модуля, которые не затрагивались перед данным конкретным применением функции *LTR43_Config()*, потому что их значения тоже буду загружены в память модуля при вызове этой функции**.**

5.2 Запуск и остановка генерации секундных меток.

Для запуска генерации секундных меток служит функция *LTR43_StartSecondMark()***.** После вызова этой функции начинается генерация внутренних секундных меток. При этом в крейт-контроллер начнут поступать специальные слова-команды, индицирующие приход каждой метки, и счетчик секундных меток сервера будет ежесекундно инкрементироваться. В случае внутренней секундной метки с *трансляцией на выход* начнется также формирование импульсов на линии 2 «СЕКУНДНАЯ МЕТКА» разъема модуля.

Для остановки генерации секундных меток предназначена функция *LTR43_StopSecondMark()*. После вызова этой функции генерация секундных меток и поступление специальных команд в крейт-контроллер прекращается.

5.3 Конфигурация метки «СТАРТ».

В отличие от секундной метки, метка «СТАРТ» генерируется ОДНОКРАТНО. Как и секундная метка, метка «СТАРТ» может быть сконфигурирована как **внутренняя, внутренняя с трансляцией на выход** и **внешняя.**

Внутренняя метка «СТАРТ» НЕ ВЫВОДИТСЯ на линию 1 «МЕТКА СТАРТ» выходного разъема. Но при генерации метки «СТАРТ» модулем в крейт-контроллер передается слово-команда специального формата, индицирующее, что пришла метка «СТАРТ». Каждая такая команда обрабатывается сервером, и счетчик меток «СТАРТ» сервера инкрементируется В модуль эти команды ***не поступают***.

Внутренняя метка «СТАРТ» с трансляцией на выход аналогична внутренней (см. предыдущий абзац), но, кроме посылки в крейт-контроллер специальной команды, при генерации метки происходит формирование импульса на линии 1 «МЕТКА СТАРТ» разъема модуля. Этот сигнал может использоваться для синхронизации работы других модулей крейтовой системы и других крейтов.

Внешняя метка «СТАРТ» считывается с линии 1 «МЕТКА СТАРТ» разъема модуля. Собственная генерация метки «СТАРТ» в этом случае модулем не производится. По приходу внешней метки «СТАРТ» аппаратура модуля высылает в крейт-контроллер специальное слово-команду. Счетчик меток «СТАРТ» при этом инкрементируется.

Для выполнения конфигурации метки «СТАРТ» необходимо должным образом заполнить соответствующие поля структуры описания модуля перед вызовом функции *LTR43_Config()*. Тип метки «СТАРТ» определяется полем **Marks.StartMark_Mode** структуры описания модуля.

Соответствие значения поля режимам генерации секундной метки:

«0» – внутренняя;

«1» – внутренняя с трансляцией на выход;

«2» – внешняя.

Например, для конфигурации метки «СТАРТ» как «Внутренняя с трансляцией на выход», нужно выполнить следующие действия:

```
TLTR43 conf;


conf.Marks.StartMark_Mode=1;
LTR43_Config(&conf);
```

Напомним, что функция *LTR43_Config()* используется не только для определения меток, но и для конфигурирования иных функций модуля. Это всегда нужно иметь в виду и следить за состоянием полей структуры описания модуля, которые не затрагивались перед данным конкретным применением функции *LTR43_Config()*, потому что их значения тоже буду загружены в память модуля при вызове этой функции.

5.4 Запуск однократной генерации метки «СТАРТ».

Для запуска ОДНОКРАТНОЙ генерации метки «СТАРТ» служит функция *LTR43_MakeStartMark()*. При вызове этой функции аппаратура модуля сгенерирует метку. В случае внутренней метки «СТАРТ» с трансляцией на выход произойдет также выдача импульса на линию 1 «МЕТКА СТАРТ» разъема модуля. Если конфигурация предполагает внешнюю метку «СТАРТ», то модуль автоматически отслеживает появление импульса на линии 1 «МЕТКА СТАРТ» разъема модуля. Как и в случае собственной генерации модулем внутренней метки «СТАРТ», при обнаружении прихода внешней метки в крейт-контроллер высылается слово-команда специального формата, которая обрабатывается сервером.

5.5 Чтение значения счетчиков меток.

Значения счетчиков, входящих в состав сервера, можно прочитать двумя способами: при помощи функции *LTR43_Recv()* и при помощи чтения поля **Channel.tmark** структуры описания модуля. Счетчики меток, организованные в сервере, являются 16-битными.

Параметр ***tmark** функции *LTR43_Recv()* после ее вызова указывает на обновленное значение массива с метками. Размер массива **tmark** должен быть равен размеру массива для получения данных (параметр **size** функции *LTR43_Recv()*). Каждый элемент массива **tmark** содержит значения счетчиков как секундных меток, так и меток СТАРТ на момент получения элемента массива **data** с аналогичным индексом. Младшие 16 бит каждого элемента массива **tmark** представляют собой значение счетчика секундных меток, старшие 16 бит – значение счетчика меток «СТАРТ». Более подробно о формате этого массива и работе с ним см. в главах **4.2.2** и **4.3.1.5** документа “Базовая библиотека работы с крейтом LTR”. В последней речь идет о функции **LTR_Recv()**, но массив **tmark** имеет тот же формат.

Поле **Channel.tmark** структуры описания модуля также содержит значения счетчика меток. После вызова любой функции библиотеки, обращающейся к модулю, значение этого поля обновляется. Младшие 16 бит этого числа представляют собой значение счетчика секундных меток, старшие 16 бит – значение счетчика меток «СТАРТ».

При выполнении функции LTR43_Open() происходит обнуление счетчиков как секундных меток, так и меток СТАРТ. То же самое реализовано в модулях LTR41 и LTR42.

6 Обмен данными по интерфейсу RS-485.

6.1 Конфигурирование интерфейса.

Модуль LTR43 позволяет осуществлять обмен информацией по асинхронному интерфейсу RS-485. При этом модуль работает в режиме мастера: после отправки пакета слов он ожидает подтверждения, которое также представляет собой пакет слов. Количество слов (на уровне интерфейса это кадры) как в выходном, так и во входном пакетах, задаются программно.

Для конфигурирования интерфейса предназначена подструктура **RS485** структуры описания модуля. Поля этой подструктуры полностью определяют параметры работы интерфейса: формат кадра, скорость обмена, наличие проверки четности, количество стоп-бит и таймауты. Ниже приводится описание полей подструктуры **RS485**:

```
struct
{
    INT FrameSize;      // Кол-во бит в кадре (5-9)
    INT Baud;          // Скорость обмена в бодах
    INT StopBit;      // Кол-во стоп-бит
    INT Parity;          // Включение бита четности
    INT SendTimeoutMultiplier; // Множитель таймаута отправки
    INT ReceiveTimeoutMultiplier; // Множитель таймаута приема подтверждения
} RS485; // Структура для конфигурации RS485
```

Таблица 3

Поле	Описание
FrameSize	Определяет формат кадра при обмене данных по RS-485. Параметр может принимать значение от 5 до 9, что соответствует количеству значащих бит в кадре от 5 до 9 соответственно
Baud	Скорость обмена в бодах. Значение параметра определяет битовую скорость обмена данными. Минимальное значение – 2400 бит/с. Скорость должна подбираться пользователем в зависимости от конкретных условий работы интерфейса (например, исходя из длины линии)
StopBit	Количество стоп-битов в кадре. Значения параметра: «0» – 1 стоп-бит; «1» – 2 стоп-бита

Parity	Включение и определение режима проверки четности. Значение параметра: «0» - проверка четности отключена; «1» – четная проверка; «2» – нечетная проверка; Если проверка четности отключена, то бит четности не добавляется в конец кадра. Если проверка включена, то после значащих бит каждого исходящего кадра добавляется бит четности, а у принимаемых кадров бит четности анализируется. В режиме четной проверки бит четности равен «0» при четном количестве единиц и «1» при нечетном. В режиме нечетной проверки бит четности равен «0» при нечетном количестве единиц и «1» при четном.
SendTimeoutMultiplier	Множитель таймаута отправки определяет максимальное время, в течение которого модуль будет пытаться отправить все кадры пакета. Если этого сделать не удастся, то по истечении таймаута модуль возвратит ошибку отправки. Таймаут отправки равен произведению множителя на количество кадров в отправляемом пакете, т.е. **to= SendTimeoutMultiplier*N.** Здесь **to** – значение таймаута **в миллисекундах**, **N** – количество слов (кадров) в отправляемом пакете. Значение таймаута, а следовательно, и указанного выше произведения, не должно превышать 255 мс. Это следует иметь ввиду при конфигурировании интерфейса.
ReceiveTimeoutMultiplier	Множитель таймаута приема определяет максимальное время, в течение которого модуль будет пытаться принять все кадры пакета подтверждения. Если этого сделать не удастся, то по истечении таймаута модуль возвратит ошибку приема. Таймаут приема равен произведению множителя на количество кадров в принимаемом пакете подтверждения, т.е. **to= ReceiveTimeoutMultiplier*N.** Здесь **to** – значение таймаута **в миллисекундах**, **N** – количество слов (кадров) в отправляемом пакете. Значение таймаута, а следовательно, и указанного выше произведения, не должно превышать 255 мс. Это следует иметь ввиду при конфигурировании интерфейса.

При вызове функции *LTR43_Config()* все указанные выше установки интерфейса будут загружены в оперативную память микроконтроллера.

Напомним, что функция *LTR43_Config()* используется не только для определения параметров интерфейса RS-485, но и для конфигурирования иных функций модуля. Это всегда нужно иметь в виду и следить за состоянием полей структуры описания модуля, которые не затрагивались перед данным конкретным применением функции

LTR43_Config(), потому что их значения тоже буду загружены в память модуля при вызове этой функции.

Внимание!!! Все кадры отправляемого пакета и принимаемого пакета-подтверждения должны иметь одинаковый формат!

Приведем пример конфигурации.

```
TLTR43  conf;

/* Определение других полей структуры описания модуля */
……..
conf.RS485.FrameSize=8;                    // 8 значащих бит в кадре
conf.RS485.Baud=4800;                      // Скорость обмена – 4800 бод
conf.RS485.StopBit=0;                      // 1 стоп-бит
conf.RS485.Parity=1;                       // Включена «четная» проверка четности
conf.RS485. SendTimeoutMultiplier =5;      // Множитель таймаута отправки – 5 мс
conf.RS485. ReceiveTimeoutMultiplier =5;   // Множитель таймаута приема – 5 мс

/* После заполнения требуемых полей вызываем функцию конфигурации */

LTR43_Config(&conf);
```

6.2 Осуществление обмена данными по интерфейсу RS-485.

Для осуществления обмена данными по интерфейсу RS-485 предназначена функция *LTR43_RS485_Exchange()*. Подробное описание параметров функции приведено ниже в Таблице 4:

Таблица 4

Параметр	Тип	Описание
hnd	PTLTR43	Указатель на структуру описания модуля

***PackToSend**	SHORT	Указатель на массив со словами, входящими в отправляемый пакет. Элемент массива с младшим индексом будет отправлен первым. Максимальное количество значащих бит в отправляемых кадрах – 9. При формировании кадров биты, не соответствующие размеру кадра, заданному ранее при конфигурации, будут проигнорированы. Слова, входящие в массив, будут переданы микроконтроллеру модуля и отправлены по интерфейсу RS-485 в порядке возрастания индекса массива **PackToSend[]**. Отправляемый пакет будет содержать количество кадров, соответствующее параметру **OutPackSize**. Если массив **PackToSend[]** содержит большее количество элементов, то элементы, не соответствующие размеру пакета, будут проигнорированы.
***ReceivedPack**	SHORT	Указатель на массив, в который будут записаны слова, принятые модулем при получении пакета-подтверждения. Элементу массива с младшим индексом соответствует первый полученный кадр пакета-подтверждения. Максимальное количество значащих бит в принимаемых кадрах – 9. При формировании кадров биты, не соответствующие размеру кадра, заданному ранее при конфигурации, будут проигнорированы. Полученный пакет будет содержать количество кадров, соответствующее параметру **InPackSize**. Если в течение определенного ранее таймаута приема пакета подтверждение необходимое количество кадров не было получено, функция завершится с соответствующей ошибкой.
OutPackSize	INT	Количество слов в отправляемом пакете. Оно соответствует количеству сформированных кадров, которые будут последовательно отправлены модулем с использованием интерфейса RS-485. Значение параметра должно быть не менее 1 и не более 10.
InPackSize	INT	Количество слов в принимаемом пакете-подтверждении. Оно соответствует количеству кадров, которые будут последовательно приняты модулем по интерфейсу RS-485 после отправки исходящего пакета. Значение параметра должно быть не менее 1 и не более 10.

При вызове функции **LTR43_RS485_Exchange()** с параметрами, описанными выше, модуль последовательно передаст кадры устройству-приемнику и перейдет в режим

ожидания кадров пакета-подтверждения. Как только пакет-подтверждение будет принят (модуль получит количество кадров, определенное параметром **InPackSize),** полученные слова тут же будут переданы крейт-контроллеру и ПК. Функция **LTR43_RS485_Exchange()** выполнит необходимые проверки, и после окончания ее работы указатель ***ReceivedPack** будет указывать на массив, содержащий полученные слова пакета-подтверждения, причем элементом массива с индексом «0» будет являться первое слово пакета (соответствующее первому полученному кадру пакета-подтверждения).

Пример применения функции *LTR43_RS485_Exchange()* при конфигурации интерфейса, определенной в гл. 6.1 настоящего Руководства:

```
SHORT OutputArray[3 ]={0xFA, 0x31, 0xC5};  // Оправляемый пакет
SHORT ConfirmArray[2 ];     // Массив для записи слов пакета-подтверждения
INT OutSize;                // Переменная для хранения размера исходящего пакета
INT InSize                  // Переменная для хранения размера пакета-подствержднения

 OutSize = 3;   // В исходящем пакете оправляем 3 слова
 InSize = 2;    //  Пакет-подтверждение будет состоять из 2-х слов

/*      Вызываем функцию      */
LTR43_RS485_Exchange(&conf, OutputArray, ConfirmArray, OutSize, InSize);
```

После работы функции в массиве **ConfirmArray[]** будут содержаться два слова, полученные в качестве подтверждения.

6.3 Расширенные возможности для обмена по RS485

В версии 1.6 прошивки контроллера AVR (эту версию можно узнать прочитав поле **ModuleInfo.FirmwareVersion** после установления связи с модулем с помощью *LTR43_Open()*) были расширены возможности для работы с интерфейсом RS485, что позволяет использовать модуль LTR43 для управления другими устройствами на шине RS485.

В версии 1.6 были внесены следующие изменения (с сохранением совместимости со старыми версиями):

1. Максимальный размер как передаваемого, так и принимаемого за один раз пакета по RS485 увеличен до 256 слов.
2. Таймаут на прием пакета теперь складывается из постоянной составляющей и составляющей, которая зависит от количества принимаемых слов (до этого была только вторая составляющая) и рассчитывается (при приеме n слов) как:

$$Tresp = ResponseTout + n*ReceiveTimeoutMultiplier$$

Для задания постоянной составляющей введена функция *LTR43_RS485_SetResponseTout()*. Вторая составляющая как и раньше устанавливается

одноименным полем описателя модуля и записывается в модуль при вызове *LTR43_Config()*.

3. Общее время Tresp не должно превышать 65535 мс. Ограничение на то, что n*ReceiveTimeoutMultiplier не должно превышать 255 мс снято.

4. Поле **SendTimeoutMultiplier** больше не учитывается, так время передачи фактически определяет сам модуль и дополнительные таймауты на это время выглядят нецелесообразными.

5. Модуль теперь может корректно обработать ситуацию, когда в ответ пришло меньше слов по RS485, чем запрашивалось. По истечению таймаута Tresp модуль возвращает те слова, которые успел принять (в версии до 1.6 таймаут отслеживался только до прихода первого слова по RS485, поэтому корректно обрабатывались только варианты когда нет ответа вообще или есть полный ответ). Для варианта, когда прием неполного ответа не является ошибочной ситуацией, введена функция *LTR43_RS485_ExchangeEx()*, в которой введен дополнительный параметр **ReceivedSize**, в котором возвращается реально принятое количество слов. При приеме слов меньше чем **InPackSize**, функция не возвращает ошибку, а верно возвращает принятое количество слов. Таким образом, при использовании этой функции всегда надо проверять значение, которое возвращено в параметре **ReceivedSize**, чтобы знать сколько слов в принятом массиве действительны.

6. Также в случае, если при приеме ответа по RS485 в n-ом слове была ошибка кадра или ошибка четности, а предыдущие n-1 слов были приняты успешно, функция *LTR43_RS485_ExchangeEx()* позволяет обработать эти первые n-1 слов. Она возвращает ошибку, но сохраняет n-1 слов в выходной массив и в **ReceivedSize** возвращает значение n-1.

7. Введен новый режим передачи, в котором конец принимаемого пакета может быть определен по таймауту, устанавливаемому с помощью функции *LTR43_RS485_SetIntervalTout()*. Если в функцию *LTR43_RS485_ExchangeEx()* передать флаг LTR43_RS485_FLAGS_USE_INTERVAL_TOUT, то, если в течение установленного таймаута после предыдущего принятого слова по RS485 не будет принято новое слово, то считается, что пакет на прием завершен, и модуль немедленно возвращает ответ, даже если не были приняты все **InPackSize** слов.

Этот режим позволяет реализовать в частности устройство мастер, работающее по протоколу **Modbus RTU**, в котором данному интервалу соответствует интервал t3.5, который для частот больше 19200 бод/с составляет 1750 мкс, а для 19200 и меньше — время передачи трех с половиной 11-битных слов (старт-бит — 8 бит данных — четность — стоп-бит).

8. Введена функция *LTR43_RS485_SetTxActiveInterval()*, которая позволяет устанавливать интервалы перед передачей (**start_of_packet**) и после передачи (**end_of_packet**) данных по RS485, в течение которых модуль оставляет активным передатчик. В это время на линии RS485 будет активный высокий уровень, меньше подверженный влиянию помех, чем состояние линии, когда передатчики всех устройств выключены. В частности, для протокола **Modbus RTU** это может гарантировать более надежное определение интервала t3.5 между пакетами в условиях помех. Однако следует быть внимательным с интервалом после завершения передачи. Если управляемое устройство (slave) начнет ответ в течение этого интервала, то на линии возникнет конфликт, так как будет активно два передатчика. Следует учитывать, что эти интервалы минимальны и реально могут отличаться в большую сторону на

время, необходимое контроллеру AVR для обработки события окончания интервала. При установке **end_of_packet** интервала следует быть уверенным, что отвечающая устройство не будет передавать данные в течение интервала хотя бы равного **end_of_packet** + 100 мкс, после завершения передачи.

7 Работа с ППЗУ микроконтроллера AVR.

Микроконтроллер AVR ATmega 8515, управляющей работой модуля LTR43, имеет в своем составе Перепрограммируемое Постоянное Запоминающее Устройство (ППЗУ) типа EEPROM объемом 512 байт с байтовым доступом. Пользователь может по своему усмотрению использовать ячейки ППЗУ. Для доступа к ним библиотека включает две функции: *LTR43_WriteEEPROM()* и *LTR43_ReadEEPROM()*.

Для чтения байта из ячейки ППЗУ с указанным адресом используется функция *LTR43_ReadEEPROM()*, где параметр **Address** определяет адрес ячейки ППЗУ, из которой следует произвести чтение, а параметр ***val** представляет собой указатель на байт, считанный из указанной ячейки. Например, чтобы считать из ячейки с адресом 150 значение, хранящееся там, следует выполнить следующий вызов функции:

LTR43_ReadEEPROM(&conf, 150, &ReadVal),
где **conf –** экземпляр структуры описания модуля,
ReadVal – переменная, в которую будет записано считанное значение.

Для записи байта по указанному адресу ППЗУ используется функция *LTR43_WriteEEPROM()*, где параметр **Address** определяет адрес ячейки ППЗУ, в которую следует произвести запись, а параметр **val** определяет байт, который будет записан в указанную ячейку. Например, чтобы записать в ячейку с адресом 150 значение 0x3F, следует выполнить следующий вызов функции:

LTR43_WriteEEPROM(&conf, 150, 0x3F),
где **conf –** экземпляр структуры описания модуля.

Приложение. Примеры конфигурирования и программирования модуля LTR43.

П1.1 Примеры конфигураций.

Перед заданием конфигурации следует инициализировать и открыть интерфейсный канал связи с модулем. Это делается посредством вызова следующих функций: **LTR43_Init()** и **LTR43_Open().** Затем следует заполнить поля структуры описания модуля требуемыми значениями и вызвать функцию **LTR43_Config().** Ниже приводятся примеры задания конфигурации модуля (определение полей структуры описания модуля):

1. Все линии ввода-вывода модуля настроены на выход, интерфейс RS485 задействовать не будем, потому оставляем его конфигурацию по умолчанию. Будет использоваться внутренняя метка «СТАРТ» с трансляцией на выход. Секундные метки не используем, поэтому оставляем их конфигурацию по умолчанию.

```
TLTR43 conf_1;                    // Объявляем  экземпляр структуры описания модуля

conf_1.IO_Ports. Port1=1;         //  Все группы линий ввода-вывода настраиваются на выход
conf_1.IO_Ports. Port2=1;
conf_1.IO_Ports. Port3=1;
conf_1.IO_Ports. Port4=1;

conf_1.Marks.StartMark_Mode=1; //  Внутренняя метка «СТАРТ»  с трансляцией на выход

/*  Остальные поля не изменяем, т.к. не будем пользоваться функциями, предполагающими
определение этих полей       */
```

2. Линии ввода-вывода IO1-IO16 (порты 1 и 2) настроены на вход, линии IO17-IO32 (порты 3 и 4) – на выход. Будем использовать внешнюю секундную метку. Метку «СТАРТ» использовать не будем. В процессе работы будет производиться обмен данными по интерфейсу RS-485 со следующими параметрами: размер кадра – 6 бит, скорость обмена – 9600 бод, проверка четности отключена, 1 стоп-бит. Множители таймаутов - 5 мс каждый.

```
TLTR43 conf_2;   // Объявляем  структуру типа TLTR43


conf_2.IO_Ports.Port1=0;          //  Настраиваем группы линий ввода-вывода
conf_2.IO_Ports.Port2=0;
conf_2.IO_Ports.Port3=1;
conf_2.IO_Ports.Port4=1;
```

conf_2.Marks.SecondMark_Mode=2; // Внешяя секундная метка

/* Определяем параметры RS-485 */

conf_2. RS485.FrameSize=6; // Размер кадра – 6 бит
conf_2. RS485.Baud=9600; // Скорость обмена – 9600 бит
conf_2. RS485.StopBit=0; // 1 стоп-бит
conf_2. RS485.Parity=0; // Проверка четности отключена
conf_2. RS485.SendTimeoutMultiplier=5; // Множители таймаутов – 5 мс
conf_2. RS485.ReceiveTimeoutMultiplier=5;

3. Линии ввода-вывода IO1-IO8 (порт 1) и IO25-IO32 (порт 4) настроены на вход, остальные линии – на выход. Будем использовать внутреннюю секундную метку с трансляцией на выход. Метку «СТАРТ» используется внешняя. В процессе работы будет производиться обмен данными по интерфейсу RS-485 со следующими параметрами: размер кадра – 9 бит, скорость обмена – 19200 бод, проверка четности отключена, 2 стоп-бита. Множитель таймаута передачи - 2 мс, приема – 3 мс.

TLTR43 conf_3; // Объявляем структуру типа TLTR43

conf_3.IO_Ports.Port1=0; // Настраиваем группы линий ввода-вывода
conf_3.IO_Ports. Port2=1;
conf_3.IO_Ports. Port3=1;
conf_3.IO_Ports. Port4=0;

conf_3.Marks.SecondMark_Mode=1; // Внутренняя секундная метка с трансляцией на выход
conf_3.Marks.StartMark_Mode=2; // Внешняя метка «СТАРТ»

/* Определяем параметры RS-485 */

conf_3. RS485.FrameSize=9; // Размер кадра – 9 бит
conf_3. RS485.Baud=19200; // Скорость обмена – 19200 бит
conf_3. RS485.StopBit=0; // 2стоп-бита
conf_3. RS485.Parity=0; // Проверка четности отключена
conf_3. RS485.SendTimeoutMultiplier=2; // Множитель таймаута передачи – 2 мс
conf_3. RS485.ReceiveTimeoutMultiplier=3; // Множитель таймаута приема – 3 мс

П1.2. Пример приложения.

Простое консольное приложение, созданное в среде Microsoft Visual C++ 2005.

```
// example.cpp : Defines the entry point for the console application.
//
```

```
#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
#include <windows.h>
#include "ltr\include\ltr43api.h"

TLTR43 hltr43;
char ErrorString[255];
char MsgString[255];

void oem_printf(char *s)
{
 CharToOem(s,s);
 printf("%s",s);
}


int main(int argc, char* argv[])
{
DWORD AcqThreadId;
DWORD ThreadSatus;

DWORD OutputWord;
DWORD InputWord;

DWORD OutputArray[5]=
 {
 0x0076D300,
 0x009A3400,
 0x00127600,
 0x00FF7100,
 0x00CA6B00
 };

INT TotalBlockCntr;

INT err;                  // Переменная для кода ошибки


//* Инициализируем канал связи с модулем и структуру описания модуля */
 sprintf(MsgString,"Выполняем инициализацию модуля\n");
 oem_printf(MsgString);

 err=LTR43_Init(&hltr43);

 if(err)
 {
 strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
 oem_printf(ErrorString);
 goto End;
  }
 else
 {
  sprintf(MsgString,"LTR43_Init()->OK\n");
```

```
  oem_printf(MsgString);
 }

// Открываем интерф. канал связи с модулем. Сетевой адрес и номер порта - по
умолчанию
// Серийный номер первого найденного модуля;
// Номер посадочного места - 1;

err=LTR43_Open(&hltr43, SADDR_DEFAULT, SPORT_DEFAULT, "", CC_MODULE1);
 if(err)
 {
   if(err==LTR_WARNING_MODULE_IN_USE)
    {
    strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
    oem_printf(ErrorString);
    }
   else
    {
     strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
     oem_printf(ErrorString);
     goto End;
    }
  }
 else
  {
  sprintf(MsgString,"LTR43_Open()->OK\n");
  oem_printf(MsgString);
 }

sprintf(MsgString,"Имя модуля: %s\n", hltr43.ModuleInfo.Name);
oem_printf(MsgString);

sprintf(MsgString,"Версия прошивки AVR: %s\n",
hltr43.ModuleInfo.FirmwareVersion);
oem_printf(MsgString);

sprintf(MsgString,"Дата создания прошивки AVR: %s\n",
hltr43.ModuleInfo.FirmwareDate);
oem_printf(MsgString);

sprintf(MsgString,"Серийный номер модуля: %s\n\n", hltr43.ModuleInfo.Serial);
oem_printf(MsgString);

 // Производим заполнение полей структуры описания модуля требуемыми значениями
 //  Настраиваем порты ввода-вывода

hltr43.IO_Ports.Port1=0;  // на вход
hltr43.IO_Ports.Port2=1;  // на выход
hltr43.IO_Ports.Port3=1;  // на выход
hltr43.IO_Ports.Port4=0;  // на вход


/* Конфигурация меток */
hltr43.Marks.SecondMark_Mode=1;     //  Секундная метка внутр. с трансляцией на
выход
```

```
hltr43.Marks.StartMark_Mode=0;      //  Метка СТАРТ внутренняя


// Вызываем функцию конфигурации модуля
err=LTR43_Config(&hltr43);
if(err)
 {
 strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
 LTR43_Close(&hltr43);
 oem_printf(ErrorString);
 goto End;
  }
 else
  {
  sprintf(MsgString,"LTR43_Config()->OK\n");
  oem_printf(MsgString);
 }


  /*  Выводим слово в порты ввода-вывода. Будут изменены сигналы только на тех
  линиях, котрые настроены на выход, т.е. на линиях 9-24. */

sprintf(MsgString,"Запишем в порты ввода-вывода слово 0x00A17400\n");
oem_printf(MsgString);

OutputWord=0x00B17400;

err=LTR43_WritePort(&hltr43, OutputWord);
if(err)
 {
   strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
   LTR43_Close(&hltr43);
   oem_printf(ErrorString);
   goto End;
  }
 else
   {
  sprintf(MsgString,"LTR43_WritePort()->OK\n");
  oem_printf(MsgString);
 }

// Считаем слово со входных линий ввода-вывода.

sprintf(MsgString,"Считываем слово со входных линий ввода-вывода\n");
oem_printf(MsgString);

err=LTR43_ReadPort(&hltr43, &InputWord);
if(err)
 {
  strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
  LTR43_Close(&hltr43);
  oem_printf(ErrorString);
  goto End;
  }
```

```
 else
 {
  sprintf(MsgString,"LTR43_ReadPort()->OK\n");
  oem_printf(MsgString);
 }
  // Выведем считанное слово на дисплей
  sprintf(MsgString, "Считанное слово: %x hex\n", InputWord);
  oem_printf(MsgString);

/*  ВЫведем массив из 5 элементов в порты ввода-вывода */

err= LTR43_WriteArray(&hltr43,  OutputArray, 5);
if(err)
  {
   strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
   LTR43_Close(&hltr43);
   oem_printf(ErrorString);
   goto End;
  }
else
  {
  sprintf(MsgString,"LTR43_WriteArray->OK\n");
  oem_printf(MsgString);
 }

 // Запускаемп генерацию секундных меток
sprintf(MsgString,"Запускаем генерацию секундных меток\n");
oem_printf(MsgString);

err= LTR43_StartSecondMark(&hltr43);
if(err)
  {
   strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
   LTR43_Close(&hltr43);
   oem_printf(ErrorString);
   goto End;
  }
else
  {
  sprintf(MsgString,"LTR43_StartSecondMarks()->OK\n");
  oem_printf(MsgString);
 }

 // Подождем 3 секунды. После этого в сервере видно, что пришло 3 секундные метки
 Sleep(3000);

sprintf(MsgString,"Останавливаем генерацию секундных меток\n");
oem_printf(MsgString);

// Останавливаем генерацию секундных меток

err= LTR43_StopSecondMark(&hltr43);
if(err)
  {
   strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
```

```
    LTR43_Close(&hltr43);
    oem_printf(ErrorString);
    goto End;
   }
else
   {
   sprintf(MsgString,"LTR43_StopSecondMark()->OK\n");
   oem_printf(MsgString);
   }

 // Сгенерируем метку СТАРТ

   sprintf(MsgString,"Сгенерируем метку СТАРТ\n");
   oem_printf(MsgString);

err= LTR43_MakeStartMark(&hltr43);
if(err)
   {
    strcpy(ErrorString, (char *) LTR43_GetErrorString(err));
    LTR43_Close(&hltr43);
    oem_printf(ErrorString);
    goto End;
   }
else
 {
  sprintf(MsgString,"LTR43_MakeStartMark()->OK\n");
  oem_printf(MsgString);
 }


End:
sprintf(MsgString,">> Для выхода нажмите любую клавишу\n");
oem_printf(MsgString);

    while(!kbhit())
         continue;

         return 0;
}
```

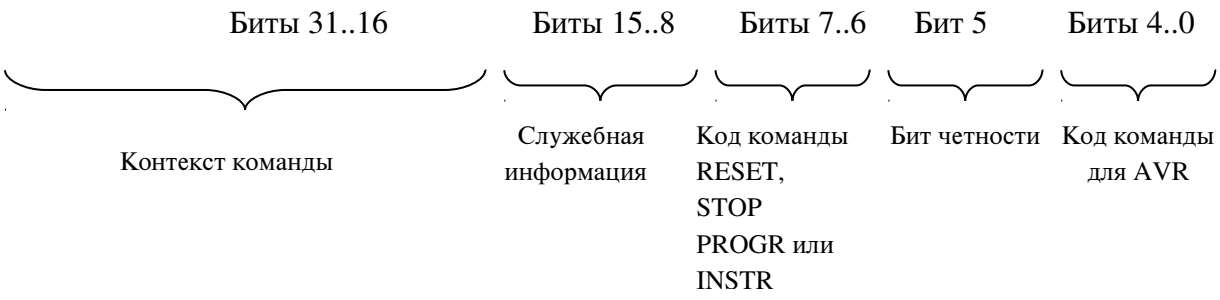
Приложение 2. Протокол обмена данных с модулем.

Замечание : Сведения, приведенные в этом Приложении, являются справочной информацией, и при работе со штатной библиотекой пользовательского интерфейса знание этой информации не требуется.

Протокол обмена данными с модулем основан на использовании формата 4-байтных пакетов команд или данных. Подробно этот формат описан в *книге «Крейтовая система LTR. Руководство пользователя»*. Гл. 4.3. Здесь остановимся только на информации, имеющей значение применительно к модулю LTR43.

Все команды от крейт-контроллера к модулю и подтверждения этих команд представляют собой 4-байтные **командные слова**. Данные, считанные с портов ввода-вывода модуля и передаваемые из модуля в крейт-контроллер, представляют собой 4-байтные **слова данных**. Слова данных также передаются модулю для записи в порты ввода-вывода. Следует отметить, что указанный протокол является общим для всех модулей данной крейтовой системы.

Формат **командного слова** применительно к модулю LTR43 имеет следующий вид:



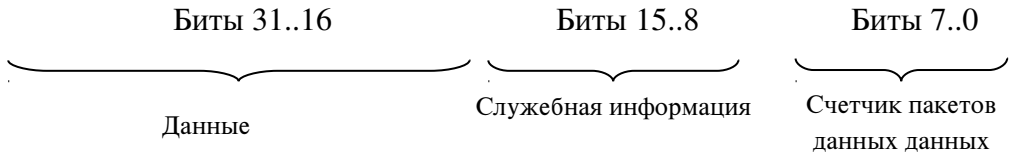
- **Код команды для AVR** – число, определяющее процедуру, которую следует выполнить процессору модуля.
- **Бит четности** – используется для контроля правильности передачи команд в микроконтроллер модуля и в обратном направлении.
- **Код команды RESET, STOP, PROGR или INSTR** – код команды общего интерфейса системы LTR.
- **Служебная информация -** информация о номере слота, бит-признак командного слова. Этот байт несет в себе информацию уровня крейт-контроллера. Кроме бита-признака команды или данных, эти биты не передаются в модуль и от него.
- **Контекст команды** – значения, передающиеся вместе с командой и используемые процессором при ее выполнении. Например, параметры конфигурации портов ввода-вывода, интерфейса RS-485, слова для передачи по RS-485.

  Команды в описанном формате передаются и в обратном направлении: от модуля к крейт-контроллеру. В этом случае они представляют собой или подтверждения выполнения команд, или содержат в контекстных полях значения, которые требовалось получить от модуля. Например, слова пакета-подтверждения при обмене данными по

RS-485. Слова-команды НЕ ИСПОЛЬЗУЮТСЯ для обмена данными с портами ввода-вывода.

Слова данных используются в рассматриваемом модуле только для передачи данных, используемых при работе с портами ввода-вывода. При программировании модуля и обмене по RS-485 слова данных **не используются**.

Формат **слова данных** имеет следующий вид:



- **Данные** - непосредственно слова, записываемые в порты ввода-вывода модуля или считанные их них.
- **Служебная информация -** информация о номере слота, бит-признак слова данных. Этот байт несет в себе информацию уровня крейт-контроллера. Кроме бита-признака команды или данных, эти биты не передаются в модуль и обратно.
- **Счетчик пакетов данных** – 8-битный счетчик пакетов данных, отправляемых модулем в крейт-контроллер. При отправке каждого нового пакета процессор инкрементирует значение этого счетчика, которое впоследствии используется для проверки правильности следования пакетов из модуля.

Ниже приведена таблица с кратким описанием используемых команд. Все команды подтверждаются ответом с тем же кодом команды при успехе, или одним из ответов об ошибке, приведенных ниже, если в описании не сказано иного.

Команда	Код	Данные	Описание
INIT	01111	00000000 00000000	В прошивке версии 1.5 и младших, данная команда должна быть выслана сразу посл сброса модуля с помощью STOP-RESET-STOP и только один раз. С версии 1.6 может использоваться повторно для перевода модуля в начальное состояние без сброса
CONFIG	00010	00ss00mm 0000dddd	Настройка направления линий и режима меток.

			d — направление линий для каждого 8-битного порта (0 — ввод, 1 — вывод) **m** — режим секундной метки (коды аналогичны полю в структуре) **s** — режим стартовой метки
CONFIG_READ_RATE	10000	ssssssss 0000pppp	Настройка скорости ввода данных для потокового ввода. Частота потокового ввода равна 15МГц/(N*(S+1)), где S — делитель из поля **s**, а N — прделитель, который определяется по коду **p**: p = 0, N = 1 p = 1, N = 8 p = 2, N = 64 p = 3, N = 258 p = 4, N = 1024
RS485_CONFIG	00110	0000bbbb bbbbbbbb 00000spp 0000ffff mmmmmmmm rrrrrrrr	Настройка параметров интерфейса RS485. Передаваться должны всегда все 3 слова. Ответ приходит только на третье слово. **b** — делитель для задания скорости интерфейса. Скорость определяется как 15МГц/(16*(b+1)). **f** — размер кадра (код 7 означает кадр из 10 бит, коды 0-3 соответствуют размеру кадра 5-8 бит) **p** — код использования четности (значения аналогичны соответствующему полю структуры) **s** — признак количества стоп-битов (0 — 1 стоп-бит, 1 — 2) **m** — множитель таймаута на передачу **r** — множитель таймаута на прием
READ_CONF_RECORD	01010	0	Запрос информации о модуле. В ответ возвращается информация из 44 слов: 1 байт — признак 0x2B 2 байта версия прошивки (сперва мажорная, затем минораня) 14 байт — строка с датой создания прошивки 8 байт — строка с названием модуля 17 байт — строка с серийным номером

			2 байта — CRC16 по всем предыдущим данным
READ_WORD	00001	0	Чтение состояния цифровых входов модуля.. В ответ возвращается два слова данных: первое с состоянием млаших 16 линий порта, затем с состоянием 16 старших линий.
START_STREAM_READ	01101	0	Запуск потокового ввода с цифровых линий. После этой команды модуль будет выдавать данные входов с настроенной до этого частотой.
STOP_STREAM_READ	01110	0	Останов ранее запущенного потокового ввода
START_SECOND_MARK	00011	0	Запуск секундных метой
STOP_SECOND_MARK	00100	0	Останов секундных меток
MAKE_START_MARK	00101	0	Генерация метки старт
RS485_SEND_BYTE	00111	0000ssss 0000rrrr 0000000d dddddddd 00000000 dddddddd 00000000 pf00000d	Команда используется для обмена по RS485. Первая передача команды содержит количество слов на передачу (**s**) и на прием (**r**) по интерфейсу RS485. После этого должно быть передано s таких же команд с данными на передачу. В ответ при приеме нужного количества слов модуль высылает 2***r** команд с принятыми словами (каждое слово представлено двумя командами), в первой команде передаются младшие 8 бит, а во второй — старший бит, признак ошибки четности (**p**) и ошибки кадра (**f**). В случае истечения таймаута до полного выполнения обмена в ответ высылается всего одна команда *RS485_ERROR_SEND_TIMEOUT* или *RS485_ERROR_CONFIRM_TIMEOUT*
WRITE_EEPROM	01000	0000000a aaaaaaaa	Запись в EEPROM. Передается двумя

		00000000 dddddddd	командами, сперва адрес, затем данные. Ответ высылается один ответ на вторую команду
READ_EEPROM	01001	0000000a aaaaaaaa	Чтение из EEPROM. В команде передается адрес, а в младших 8 битах контекста команды — прочитанное значение.

Вывод данных на цифровые выходы, в отличии остальных операций, осуществляются не с помощью слов команд, а передачей слов данных в модуль. При этом в первом слове передается состояние младших 16 линий (IO1-IO16), во втором слове — старших (IO17-IO32). Третье и четвертое слово являются повторением первого и второго соответственно. В случае не совпадения слов модуль ответить командой *DATA_ERROR*, сигнализирующей об ошибке. В случае успешного вывода, в ответ передается команда подтверждения вывода:

Команда	Код	Данные	Описание
DATA_OUTPUT_CONFIRM	10110	-	Подтверждение вывода на цифровые линии

В случае возникновения ошибок модуль может в вместо стандартного ответа вернуть одну из следующих команд:

Команда	Код	Данные	Описание
PARITY_ERROR	10111	-	Возвращается в случае ошибки бита четности в принятой модулем команде
RS485_ERROR_CONFIRM_TIMEOUT	11000	-	Передается в случае, если за таймаут на прием по RS485 не были приняты все запрашиваемые слова
RS485_ERROR_SEND_TIMEOUT	11001	-	Передается, если за таймаут на передачу не были переданы все слова по RS485 (только для прошивки 1.5 и ниже)
DATA_ERROR	11010	00000000 eeeeeeee	Для прошивки 1.5 и ниже данное слово возвращалось при несовпадении двух повторных слов на вывод данных. В версии 1.6 эта команда также используется для оповещения других ошибок, а данные в команде содержат код ошибки (**e**): 0 — при несовпадении данных на вывод, как и для версии 1.5 1 — принята неподдерживаемая команда 2 — неверные параметры команды 3 — принята недопустимая команда для данного состояния (например нельзя осуществлять обмен по RS485 при запущенном потоковом вводе)

Приложение 3. Новые команды в версии прошивки 1.6.

В версии прошивки 1.6 были добавлены следующие команды:

1. Установка дополнительных параметров

Для установки дополнительных параметров используется команда CMD_CONFIG_PAR_EX:

Код команды	Данные (15-8)	Данные (7-0)
10011	Код параметра	Значение параметра

Старшая часть данных определяет устанавливаемый параметр, а младшая — его значение. Доступны следующие параметры:

Параметр	Код	Значение
PARAM_RS485_RESPONSE_TOUT_L	1	Младший байт значения постоянной составляющей таймаута на ответ по RS485 в мс.
PARAM_RS485_RESPONSE_TOUT_H	2	Старший байт значения постоянной составляющей таймаута на ответ по RS485 в мс. Передается вслед за младшим байтом. Значение таймаута определяется как (256*H + L) мс
PARAM_RS485_INTERVAL_TOUT	3	Таймаут на окончания принимаемого пакета по RS485. Единица соответствует 250 мкс, минимальное значение 7 (1750 мкс), максимальное 255 (63 750 мкс). Если включен специальный режим и с момента приема предыдущего пакета за этот интервал не будет принято следующего, то это будет означать признак конца пакета.
PARAM_RS485_TX_ACTIVE_SOP_INTERVAL	4	Интервал времени перед передачей пакета по RS485, в течение которого модуль активирует передатчик RS485, но не передает данные. Может использоваться для повышения помехозащищенности. Единица соответствует 250 мкс — допустимые значения от 0 до 255 (63 750 мкс)
PARAM_RS485_TX_ACTIVE_EOP_INTERVAL	5	Интервал времени после передачи пакета по RS485, в течение которого модуль оставляет передатчик включенным.

		Единица соответствует 250 мкс. **Внимание**: При использовании, следует быть уверенным, что отвечающее устройство не начнет передавать данные в течение этого интервала с дополнительным запасом в 100 мкс.

В ответ на команду приходит команда с тем же кодом (0x13).

2. Передача расширенного кадра по RS485

Для осуществления обмена по RS485 с расширенным размером буфера на прием и передачу и использования возможности приема неполного пакета используются новые команды:

1. В начале каждого обмена по RS485 передается 3 слова с параметрами текущего обмена, вслед за которыми передается нужно количество слов с данными для передачи по RS485. В ответ на каждое слово модуль высылает подтверждение в виде команды с тем же кодом (0x11). Порядок и формат приведены в таблице:

Команда	Данные	Описание
10001	0001000s ssssssss	s — количество слов на передачу (от 0 до 256)
10001	0010000r rrrrrrrr	r — количество слов на прием (от 0 до 256)
10001	00000000 0000000i	i — признак, использовать ли таймаут на интервал между словами как признак конца принимаемого пакета
10001	0011000d dddddddd	Данные на передачу по RS485. Должно быть s таких пакетов

Внимание: При передаче данных не следует пересылать больше 16 команд, на которых не пришли подтверждения, иначе контроллер AVR может не успеть обработать. Именно для этого введены подтверждения на каждую команду. Это ограничение справедливо для всех команд и данных, но остальные операции, как правило, не требуют пересылки большого количества команд подряд.

2. Далее передается команда CMD_RS485_START_EXCHANGE (0x12), по которой и начинается обмен. На эту команду не пересылается специальный ответ. Все переданные до этого слова модуль вначале передает по RS485, а затем снова переводит интерфейс RS485 в режим приема и ожидает заданное количество слов (r).

Команда	Данные	Описание
10010	0	Запуск обмена данных

3. После того, как AVR примет все слова по RS485 или истечет таймаут (или будет признак окончания кадра при i=1), модуль пошлет в ответ следующую информацию:

Сперва заголовок с количеством принятых слов:

Команда	Данные	Описание
11011	00000000 ssssssss	s — младшие 8 бит, задающих размер ответа

11100	00000000 0000000s	s — старший бит, задающий размер ответа

Вслед за заголовком ответа передается указанное количество принятых слов. Каждое слово передается в виде двух команд следующим образом

Команда	Данные	Описание
11101	00000000 dddddddd	d — младшие 8 бит принятого слова
11110	00000000 pf00000d	d — старший бит принятого слова p — признак, что это слово принято с ошибкой четности f — признак, что это слово было принято с ошибкой кадра (нет стопового бита)

Следует также учитывать, что в случае ошибочных параметров или других ошибок в передаче команд, модуль имеет право вернуть в ответ не указанный заголовок с данными, а одну из команд, сообщающих об ошибке, приведенных в предыдущем приложении (например *DATA_ERROR* или *PARITY_ERROR*).