Библиотека пользовательского интерфейса модуля LTR41

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.1

Март 2007 г.

Автор руководства:
Милованов А.Н.

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 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	08.02.2007	Первая доступная для пользователя ревизия
1.0.1	21.03.2007	Добавлено описание функции LTR41_IsOpened()

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 LTR41.

Модуль LTR41 предназначен для цифрового асинхронного ввода 16-ти TTL/CMOS-сигналов (5V-логика), а также токовых логических сигналов (до 25 мА) с поканальной опторазвязкой. Кроме того, модуль предназначен для синхронизации сбора данных в одном крейте или в многокрейтовой системе по внутренним или внешним синхросигналам. 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 и с портами ввода/вывода модуля. Управляющая программа микроконтроллера записывается в его Flash-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 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, открытие интерфейсного канала связи с модулем;
- 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, управление генерацией меток.
- Закрытие интерфейсного канала связи с модулем.

2 Библиотека интерфейса пользователя модуля LTR41, 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 **TLTR41)** предназначена для инициализации, хранения и изменения данных о конфигурации модуля в рамках создаваемой программы. При помощи полей этой структуры определяются типы синхрометок. Перед вызовом функции конфигурации модуля **LTR41_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 **ltr41api.dll** предназначены для конфигурирования, программирования, сбора данных и диагностики модуля.

2.2.1 Классификация функций пользовательской библиотеки.

Функции, входящие в состав пользовательской библиотеки модуля LTR41, можно разделить на следующие группы:

- Функции инициализации и открытия;
- Функции конфигурирования;

- Функции сбора данных;
- Функция закрытия;
- Вспомогательные функции.

К **функциям инициализации и открытия** относятся функции **LTR41_Init()** и **LTR41_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 **LTR41_Config().** Эта функция передает все конфигурационные параметры, записанные в полях структуры описания модуля, в оперативную память микроконтроллера. После выполнения указанных функций устройство готово к сбору данных.

Функции сбора данных: LTR41_ReadWord(), LTR41_StartStreamRead(), LTR41_StopStreamRead(), LTR41_Recv(), LTR41_ProcessData. Они предназначены для чтения данных с линий ввода модуля, а также для проверки правильности полученных данных.

Функции управления метками:

LTR41_StartSecondMark(),
LTR41_StopSecondMark(),
LTR41_MakeStartMark(),

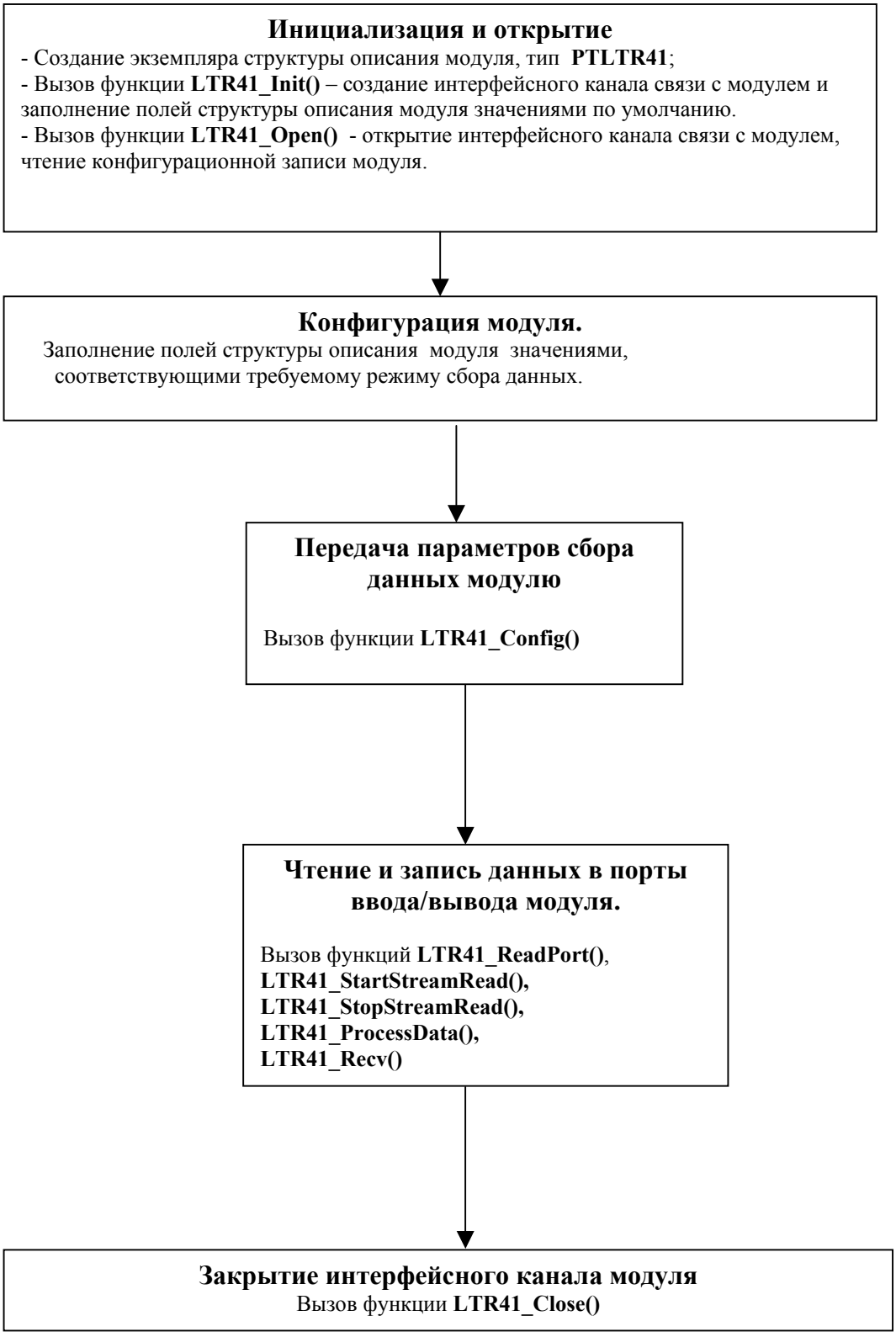
При помощи этих функций производится запуск и остановка генерации секундной метки, а также формирование метки «СТАРТ».

Вспомогательные функции: **LTR41_GetErrorString()** возвращает строку с описанием ошибки, соответствующую коду ошибки; **LTR41_ReadEEPROM()** выполняет чтение информации из указанной ячейки пользовательского ППЗУ, **LTR41_WriteEEPROM()** выполняет запись информации в указанную ячейку пользовательского ППЗУ.

Функция закрытия: LTR41_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 **ltr41api.dll.**

3 Подробное описание библиотеки ltr41api.

Для получения возможности вызова интерфейсных функций библиотеки ***ltr41api.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 **ltr41api.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\ltr41api.lib** и **LTR\INCLUDE\ltr41api.h**;

  Microsoft Visual C++ :
  - подключить к проекту файлы **LTR\LIB\MSVC\ltr41api.lib** и **LTR\INCLUDE\ltr41api.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 **LTR41_Open().** Определение структуры приводится ниже:

```
typedef struct
  {
   TLTR Channel;
   int size;                          // Размер структуры
   double StreamReadFreq;             // Частота выдачи данных при потоковом вводе

   struct
    {
    int SecondMark_Mode;    // Тип секундной метки. 0 – внутр., 1-внутр.+выход, 2-внешняя
    int StartMark_Mode;     // Тип метки «СТАРТ». 0 – внутр., 1-внутр.+выход, 2-внешняя
     } Marks;                          // Подструктура конфигурирования меток

   TINFO_LTR41 ModuleInfo;   // Конфигурационная запись

  } TLTR41, *PTLTR41;                 // Структура описания модуля
```

Далее приводится описание полей этой структуры.

Таблица 1

Поле			**Тип**	**Описание**
size			**INT**	объем памяти, выделяемой под структуру
Channel			**TLTR**	Подструктура, представляющая собой описание интерфейсного канала связи с модулем.
StreamReadRate			**double**	Частота выдачи данных при потоковом вводе
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_LTR41.** Данный тип представляет собой структуру. Ее описание приводится ниже в этой главе настоящего Руководства

Идентификационная информация модуля:

```
typedef struct
```

```
{
  CHAR Name[16];          // Имя модуля
  CHAR Serial[24];        // Серийный номер модуля
  CHAR FirmwareVersion[8];  // Версия БИОСа
  CHAR FirmwareDate[16];    // Дата создания БИОСа

} TINFO_LTR41,*PTINFO_LTR41;
```

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

3.2 Описание функций библиотеки ltr41.dll.

Формат: INT LTR41_Init(TLTR41 *hnd)

Параметр: hnd – указатель на структуру описания модуля (тип **PTLTR41)**

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(TLTR41);  // Размер структуры описания модуля

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

// 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 записи будут содержать только «\0»
```

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

Формат: INT LTR41_Open(TLTR41 *hnd, DWORD net_addr, WORD net_port, CHAR *crate_sn, INT slot_num)

Параметры:

- **hnd** – указатель на структуру описания модуля (тип **PTLTR41)**
- **net_addr** – сетевой адрес сервера A.B.C.D в формате HEX: 0xABCD. 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.
- **net_port** – сетевой порт сервера
- **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 LTR41_IsOpened(PTLTR41 hnd)

Параметр: **hnd** – указатель на структуру описания модуля, тип TLTR41

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 LTR41_Close(TLTR41 *hnd)

Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR41**

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

Формат: INT LTR41_Config(TLTR41 *hnd)

Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR41**

Описание: функция передачи модулю конфигурационной информации, определенной ранее в полях структуры описания модуля. Информация о частоте выдачи данных при потоковом сборе и о конфигурации меток загружается в оперативную память микроконтроллера модуля. Перед использованием этой функции поля структуры описания модуля должны быть заполнены требуемыми значениями. Чтение данных с входных линий, а также генерацию временнЫх меток следует осуществлять *только после выполнения этой функции*.

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

Формат: INT LTR41_ReadPort(TLTR41 *hnd, DWORD *InputData)
Параметры: hnd – указатель на структуру описания модуля **PTLTR41;** *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 **глТаве 4.2**. настоящего Руководства.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_StartStreamRead(TLTR41 *hnd)
Параметр: hnd – указатель на структуру описания модуля **PTLTR41;**
Описание: Запускает потоковое (непрерывное) чтение значений сигналов с входных линий. Модуль начнет выдавать пакеты данных с частотой, определяемой полем **StreamReadRate** структуры описания модуля. Подробнее о непрерывном чтении данных см. в *главе 4.3* настоящего Руководства.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_Recv(TLTR41 *hnd, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)
Параметр: hnd – указатель на структуру описания модуля **PTLTR41;** ***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 содержатся в массиве, адресуемом указателем ***data**. Указатель ***tmark** адресует массив, содержащий счетчики меток (секундных 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 завершается. Положительное 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 **LTR41_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 LTR41_ProcessData(TLTR41 *hnd, DWORD *src, WORD *dest, DWORD *size)
Параметры: hnd – указатель на структуру описания модуля **PTLTR41;** ***src –** указатель на массив с исходными словами данных, полученными из модуля. Элементы массива – «сырые» данные, полученные непосредственно из крейт-контроллера; ***dest –** указатель на массив сформированных 16-битных слов данных; ***size** – указатель на количество слов данных в массивах как входном, так и выходном
Описание: Выполняет проверку целостности данных, поступающих при потоковом чтении из портов ввода/вывода, а также формирует готовые 16-битные слова, соответствующие считанным значениям цифровых линий. О формате слов подробнее см. в описании однократного чтения из порта ввода/вывода, в *главе 4.2.* настоящего Руководства. При потоковом чтении пользователю необходимо самостоятельно получать слова-данные из модуля при помощи функции **LTR41_Recv(),** а затем передавать полученные данные функции **LTR41_ProcessData(),** после работы которой массив, адресуемый указателем ***dest**, будет содержать сформированные 16-битные слова. Параметр ***size** при входе в функцию указывает на размер исходного массива, при выходе из функции – выходного. Если в процессе работы функции ошибок обнаружено не было, то значение, адресуемое ***size** после работы функции не изменится. Если функция выявляет сбой в последовательности значений счетчика принимаемых слов-данных, то ее выполнение немедленно прерывается и возвращается код ошибки. В этом случае параметр ***size** будет адресовать значение, равное количеству верных элементов, записанных в выходной массив до обнаружения ошибки. Подробнее о непрерывном чтении данных и, в частности, о применении функции **LTR41_ProcessData()** см. *главе 4.3* настоящего Руководства.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_StopStreamRead(TLTR41 *hnd)
Параметры: hnd – указатель на структуру описания модуля **PTLTR41;**
Описание: Останавливает потоковое (непрерывное) чтение значений сигналов с входных линий модуля.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_StartSecondMark(TLTR41 *hnd)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR41**
Описание: Выполняет запуск генерации секундной метки.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_StopSecondMark(TLTR41 *hnd)
Параметры: - **hnd –** указатель на структуру описания модуля типа **descr_ltr43**
Описание: Выполняет остановку генерации секундной метки.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_MakeStartMark(TLTR41 *hnd)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR41**
Описание: Выполняет однократную генерацию метки «СТАРТ».
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR41_ReadEEPROM(TLTR41 *hnd, INT Address, BYTE *val)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR41;** - **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 LTR41_WriteEEPROM(TLTR41 *hnd, INT Address, BYTE val)
Параметры: - **hnd –** указатель на структуру описания модуля типа **PTLTR41;** - **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 LTR41_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

4 Организация обмена данными с портами ввода-вывода.

4.1 Конфигурация линий ввода-вывода

Модуль LTR41 имеет 16 цифровых линий, которые ВСЕГДА настроены на вход.

4.2 Чтение 16-битного слова из порта ввода модуля.

Для чтения сигналов с входных линий предназначена функция **LTR41_ReadPort()**. После выполнения функции параметр-указатель ***InputData** будет адресовать 16-битное значение, считанное из порта ввода. Значение, адресуемое ***InputData,** представляет собой 16-битное слово, каждый бит которого определяет значение сигнала для соответствующей входной линии. Биты 15..0 этого числа соответствуют линиям ввода-вывода IN16..IN1. Значение бита “1” соответствует уровню логической единицы на соответствующей линии, значение “0” – уровню логического нуля.

Пример вызова функции:

DWORD WordFromPort; // Переменная для хранения считанного значения

LTR41_ReadPort(&conf, &WordFromPort);

Если, например, считано значение **WordFromPort**=0x7A4C hex, то это соответствует следующим сигналам на входных линиях:

Таблица 2

Линия	**Уровень сигнала**	**Линия**	**Уровень сигнала**	**Линия**	**Уровень сигнала**	**Линия**	**Уровень сигнала**
IN1	0	**IN 5**	0	**IN 9**	0	**IN 13**	1
IN 2	0	**IN 6**	0	**IN 10**	1	**IN 14**	1
IN 3	1	**IN 7**	1	**IN 11**	0	**IN 15**	1
IN 4	1	**IN 8**	0	**IN 12**	1	**IN 16**	0

4.3 Потоковое (непрерывное) чтение данных из порта ввода 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 чтения необходимо выполнить функцию **LTR41_Config()**. Следует иметь ввиду, что частота выдачи данных имеет дискретный диапазон значений, поэтому функция **LTR41_Config()** сама определит ближайшее возможное значение к требуемому пользователем и подкорректирует поле **StreamReadRate** структуры описания модуля. Таким образом, после выполнения функции **LTR41_Config()** поле **StreamReadRate** структуры описания модуля будет содержать *фактическую* частоту выдачи данных.

Потоковый ввод запускается функцией **LTR41_StartStreamRead().**

В случае потокового ввода считывать массивы слов данных из модуля необходимо при помощи функции **LTR41_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 порта ввода, необходимо применить функцию **LTR41_StartStreamRead().** Затем при помощи функции **LTR41_Recv()** следует циклически считывать массивы данных, поступающие из модуля.

Получив порцию данных, необходимо вызвать функцию **LTR41_ProcessData()** для проверки целостности потока данных и для формирования конечных 16-битных значений. Указатель на считанный массив данных должен передаваться функции **LTR41_ProcessData()** (параметр ***src**). При вызове этой функции параметр ***size** должен указывать на размер **исходного,** массива.

После выполнения функции **LTR41_ProcessData()** будет сформирован выходной массив, содержащий готовые 16-битные данные. Этот массив адресуется параметром ***dest**.

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. Если в процессе работы функции ошибок обнаружено не было, то значение, адресуемое ***size** после работы функции не изменится. Если были найдены ошибки и выполнение функции прервалось, то ***size** будет адресовать значение, равное количеству верных элементов, записанных в выходной массив до обнаружения ошибки.

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[1000];

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

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

   /*  Вызываем функцию конфигурации модуля   */
   err=LTR41_Config(&conf);
    if(err) return (err);

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

   err=LTR41_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 */
   LTR41_Recv(&hltr41, SourceArray, NULL, size, 2000);

   /*Далее вызываем функцию LTR41_ProcessData() */

   err=LTR41_ProcessData(&conf, SourceArray, DestArray, &size);
    if(err)
      {
       LTR41_StopStreamRead((&conf);
       return(err);
    }
```

```
/* После работы функции массив DestArray[] будет содержать готовые 16-битные слова */
    /*************************************************************
     Обработка, отображение данных.
     Если требуется – выход из цикла
     *************************************************************/

    /* Останавливаем потоковое чтение из портов ввода-вывода */

    err=LTR41_StopStreamRead((&conf);
     if(err) return(err);

    /* Продолжение работы с модулем */
    ……
```

5 Формирование меток.

Модуль LTR41 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 **LTR41_Config().** Тип секундной метки определяется полем **Marks.SecondMark_Mode** структуры описания модуля.

Соответствие значения поля режимам генерации секундной метки:

«0» – внутренняя;

«1» – внутренняя с трансляцией на выход;

«2» – внешняя.

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

```
TLTR41  conf;

conf.Marks.SecondMark_Mode=1;
LTR41_Config(&conf);
```

Напомним, что функция **LTR41_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 **LTR41_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 **LTR41_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 **LTR41_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 **LTR41_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:

```
TLTR41  conf;

conf.Marks.StartMark_Mode=1;
LTR41_Config(&conf);
```

Напомним, что функция **LTR41_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 **LTR41_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 **LTR41_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 **LTR41_Recv()** и при помощи чтения поля **Channel.tmark** структуры описания модуля. Счетчики меток, организованные в сервере, являются 16-битными.

Параметр ***tmark** функции **LTR41_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 **LTR41_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 LTR41_Open() происходит обнуление счетчиков как секундных меток, так и меток СТАРТ. То же самое реализовано в модулях LTR42 и LTR43.

6 Работа с ППЗУ микроконтроллера AVR.

Микроконтроллер AVR ATmega 8515, управляющей работой модуля LTR41, 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: **LTR41_WriteEEPROM()** и **LTR41_ReadEEPROM().**

Для чтения байта из ячейки ППЗУ с указанным адресом используется функция **LTR41_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:

LTR41_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 **LTR41_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:

LTR41_WriteEEPROM(&conf, 150, 0x3F),
где **conf –** экземпляр структуры описания модуля.

Приложение. Примеры конфигурирования и программирования модуля LTR41.

П1.1 Примеры конфигураций.

Перед заданием конфигурации следует инициализировать и открыть интерфейсный канал связи с модулем. Это делается посредством вызова следующих функций: **LTR41_Init()** и **LTR41_Open().** Затем следует заполнить поля структуры описания модуля требуемыми значениями и вызвать функцию **LTR41_Config().** Ниже приводятся примеры задания конфигурации модуля (определение полей структуры описания модуля):

1. Будем использовать внутреннюю секундную метку с трансляцией на выход. Метку «СТАРТ» используется внешняя. При потоковом вводе данные должны выдаваться модулем с частотой 10 кГц

```
TLTR41 conf;   // Объявляем  структуру типа TLTR41

conf.StreamReadRate=10000;  // Определяем частоту выдачи данных при потоковом сборе

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

П1.2. Пример приложения.

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

```
#include "stdafx.h"
#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
#include <windows.h>
#include "ltr\include\ltr41api.h"
```

```
TLTR41 hltr41;
char ErrorString[255];
char MsgString[255];

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


int main(int argc, char* argv[])
{

WORD InputWord;
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=LTR41_Init(&hltr41);

 if(err)
 {
 strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
 oem_printf(ErrorString);
 goto End;
  }
 else
 {
  sprintf(MsgString,"LTR41_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=LTR41_Open(&hltr41, SADDR_DEFAULT, SPORT_DEFAULT, "", CC_MODULE1);
 if(err)
 {
   if(err==LTR_WARNING_MODULE_IN_USE)
    {
    strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
    oem_printf(ErrorString);
    }
   else
    {
     strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
     oem_printf(ErrorString);
     goto End;
    }
  }
 else
  {
  sprintf(MsgString,"LTR41_Open()->OK\n");
  oem_printf(MsgString);
 }

sprintf(MsgString,"Имя модуля: %s\n", hltr41.ModuleInfo.Name);
```

```
oem_printf(MsgString);

sprintf(MsgString,"Версия прошивки AVR: %s\n",
hltr41.ModuleInfo.FirmwareVersion);
oem_printf(MsgString);

sprintf(MsgString,"Дата создания прошивки AVR: %s\n",
hltr41.ModuleInfo.FirmwareDate);
oem_printf(MsgString);

sprintf(MsgString,"Серийный номер модуля: %s\n\n", hltr41.ModuleInfo.Serial);
oem_printf(MsgString);

 // Производим заполнение полей структуры описания модуля требуемыми значениями

/* Конфигурация меток */
hltr41.Marks.SecondMark_Mode=1;     //  Секундная метка внутр. с трансляцией на
выход
hltr41.Marks.StartMark_Mode=0;      //  Метка СТАРТ внутренняя



// Вызываем функцию конфигурации модуля
err=LTR41_Config(&hltr41);
if(err)
 {
 strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
 LTR41_Close(&hltr41);
 oem_printf(ErrorString);
 goto End;
  }
 else
  {
  sprintf(MsgString,"LTR41_Config()->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=LTR41_ReadPort(&hltr41, &InputWord);
if(err)
 {
  strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
  LTR41_Close(&hltr41);
  oem_printf(ErrorString);
  goto End;
  }
 else
 {
  sprintf(MsgString,"LTR41_ReadPort()->OK\n");
  oem_printf(MsgString);
 }
  // Выведем считанное слово на дисплей
  sprintf(MsgString, "Считанное слово: %x hex\n", InputWord);
  oem_printf(MsgString);

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

```
err= LTR41_StartSecondMark(&hltr41);
if(err)
  {
   strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
   LTR41_Close(&hltr41);
   oem_printf(ErrorString);
   goto End;
  }
else
  {
  sprintf(MsgString,"LTR41_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= LTR41_StopSecondMark(&hltr41);
if(err)
  {
   strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
   LTR41_Close(&hltr41);
   oem_printf(ErrorString);
   goto End;
  }
else
  {
  sprintf(MsgString,"LTR41_StopSecondMark()->OK\n");
  oem_printf(MsgString);
  }

 // Сгенерируем метку СТАРТ

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

err= LTR41_MakeStartMark(&hltr41);
if(err)
  {
   strcpy(ErrorString, (char *) LTR41_GetErrorString(err));
   LTR41_Close(&hltr41);
   oem_printf(ErrorString);
   goto End;
  }
else
 {
  sprintf(MsgString,"LTR41_MakeStartMark()->OK\n");
  oem_printf(MsgString);
 }


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

   while(!kbhit())
        continue;
```

```
        return 0;
}
```

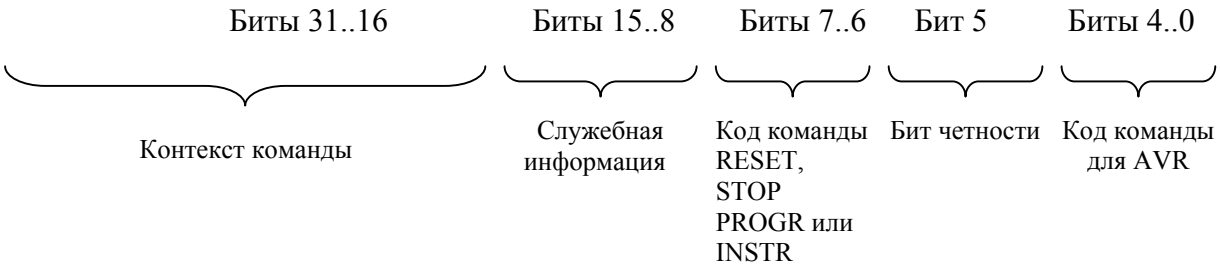
Приложение 2. Протокол обмена данными с модулем.

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 LTR41.

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 крейтовой системы LTR.

Формат **командного слова** применительно к модулю LTR41 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 посадочного места, 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, частота выдачи данных при потоковом сборе.

  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. 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.

Формат **слова данных** имеет следующий вид:

- **Данные** - непосредственно слова, считанные из порта ввода.
- **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.