Библиотека пользовательского интерфейса модуля LTR42

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.1

Март 2007г.

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

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 LTR42.

Модуль LTR42 предназначен для цифрового асинхронного вывода 16-ти управляющих сигналов внешними исполнительными устройствами через оптореле́йные выходы с поканальной гальваноразвязкой. Кроме того, модуль предназначен для синхронизации сбора данных в одном крейте или в многокрейтовой системе по внутренним или внешним синхросигналам. Модуль предназначен для управления МЕДЛЕННЫМИ устройствами и не может использоваться, например, для организации последовательного интерфейса! Период вывода слов в порт не может превышать 1 мс! Подробно об аппаратной 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 и контролирующий обмен информацией с крейт-контроллером. Управляющая программа микроконтроллера записывается в его Flasf-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 и загрузка их в память микроконтроллера AVR ATmega 8515;
- Вывод данных на выходные линии. Управление генерацией меток.
- Закрытие интерфейсного канала связи с модулем.

2 Библиотека пользовательского интерфейса модуля LTR42, 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 **TLTR42)** предназначена для инициализации, хранения и изменения данных о конфигурации модуля в рамках создаваемой программы. Поскольку с точки зрения программного обеспечения и функциональности модуль довольно прост, в нем нет значительного количества настраиваемых параметров, как в большинстве других модулей системы LTR. В данном модуле настраиваются только режимы генерации меток, однако структура описания модуля все равно введена в библиотеку пользовательского интерфейса. Это сделано для того чтобы обеспечить единообразие конфигурирования и программирования всех модулей системы LTR.

При помощи полей этой структуры задается только режим генерации меток. Перед вызовом функции конфигурации модуля **LTR42_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 **ltr42api.dll** предназначены для конфигурирования, программирования, сбора данных и диагностики модуля.

2.2.1 Классификация функций пользовательской библиотеки.

Функции, входящие в состав пользовательской библиотеки модуля LTR42, 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 **LTR42_Init()** и **LTR42_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 **LTR42_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.

Функция вывода данных: LTR42_WriteWord(), Она предназначена для вывода данных на выходные линии модуля.

Функции управления метками:

LTR42_StartSecondMark(),
LTR42_StopSecondMark(),
LTR42_MakeStartMark(),

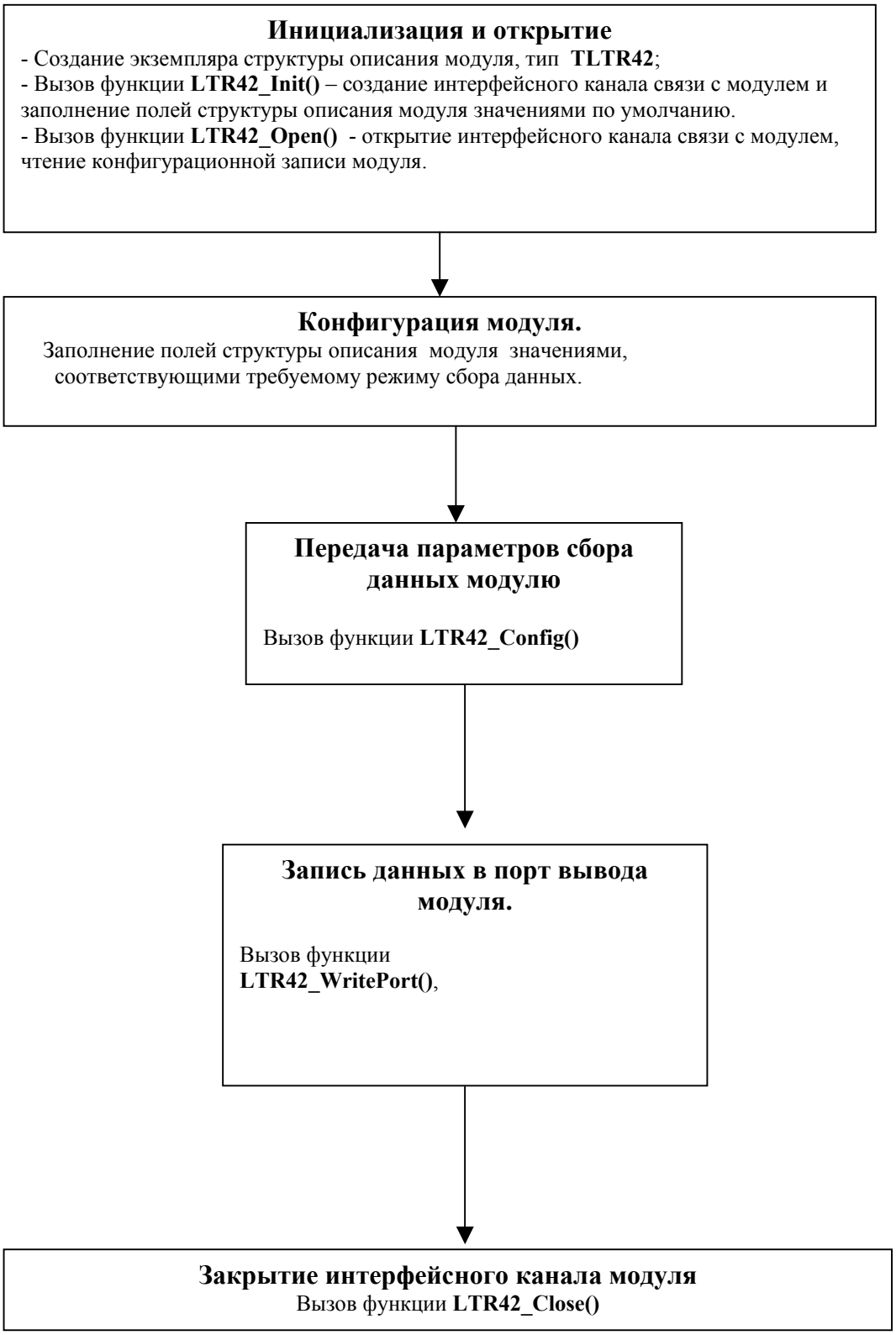
При помощи этих функций производится запуск и остановка генерации секундной метки, а также формирование метки «СТАРТ».

Вспомогательные функции: LTR42_GetErrorString() возвращает строку с описанием ошибки, соответствующую коду ошибки; **LTR42_ReadEEPROM ()** выполняет чтение информации из указанной ячейки пользовательского ППЗУ, **LTR42_WriteEEPROM()** выполняет запись информации в указанную ячейку пользовательского ППЗУ.

Функция закрытия: LTR42_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 **ltr42api.dll.**

3 Подробное описание библиотеки ltr42api.

Для получения возможности вызова интерфейсных функций библиотеки ***ltr42api.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 **ltr42api.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\ltr42api.lib** и **LTR\INCLUDE\ltr42api.h**;

  Microsoft Visual C++ :
  - подключить к проекту файлы **LTR\LIB\MSVC\ltr42api.lib** и **LTR\INCLUDE\ltr42api.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.

Для рассматриваемого модуля поля данной структуры содержат информацию о интерфейсном канале модуля и о конфигурации меток. Также в эту структуру считывается конфигурационная запись модуля при вызове функции **LTR42_Open().** Определение структуры приводится ниже:

```
typedef struct
  {
   TLTR Channel;
   int size;                          // Размер структуры
   BOOLEAN AckEna;                    // Флаг включения подтверждений
   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_LTR42 ModuleInfo;

 } TLTR42, *PTLTR42;                   // 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.
AckEna		**BOOLEAN**	Флаг подключения подтверждений. «1» - подтверждения кадого выведенного в порт слова включены. «0» - отключены.
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_LTR42.** Данный тип представляет собой структуру. Ее описание приводится ниже в этой главе настоящего Руководства

Идентификационная информация модуля:

```
typedef struct
{
  CHAR Name[16];
  CHAR Serial[24];
  CHAR FirmwareVersion[8];     // Версия прошивки микроконтроллера
  CHAR FirmwareDate[16];       // Дата создания данной версии прошивки микроконтроллера

} TINFO_LTR42,*PTINFO_LTR42;
```

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 ltr42.dll.

Формат: INT LTR42_Init(TLTR42 *hnd)

Параметр: *hnd – указатель на структуру описания модуля (тип **TLTR42)**

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(TLTR42);
AckEna=1; // По умолчанию включен режим с подтверждениями

// 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 LTR42_Open(TLTR42 *hnd, DWORD net_addr, WORD net_port, CHAR *crate_sn, INT slot_num)

Параметры:

- ***hnd** – указатель на структуру описания модуля (тип **TLTR42)**
- **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 посадочного места, 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 LTR42_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 LTR42_Close(TLTR42 *hnd)
Параметр: *hnd – указатель на структуру описания модуля, тип **TLTR42**
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 LTR42_Config(TLTR42 *hnd)
Параметр: *hnd – указатель на структуру описания модуля, тип **TLTR42**
Описание: функция передачи модулю конфигурационной информации, определенной ранее в полях структуры описания модуля. Перед использованием этой функции поля структуры описания модуля должны быть заполнены требуемыми значениями. Вывод данных, 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 LTR42_WritePort(TLTR42 *hnd, WORD OutputData)
Параметры: *hnd – указатель на структуру описания модуля **TLTR42;** OutputData - 16-битное слово, определяющее уровень сигнала на каждой из выходных линий
Описание: Выполняет однократную запись 16-битного слова данных в порт вывода. О формате параметра **OutputData** см. подробно в *главе 4.1.* настоящего Руководства. Функция сама отслеживает значение флага AckEna и в зависимости от этого ожидает или нет подтверждения от модуля вывода слова в порт.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR42_StartSecondMark(TLTR42 *hnd)
Параметры: - ***hnd –** указатель на структуру описания модуля типа **TLTR42**
Описание: Выполняет запуск генерации секундной метки.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR42_StopSecondMark(TLTR42 *hnd)
Параметры: - ***hnd –** указатель на структуру описания модуля типа **TLTR42**
Описание: Выполняет остановку генерации секундной метки.
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR42_MakeStartMark(TLTR42 *hnd)
Параметры:

- ***hnd –** указатель на структуру описания модуля типа **TLTR42**
Описание: Выполняет однократную генерацию метки «СТАРТ».
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR42_ReadEEPROM(TLTR42 *hnd, INT Address, BYTE *val)
Параметры: - ***hnd –** указатель на структуру описания модуля типа **TLTR42;** - **Addres –** адрес ячейки ППЗУ, из которой следует считать байт; - ***val –** указатель на считанное 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 LTR42_WriteEEPROM(TLTR42 *hnd, INT Address, BYTE val)
Параметры: - ***hnd –** указатель на структуру описания модуля типа **TLTR42;** - **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 LTR42_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 вывода данных.

4.1 Запись 16-битного слова в порт модуля.

Для записи слова данных в порт модуля LTR42 предназначена функция **LTR42_WritePort()**. Параметр **OutputData** представляет собой 16-битное слово, каждый бит которого определяет значение сигнала для соответствующей выходной линии. Биты 15..0 параметра **OutputData** соответствуют линиям вывода OUT16..OUT1. Значение бита “1” соответствует уровню логической единицы на соответствующей линии, значение “0” – уровню логического нуля.

Например, после вызова функции **LTR42_WritePort(&conf, 0xAF03),** где **conf** – экземпляр структуры описания модуля, на выходных линиях установятся следующие уровни сигнала:

Таблица 2

Линия	**Уровень сигнала**	**Линия**	**Уровень сигнала**
OUT1	1	**OUT 9**	1
OUT 2	1	**OUT 10**	1
OUT 3	0	**OUT 11**	1
OUT 4	0	**OUT 12**	1

OUT 5	0	**OUT 13**	0
OUT 6	0	**OUT 14**	1
OUT 7	0	**OUT 15**	0
OUT 8	0	**OUT 16**	1

Следует иметь ввиду, что данный модуль предназначен для медленного вывода и управления медленными устройствами! Период вывода слов в порт НЕ МОЖЕТ быть менее 1 мс. Управляющая программа микроконтроллера при помощи программируемого таймера блокирует вывод слова в порт, если с момента вывода предыдущего слова не прошло 1 мс. Если включен режим с подтверждением (**AckEna**=1), то после вывода слова модуль отправляет подтверждение в крейт-контроллер и ПК. Функция **LTR42_WritePort()** ожидает подтверждения и только после этого завершается. Таким образом, успешное завершение упомянутой функции свидетельствует о том, что слово ГАРАНТИРОВАННО было выведено в порт.

Если пользователю необходимо для ускорения процесса вывода отказаться от подтверждений, на получение которых тратится от нескольких миллисекунд до нескольких десятков миллисекунд, то он может отключить подтверждения программно. Для этого следует флаг **AckEna** положить равным 0. Пользователь может вызывать функции **LTR42_WritePort()** последовательно или через какой-то временной промежуток, определяемый им самим. Слова будут отправляться в порт быстрее (не будет тратиться время на получение подтверждений). Однако гарантий того, что слово было выведено в порт и не было заблокировано таймером 1 мс в этом случае нет. Вся ответственность за целостность доставки данных на выход модуля ложится при этом на пользователя. Следует отметить, что в таком режиме сложно контролировать скорость вывода в порт, например, из-за загрузки операционной системы и буферизации данных перед отправкой по USB. Поэтому *настоятельно не рекомендуется* использовать режим с отключенными подтверждениями вывода.

5 Формирование меток.

Модуль LTR42 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 **LTR42_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:

```
TLTR42 conf;

conf.Marks.SecondMark_Mode=1;
LTR42_Config(&conf);
```

5.2 Запуск и остановка генерации секундных меток.

Для запуска генерации секундных меток служит функция **LTR42_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 **LTR42_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 **LTR42_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:

```
TLTR42 conf;


conf.Marks.StartMark_Mode=1;
LTR42_Config(&conf);
```

5.4 Запуск однократной генерации метки «СТАРТ».

Для запуска ОДНОКРАТНОЙ генерации метки «СТАРТ» служит функция **LTR42_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 **LTR_Recv()**, входящей в библиотеку крейт-контроллера ltrapi.dll, и при помощи чтения поля **Channel.tmark** структуры описания модуля. Счетчики меток, организованные в сервере, являются 16-битными.

Параметр ***tmark** функции **LTR_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 **LTR_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? а также о функции **LTR_Recv()** см. в главах **4.2.2** и **4.3.1.5** документа “Базовая библиотека работы с крейтом LTR”.

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 LTR42_Open() происходит обнуление счетчиков как секундных меток, так и меток СТАРТ. То же самое реализовано в модулях LTR41 и LTR43.

6 Работа с ППЗУ микроконтроллера AVR.

Микроконтроллер AVR ATmega 8515, управляющей работой модуля LTR42, 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: **LTR42_WriteEEPROM()** и **LTR42_ReadEEPROM().**

Для чтения байта из ячейки ППЗУ с указанным адресом используется функция **LTR42_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:

LTR42_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 **LTR42_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:

LTR42_WriteEEPROM(&conf, 150, 0x3F),
где **conf –** экземпляр структуры описания модуля.

Приложение. Примеры конфигурирования и программирования модуля LTR42.

П1.1 Примеры конфигураций.

Перед заданием конфигурации следует инициализировать и открыть интерфейсный канал связи с модулем. Это делается посредством вызова следующих функций: **LTR42_Init()** и **LTR42_Open().** Затем следует заполнить поля структуры описания модуля требуемыми значениями и вызвать функцию **LTR42_Config().** Ниже приводится пример задания конфигурации модуля (определение полей структуры описания модуля):

Будем использовать внутреннюю секундную метку с трансляцией на выход. Метку «СТАРТ» используется внешняя.

```
TLTR42 conf;   // Объявляем  структуру типа TLTR42

conf.AckEna=1;                          // Включаем режим с подтверждениями
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\ltr42api.h"

TLTR42 hltr42;
char ErrorString[255];
char MsgString[255];

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

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

WORD OutputWord;
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=LTR42_Init(&hltr42);

 if(err)
 {
 strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
 oem_printf(ErrorString);
 goto End;
  }
 else
 {
  sprintf(MsgString,"LTR42_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=LTR42_Open(&hltr42, SADDR_DEFAULT, SPORT_DEFAULT, "", CC_MODULE1);
 if(err)
 {
   if(err==LTR_WARNING_MODULE_IN_USE)
    {
    strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
    oem_printf(ErrorString);
    }
   else
    {
     strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
     oem_printf(ErrorString);
     goto End;
    }
  }
 else
  {
  sprintf(MsgString,"LTR42_Open()->OK\n");
  oem_printf(MsgString);
 }

sprintf(MsgString,"Имя модуля: %s\n", hltr42.ModuleInfo.Name);
oem_printf(MsgString);

sprintf(MsgString,"Версия прошивки AVR: %s\n",
hltr42.ModuleInfo.FirmwareVersion);
oem_printf(MsgString);

sprintf(MsgString,"Дата создания прошивки AVR: %s\n",
hltr42.ModuleInfo.FirmwareDate);
oem_printf(MsgString);
```

```
sprintf(MsgString,"Серийный номер модуля: %s\n\n", hltr42.ModuleInfo.Serial);
oem_printf(MsgString);

 // Производим заполнение полей структуры описания модуля требуемыми значениями

hltr42.AckEna=1; // подтверждения включены

/* Конфигурация меток */
hltr42.Marks.SecondMark_Mode=1;     //  Секундная метка внутр. с трансляцией на
выход
hltr42.Marks.StartMark_Mode=0;      //  Метка СТАРТ внутренняя



// Вызываем функцию конфигурации модуля
err=LTR42_Config(&hltr42);
if(err)
 {
 strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
 LTR42_Close(&hltr42);
 oem_printf(ErrorString);
 goto End;
  }
 else
  {
  sprintf(MsgString,"LTR42_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=0xB174;

err=LTR42_WritePort(&hltr42, OutputWord);
if(err)
 {
   strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
   LTR42_Close(&hltr42);
   oem_printf(ErrorString);
   goto End;
  }
 else
   {
  sprintf(MsgString,"LTR42_WritePort()->OK\n");
  oem_printf(MsgString);
 }



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

err= LTR42_StartSecondMark(&hltr42);
if(err)
  {
   strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
   LTR42_Close(&hltr42);
```

```
   oem_printf(ErrorString);
   goto End;
  }
else
  {
  sprintf(MsgString,"LTR42_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= LTR42_StopSecondMark(&hltr42);
if(err)
  {
   strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
   LTR42_Close(&hltr42);
   oem_printf(ErrorString);
   goto End;
  }
else
  {
  sprintf(MsgString,"LTR42_StopSecondMark()->OK\n");
  oem_printf(MsgString);
  }

 // Сгенерируем метку СТАРТ

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

err= LTR42_MakeStartMark(&hltr42);
if(err)
  {
   strcpy(ErrorString, (char *) LTR42_GetErrorString(err));
   LTR42_Close(&hltr42);
   oem_printf(ErrorString);
   goto End;
  }
else
 {
  sprintf(MsgString,"LTR42_MakeStartMark()->OK\n");
  oem_printf(MsgString);
 }


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

   while(!kbhit())
        continue;

        return 0;
}
```

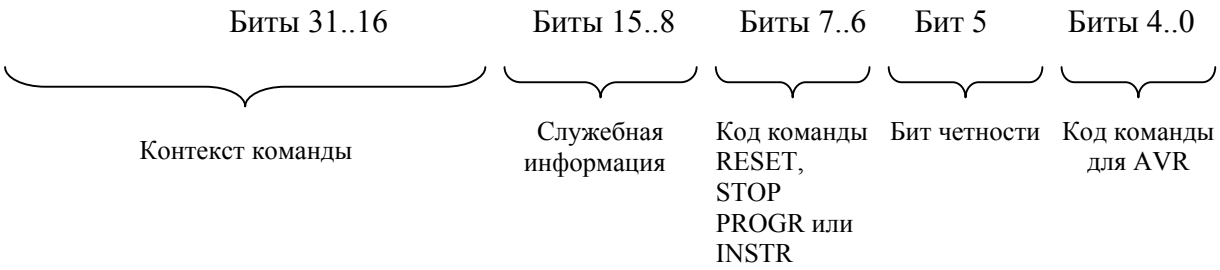
Приложение 2. Протокол обмена данными с модулем.

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 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 LTR42 имеет следующий вид:

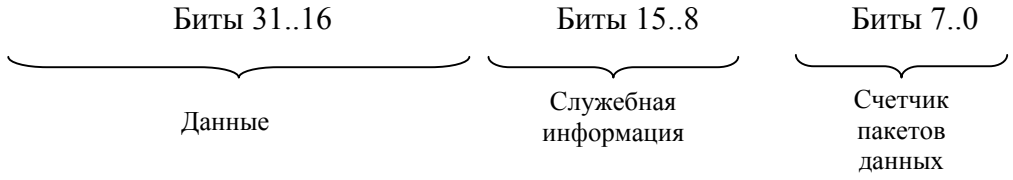


- **Код команды для AVR** – число, определяющее процедуру, которую следует выполнить процессору модуля.
- **Бит четности** – используется для контроля правильности передачи команд в микроконтроллер модуля и в обратном направлении.
- **Код команды RESET, STOP, PROGR или INSTR** – код команды общего интерфейса 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, сведения о режимах генерации меток

  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 вывода данных в порт модуля.

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

Формат **слова данных** имеет следующий вид:



- **Данные** - непосредственно слова, записываемые в порт модуля.
- **Служебная информация -** информация о номере посадочного места, 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 слов данных** – 8-битный счетчик слов данных, отправляемых модулем в крейт-контроллер. При отправке каждого нового слова процессор инкрементирует значение этого счетчика, которое впоследствии используется для проверки правильности следования слов из модуля.