Библиотека пользовательского интерфейса модуля LTR51

Крейтовая система LTR

Руководство программиста

Ревизия 1.0.2

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	20.10.2006	Первая доступная для пользователя ревизия
1.0.1	22.11.2006	Произведены изменения в формате структуры описания модуля и именах полей в целях унификации программного интерфейса всех модулей
1.0.2.	21.03.2007	Изменен пример приложения

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 LTR51

Модуль-частотомер LTR51 предназначен для многоканального измерения частоты сигналов сложной формы. Конструктивно модуль состоит из платы-носителя с размещенными на ней измерительными субмодулями H-51FL или H-51FH, при этом для работы устройства необходимо наличие хотя бы одного субмодуля. На модуль LTR51 могут быть установлены от 1 до 8 двухканальных субмодулей H-51FL или H-51FH в любом сочетании; в минимальной комплектации модуль LTR51 содержит 2 канала, в максимальной – 16 каналов. 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 расположена Программируемая Логическая Интегральная Схема (ПЛИС) ALTERA ACEX EP1K10TC144, содержащая программу, стираемую при пропадании питания микросхемы. Поэтому прошивка для данной ПЛИС загружается в микросхему при каждом новом сеансе работы с модулем, если этому сеансу предшествовало выключение питания.

Также на плате модуля установлен микроконтроллер **AVR Atmega 8515**, осуществляющий общее управление модулем и контролирующий обмен информацией устройства с крейт-контроллером. Управляющая программа микроконтроллера записывается в его Flash-память при изготовлении модуля. В этой памяти также содержится идентификационная информация (серийный номер, версия прошивки ПЛИС, 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 функций пользовательского интерфейса.

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, аппаратный сброс микроконтроллера AVR, загрузка кода прошивки в ПЛИС ACEX EP1K10TC144
- Установка параметров работы (определение требуемых каналов сбора данных, установка порогов компараторов, установка времени счета и/или параметров сбора данных);
- Старт сбора данных;
- Получение и обработка данных (чаще всего этот процесс циклический);
- Останов сбора данных;
- Закрытие интерфейсного канала связи с модулем.

2 Библиотека интерфейса пользователя модуля LTR51, 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 **TLTR51)** предназначена для инициализации, хранения и изменения данных о конфигурации модуля в рамках создаваемой программы. При помощи полей этой структуры задается конфигурация модуля: определяются задействованные физические каналы, устанавливаются параметры измерения частоты, определяются пороги срабатывания компараторов для каждого канала и другие параметры. Перед вызовом функции конфигурации модуля **LTR51_Config()** следует обязательно заполнить поля структуры

описания модуля правильными значениями. В противном случае модуль будет выдавать неверные данные.

2.2 Функции пользовательской библиотеки (общие сведения).

Функции библиотеки пользователя **ltr51api.dll** предназначены для конфигурирования, программирования, сбора данных и диагностики модуля.

2.2.1 Классификация функций пользовательской библиотеки.

Функции, входящие в состав пользовательской библиотеки модуля LTR51, 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 **LTR51_Init()** и **LTR51_Open().** Их следует вызывать в первую очередь. Они предназначены для заполнения полей структуры описания модуля значениями по умолчанию, открытия интерфейсного канала связи с модулем, аппаратного сброса модуля, загрузки ПЛИС ALTERA ACEX, определения наличия субмодулей и выполнения необходимых проверок. Без вызова этих функций дальнейшая работа с модулем невозможна.

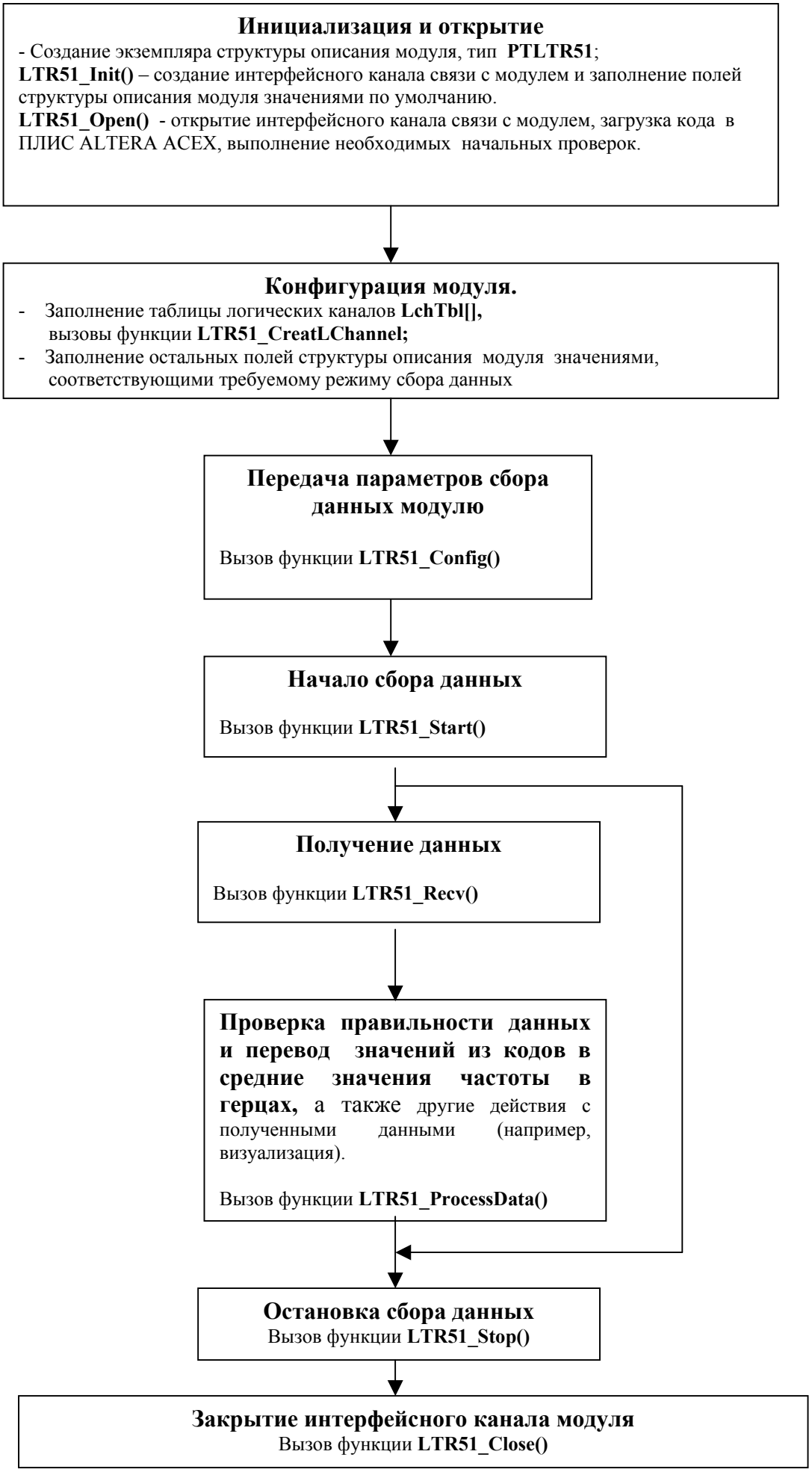
Функция конфигурирования – LTR51_Config(). Данная функция загружает все конфигурационные параметры, записанные в полях структуры описания модуля, в оперативную память микроконтроллера AVR. После выполнения указанных функций устройство готово к сбору данных.

Функции сбора данных: LTR51_Start(), LTR51_Stop(), LTR51_Recv(), LTR51_ProcessData(). Они предназначены для старта и остановки сбора данных, для получения данных, для проверки их целостности и для вычисления средних значений частоты сигнала.

Вспомогательные функции: LTR51_IsOpened() проверяет, создан ли уже интерфейсный канал связи с модулем; **LTR51_GetErrorString()** возвращает строку с описанием ошибки, соответствующую коду ошибки; **LTR51_ReadEEPROM ()** выполняет чтение информации из указанной ячейки пользовательского ППЗУ, **LTR51_WriteEEPROM()** выполняет запись информации в указанную ячейку пользовательского ППЗУ.

Функция закрытия: LTR51_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 для LTR51.



При программировании модуля следует придерживаться указанной выше схемы. Далее будут подробно рассмотрены компоненты пользовательской библиотеки **ltr51.dll.**

3 Подробное описание библиотеки ltr51api.

Для получения возможности вызова интерфейсных функций библиотеки ***ltr51api.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 **ltr51api.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\ltr51api.lib** и **LTR\INCLUDE\ltr51api.h**;

  Microsoft Visual C++ :
  - подключить к проекту файлы **LTR\LIB\MSVC\ltr51api.lib** и **LTR\INCLUDE\ltr51api.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 наличии субмодулей в слотах платы-носителя, о параметрах сбора данных (время счета, частота дискретизации, **BASE**, количество логических каналов, Таблица Логических Каналов и т.д.). Также в эту структуру считывается идентификационная запись модуля при вызове функции **LTR51_Open()** и информация о доступных физических каналах, полученная после опроса микроконтроллером AVR**.** Определение типа, соответствующего структуре описания модуля, приводится ниже:

```
typedef struct
  {

   INT size;                  // Размер структуры
   TLTR Channel;              // Интерфейсный канал связи с модулем
   WORD ChannelsEna;          // Маска доступных каналов

   INT LChQnt;                // Количество логических каналов
   DWORD LChTbl[16];          // Таблица Логических Каналов

   INT  SetUserPars;          // Указывает, задаются ли параметры Fs и Base пользователем

   double Fs;                 // Частота дискретизации, Гц
   WORD Base;                 //  Значение BASE
   double F_Base;             //  Частота измерений, Гц
    INT AcqTime;              // Время счета
    INT TbaseQnt;             // Количество периодов измерения, обеспечивающих данное
                              // время счета
   TINFO_LTR51    ModuleInfo;    // Идентификационная информация модуля


 } TLTR51, *PTLTR51; // 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.

ChannelsEna	**WORD**	Маска доступных физических каналов. Это поле получает значение после вызова функции **LTR51_Open()**. Значение каждого бита определяет, доступен ли соответствующий физический канал. Биты <15..0> соответствуют физическим каналам 16..1. Если значение бита «1», соответствующий канал подключен, если «0» - канал недоступен. Косвенно значение этой маски определяет наличие субмодулей в слотах платы-носителя
LChQnt	**INT**	Количество используемых логических каналов. Разъяснение понятия “логический канал” см. ниже, в *гл. 5.1* настоящего Руководства.
LChTbl[16]	**DWORD**	Таблица логических каналов. Разъяснение понятия “таблица логических каналов” см ниже, в *гл. 5.1* настоящего Руководства.
SetUserPars	**INT**	Флаг установки пользовательских значений для параметров **Fs** и **Base**. Если значение флага 0, то при сборе данных используются оптимальные параметры: **Fs**=500 кГц, **BASE**=5000. Изменение полей **Fs** и **Base** вручную в данном случае эффекта не даст. Если значение флага ненулевое, то значения полей **Fs** и **Base** определяются пользователем.
Fs	**double**	Значение частоты дискретизации в Герцах. Допустимые значения – от 306 Гц до 500 кГц
Base	**WORD**	Значение параметра **BASE**. Допустимые значения – от 70 до 65535. Параметр определяет период и частоту измерения: Период измерения: $\mathbf{T_{BASE}=BASE/Fs}$
F_Base	**double**	Частота измерения, Гц. $\mathbf{F_{BASE}=Fs/BASE=}$ $\mathbf{=1/\ T_{BASE}}$
AcqTime	**INT**	Величина времени счета в миллисекундах
TbaseQnt	**INT**	Количество периодов измерения, обеспечивающее время счета, равное **AcqTime.**
ModuleInfo		Идентификационная информация модуля. Тип – **TINFO_LTR51.** Данный тип представляет собой структуру. Ее описание приводится ниже в этой главе настоящего Руководства

Для задания режима работы модуля пользователю необходимо самостоятельно (вручную) определить следующие поля этой структуры: **LChQnt**, **LChTbl[]**, **SetUserPars** и **AcqTime**. Также в зависимости от значения **SetUserPars** может потребоваться определение полей **Fs** и **Base**. Поля **Channel**, **TbaseQnt, F_Base** и **ModuleInfo** (а если **SetUserPars**=0, то также **Fs** и **Base**) являются выходными, их значения изменяются функциями, о которых речь пойдет ниже. В конце данного Руководства приводятся примеры задания конфигурации.

Идентификационная информация модуля:

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

} TINFO_LTR51,*PTINFO_LTR51;
```

3.2 Описание функций библиотеки ltr51api.dll.

Формат: INT LTR51_Init(PTLTR51 hnd)
Параметр: hnd – указатель на структуру описания модуля (тип **PTLTR51)**

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(TLTR51);
hnd->LChQnt=16; // Число логических каналов
hnd->SetUserPars=0; // Пользователем задается только время счета
hnd->AcqTime=1000; // Время счета по умолчанию – 1 с
 	
Также эта функция создает 16 логических каналов, соответствующих всем возможным физическим. Для каждого канала значение диапазона порогов +/-1,2 В; значения верхнего и нижнего порогов соответственно равны 0.8 и 0.2 В. Также для всех каналов режим выделения активных перепадов – фронты.
 	
// При инициализации все поля идентификационной записи будут содержать только «\0»

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

Формат: INT LTR51_Open(PTLTR51 hnd, DWORD net_addr, WORD net_port, CHAR *crate_sn, INT slot_num, CHAR *ttf_name)

Параметры:

- **hnd** – указатель на структуру описания модуля (тип **PTLTR51);**
- **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 крейта, в котором расположен модуль (***нумерация с единицы!***);
- **ttf_name** – полный путь к текстовому файлу **ltr51.ttf** с кодом прошивки, который данная функция загрузит в ПЛИС ALTERA ACEX. Без корректной передачи микроконтроллеру кода, считанного из этого, дальнейшая работа программы невозможна.

Описание: Функция открывает интерфейсный канал связи с модулем, загружает код прошивки в ПЛИС ALTERA ACEX, выполняет необходимые проверки, а также считывает из ППЗУ модуля его идентификационную запись. Также функция определяет доступные физические каналы, информация о которых будет содержаться в поле **ChannelsEna** структуры описания модуля. После работы функции в соответствующих полях идентификационной подструктуры будут находиться: версия управляющей программы микроконтроллера AVR, дата ее создания, версия прошивки ПЛИС, имя модуля и серийный номер модуля, а также информация о доступных физических каналах.

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 LTR51_Close(PTLTR51 hnd)

Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR51**

Описание: Выполняет закрытие интерфейсного канала связи с модулем. Эту функцию следует вызывать всегда перед окончанием работы с модулем.

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

Формат: INT LTR51_IsOpened(PTLTR51 hnd)

Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR51**

Описание: Выполняет проверку: создан ли интерфейсный канал связи с модулем. Если функция возвращает нулевой результат – канал создан. Если ненулевой, то не создан

Возвращаемое значение: код ошибки, тип **int.** Если “0” – интерфейсный канал связи с модулем создан

Формат: INT LTR51_CreateLChannel(INT PhysChannel,
double *HighThreshold,
double *LowThreshold,
INT ThresholdRange, int EdgeMode)

Параметры:

- **PhysChannel** – номер физического канала, связываемого с создаваемым логическим. *Нумерация с единицы*.
- ***HighThreshold** – указатель на значение напряжения верхнего порога гистерезиса для данного физического канала;
- ***LowThreshold** – указатель на значение напряжения нижнего порога гистерезиса для данного физического канала;
- **ThresholdRange** – диапазон напряжений порогов гистерезиса;
- **EdgeMode** - режим выделения активных перепадов. «0» - по фронтам, «1» - по спадам.

Описание: Функция создает логический канал, т.е. формирует 32-битное слово, которое следует записать в соответствующую ячейку таблицы логических каналов. Подробнее о формате слова см. 5.1. настоящего Руководства.

Возвращаемое значение: Сформированное значение логического канала, тип **INT.**

Формат: INT LTR51_Config(PTLTR51 hnd)

Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR51**

Описание: функция передает модулю конфигурационную информацию, определенную ранее в полях структуры описания модуля. Информация о диапазонах порогов гистерезиса, о значениях верхнего и нижнего порогов, о режимах выделения активных перепадов передается микроконтроллеру модуля. Кроме того, передается информация и о величинах частоты дискретизации, параметра **BASE** и времени счета. Перед вызовом этой функции пользователю обязательно следует установить правильные значения для полей **LChQnt**, **LChTbl[]**, **SetUserPars**, **AcqTime** (если требуется, то также **Fs** и **Base**) структуры описания модуля. Очевидно, что предварительно должна быть создана Таблица Логических Каналов. Без правильной установки полей структуры и без последующего вызова функции **LTR51_Config()** дальнейшая работа модуля может быть некорректной. Поэтому стартовать сбор данных следует *только после выполнения этой функции*.

Если поле **SetUserPars**=1, то после работы функции значения полей **Fs**, **Base** и **AcqTime** будут подкорректированы, а значение поля **TbaseQnt** – рассчитано.

Если поле **SetUserPars**=0, то после работы функции поле **Fs** примет значение 500 000.0, **Base** будет равно 5000, **AcqTime** будет подкорректировано, а **TbaseQnt** – рассчитано.

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

Формат: INT LTR51_Start(PTLTR51 hnd)

Параметр: hnd – указатель на структуру описания модуля, тип **PTLTR51;**

Описание: Запускает сбор данных. Модуль начинает выдавать слова данных, содержащие параметры частотного сигнала **N** и **M**.

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

Формат: INT LTR51_Stop(PTLTR51 hnd)
Параметры: hnd – указатель на структуру описания модуля, тип **PTLTR51;**
Описание: Останавливает сбор данных
Возвращаемое значение: код ошибки, тип **int.** Если “0” – функция выполнилась без ошибок

Формат: INT LTR51_Recv(PTLTR51 hnd, DWORD *data, DWORD *tmark, DWORD size, DWORD timeout)
Параметр: hnd – указатель на структуру описания модуля **PTLTR212;** ***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 размером **size** . Полученные слова при выходе из функции содержатся 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 **LTR51_GetErrorString().** *Примечание: Описание этой функции соответствует описанию функции **LTR_Recv()** библиотеки крейт-контроллера ltrap.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 LTR51_ProcessData(PTLTR51 hnd, DWORD *src, DWORD *dest, double *Frequency, DWORD *size);
Параметры: hnd – указатель на структуру описания модуля, тип **PTLTR51;** ***src –** указатель на массив с исходными («сырыми») словами данных, полученными из модуля. Каждая пара слов содержит информацию о параметрах частотного сигнала **M** и **N**. ***dest –** массив сформированных 32-битных слов данных, где информация о параметрах частотного сигнала **M** и **N** приведена в более удобном формате и удалены все служебные поля. В данном массиве содержится информация только для всех *логических* каналов. ***Frequency –** указатель на массив, который после работы функции будет содержать значения средней частоты для каждого *логического* канала. ***size –** указатель на переменную, хранящую количество элементов. При входе в функцию указывает на количество элементов во входном массиве **src[]**, на выходе функции указывает на количество элементов в выходном массиве **dest[]**.

Описание: Выполняет проверку целостности потока данных, поступающих из модуля в процессе сбора данных, а также выполняет расчет средней частоты для всех определенных логических каналов. Кроме того, для всех логических каналов функция удаляет служебные поля из слов данных и выдает массив с только параметрами частотного сигнала **M** и **N**, предоставляя пользователю возможность самостоятельно обрабатывать данные и использовать модуль в для решения других задач, отличных от измерения средней частоты. Например, для подсчета импульсов или измерения временнЫх интервалов.

При запущенном сборе данных пользователю необходимо самостоятельно получать слова-данные из модуля при помощи функции **LTR51_Recv(),** а затем передавать полученный массив функции **LTR51_ProcessData().** Вместо функции **LTR51_Recv()** можно воспользоваться функцией библиотеки крейт-контроллера **LTR_Recv()**, но вместо параметра **hnd** применить **hnd->Channel**. После работы функции массив, адресуемый указателем ***dest**, будет содержать 32-битные слова, в каждом из которых представлена информация как об **N**, так и об **M** (в отличие от входного массива **src[]**, где для кодировки двух этих параметров использовались два слова данных). Причем в массиве **dest[]** будет содержаться информация только по *логическим* каналам. Параметр ***size** при входе в функцию указывает на размер исходного массива **src[]**, при выходе из функции – на размер выходного массива **dest[]**. Подробнее о непрерывном чтении данных, о применении функции **LTR51_ProcessData()** и ее параметрах см. в *гл. 7.1* настоящего Руководства.

Если функция выявляет сбой в последовательности значений счетчика отправляемых слов-данных, то ее выполнение немедленно прерывается и возвращается код ошибки. Параметр ***size** в этом случае указывает на количество верных слов данных в массиве **dest[]**, которые удалось сформировать до сбоя.

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

Формат: INT LTR51_ReadEEPROM(PTLTR51 hnd, INT Address, BYTE *val)

Параметры:
- **hnd –** указатель на структуру описания модуля, тип **PTLTR51;**
- **Address –** 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 LTR51_WriteEEPROM(PTLTR51 hnd, INT Address, BYTE val)

Параметры:
- **hnd –** указатель на структуру описания модуля, тип **PTLTR51;**
- **Address –** 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 LTR51_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 TLTR51) и ***УКАЗАТЕЛЬ*** на структуру описания модуля (тип PTLTR51).

4 Инициализация и открытие модуля

Перед началом работы с модулем необходимо провести инициализацию и открытие интерфейсного канала модуля. Это достигается путем последовательного вызова двух функций: **LTR51_Init()** и **LTR51_Open()**.

Функция **LTR51_Init()** выполняет заполнение полей структуры описания модуля значениями по умолчанию, а также инициализирует интерфейсный канал связи с модулем.

Функция **LTR51_Open()** выполняет открытие интерфейсного канала связи с модулем, загрузку кода прошивки в ПЛИС ALTERA ACEX, начальные проверки работы низкоуровнего программного обеспечения, а также определяет доступные каналы в субмодулях. Прошивка ПЛИС находится в прилагаемом текстовом файле **ltr51.ttf**. Полный путь к этому файлу должен быть указан в параметре ***ttf_name** функции **LTR51_Open()**. При вызове функции программный код из указанного файла будет передан ПЛИС, а затем микроконтроллер AVR произведет тестирование интерфейса между самим микроконтроллером и ПЛИС с выдачей команды-ответа об удачном или неудачном завершении теста. Если загрузка ПЛИС не выполнится надлежащим образом, дальнейшая работа с модулем будет невозможна.

После работы функции **LTR51_Open()** изменится значение поля **ChannelsEna** структуры описания модуля. Это 16-битное поле представляет сбой маску доступных физических каналов.

Физическим каналом называется аппаратный канал ввода данных, используемый для подключения электрического сигнала к модулю LTR51. Модуль имеет 16 физических каналов, по 2 канала на каждом из 8-ми возможных субмодулей H51-Fxx. Полная информация о физических каналах и субмодулях указана в книге «Крейтовая система LTR. Руководство пользователя». Значение каждого бита маски **ChannelsEna** индицирует, доступен ли соответствующий физический канал. Биты <15..0> данного поля соответствуют физическим каналам 16-1. Значение бита «1» говорит о том, что канал подключен, «0» - что канал недоступен. Как правило, нулевые значения в каких-либо битах **ChannelsEna** свидетельствуют об отсутствии в соответствующем слоте субмодуля H-51Fxx. Используя значение маски **ChannelsEna,** можно узнать конфигурацию субмодулей: какие из возможных 16-ти субмодулей имеются в слотах, а в каких слотах субмодули отсутствуют. Например, если после вызова функции **LTR51_Open()** значение **ChannelsEna** равно 1100001100111111 BIN (0xC33F HEX), то это указывает, что доступны каналы 1,2,3,4,5,6,9,10,15,16. Значит, субмодули имеются только в 1,2,3,5 и 8 слотах платы-носителя. Остальные слоты пусты. Если даже при наличии субмодуля в слоте платы-носителя какие-либо биты в маске будут иметь значение «0», то это говорит о неисправности соответствующих физических каналов ввода данных. Поэтому после открытия модуля вызовом функции **LTR51_Open()** всегда полезно сверять фактическую конфигурацию субмодулей H-51Fxx и ее соответствие значению маски **ChannelsEna**. Если выявлено несоответствие – значит, некоторые физические каналы неисправны.

Соответствие субмодулей и физических каналов приведено в следующей таблице:

Таблица 2

Номер слота субмодуля на плате-носителе	Номера физических каналов
1	1,2
2	3,4
3	5,6
4	7,8
5	9,10
6	11,12
7	13,14
8	15,16

5 Таблица логических каналов

5.1 Создание таблицы логических каналов

Одно из полей структуры описания модуля – Таблица Логических Каналов **LChTbl[]**– представляет собой массив, содержащий информацию об используемых физических каналах модуля, соответствующих им порогах гистерезиса, а также о последовательности опроса каналов. В отличие от физического канала, **Логический канал** представляет собой 32-битное слово специального формата, содержащее информацию о значениях порогов гистерезиса и режиме выделения активных перепадов для данного физического канала. **Таблицей Логических Каналов** называется массив, каждый элемент которого представляет собой логический канал. Совокупность элементов этой таблицы определяет те физические каналы, которые будут обрабатываться на верхнем уровне и для которых будет производиться подсчет средней частоты. Для физических каналов, не представленных в Таблице Логических Каналов, подсчет средней частоты производиться не будет. Заполнение этого массива (таблицы логических каналов) *осуществляется пользователем*. Для создания логического канала следует вызвать функцию **LTR51_CreateLChannel(),** где в параметрах нужно указать номер физического канала, связываемого с данным логическим, значения верхнего и нижнего порогов гистерезиса, режим выделения активных перепадов, а также диапазон напряжения порогов (+/-1,2 В или +/-10 В). Функция возвращает сформированное значение логического канала, которое следует записать в Таблицу вручную. В Таблице Логических Каналов элементы, соответствующие различным физическим каналам модуля, **не обязательно** должны располагаться последовательно, в порядке возрастания номеров физических каналов. При работе модуля данные будут приходить со всех 16-ти возможных физических каналов, независимо от того, во все ли слоты установлены субмодули. Однако при вызове функции **LTR51_ProcessData()** среднее значение частоты будет рассчитываться **только** для каналов, определенных в Таблице Логических Каналов, а данные с остальных физических каналов будут при этом игнорироваться. Выходной массив **dest[]** также будет содержат данные *ТОЛЬКО* для каналов, внесенных в Таблицу Логических Каналов. Поле структуры описания модуля **LChQnt** должно содержать число задействованных логических каналов (оно же – число элементов таблицы логических каналов). Максимальное число логических каналов – 16.

Важно помнить, что Таблица Логических Каналов **не должна содержать неинициализированных полей в диапазоне индексов от 0 до LChQnt-1.** Т.е. если заявлено количество логических каналов, равное значению поля **LChQnt**, то все элементы массива **LchTable[]** с индексами от 0 до **LChQnt**-1 должны быть заполнены верными значениями логических каналов. В противном случае данные, выдаваемые модулем, будут некорректными. **Всегда следует определять количество задействованных логических**

каналов и следить за тем, чтобы все элементы Таблицы Логических Каналов и индексами от 0 до LChQnt-1 были инициализированы верными значениями!

Необходимо отметить, что программируемые потенциометры субмодулей имеют 256 позиций, поэтому значения порогов гистерезиса представляют собой дискретный ряд. После работы функции **LTR51_CreateLChannel()** параметры ***HighThreshold** и ***LowThreshold** будут указывать на исправленные значения напряжений порогов, которые, вообще говоря, могут отличаться от задаваемых пользователем.

Формат логического канала (32-битное слово):

Биты 24..31 Биты 16..23 Биты 8..15 Биты 0..7

Биты C <0..7> - номер физического канала, связываемого с данным логическим (нумерация с нуля);
Биты T <8..15> - Режим выделения активных перепадов для данного физического канала;
Биты L <16..23> - Значение кода потенциометра для установки величины напряжения нижнего порога гистерезиса;
Биты H <24-31> - Значение кода потенциометра для установки величины напряжения верхнего порога гистерезиса;

Значение кода потенциометра рассчитывается следующим образом:

Uп=128*(Ku*U/ Uref + 1), где **U** – устанавливаемое напряжение порога,
Uref – опроное напряжение; Uref=2,048 В.
Ku= –1,6737 для диапазона напр. порогов +/-1,2 В;
Ku= –0,2010 для диапазона порогов +/-10 В

Формат логического канала здесь приведен как справочная информация. Пользователю не требуется самостоятельно формировать значения этих слов и рассчитывать коды потенциометров. Это делается автоматически при помощи функции **LTR51_CreateLChannel().** Пользователю только следует вызвать эту функцию с требуемыми параметрами, а возвращаемое значение и будет являться сформированным логическим каналом, которое затем следует записать в Таблицу Логических Каналов под требуемым индексом.

Например, требуется создать логический канал для 3-го физического канала со следующими параметрами: диапазон напряжений порогов гистерезиса: +/- 1,2 В; режим выделения активных перепадов - по фронтам; значение напряжения верхнего порога +0.7 В, нижнего - + 0.2 В.

```
TLTR51 conf;
DWORD LChannel;
double HighThreshold=0.7000;
double LowThreshold=0.2000;
int ThresholdRange=1;
int EdgeMode=0;

LChannel= LTR51_CreateLChannel (3, &HighThreshold, &LowThreshold,
                                ThresholdRange, EdgeMode);
```

После работы функции переменная **LChannel** примет следующее значение:
LChannel =0x376B0002. Также функция изменит значения входных параметров, **HighThreshold** и **LowThreshold**, и по возвращении из нее они будут иметь следующие значения:

HighThreshold=0.6979;
LowThreshold=0.2008.

После этого следует записать полученное значение в таблицу логических каналов. Пускай это будет элемент таблицы с индексом 2:
conf.LChTbl[2]= LChannel;

Еще пример: имеем 4 субмодуля Н-51хх и требуется снимать информацию со всех 8-ми физических каналов с одинаковыми параметрами: диапазон порогов +/-10 В, напряжение верхнего порога +7 В, напряжение нижнего порога –5 В. Тогда Таблицу Логических Каналов формируем следующим образом:

```
TLTR51 conf;
DWORD LChannel;

double HighThreshold= 7.000;   // Напряжение верхнего порога, В
double LowThreshold= -5.000;  //  Напряжение нижнего порога, В
int ThresholdRange=0;          // Диапазон порогов +/- 10 В
int EdgeMode=0;                // Для всех каналов режим выдел. акт. перепадов – по фронтам

conf.LChQnt=8;                 // Количество логических каналов 8



for(i=0; i< conf.LChQnt; i++)
 {
 LChTbl[i]= LTR51_CreateLChannel(i+1, &HighThreshold, &LowThreshold,
                                 ThresholdRange, EdgeMode);
}
```

Следует иметь ввиду, что физические каналы при вызове данной функции нумеруются с единицы (1-16), а индексы элементов Таблицы Логических каналов – с нуля (0-15).

Обязательно нужно следить за правильным значением параметра **ThresholdRange**. Одни и те же величины кодов потенциометров соответствуют разным значениям порогов для диапазонов +/-1,2 В и +/-10 В. Если значение параметра **ThresholdRange** не соответствует положению джампера для рассматриваемого канала, то данные, выдаваемые модулем, будут неверными. Программной возможности считывать положение джамперов, переключающих диапазоны установки порогов, на данный момент нет. Поэтому соответствие программных настроек и аппаратного положения джамперов должно контролироваться пользователем. Для удобства заданную конфигурацию джамперов можно записать в ППЗУ микроконтроллера AVR в произвольном формате. Используя функции записи/чтения байтов в/из ППЗУ, пользователь может сохранять любую информацию, в том числе и положение джамперов в различных каналах.

6 Особенности конфигурации модуля и задание режима сбора данных

6.1 Принцип формирования выходных данных и подсчет средней частоты сигнала

Для получения достоверных данных о частотном сигнале и для возможности расчета средней частоты без потери информации, данные, поступающие из модуля, содержат два параметра частотного сигнала для каждого физического канала: **N** и **M**. Подробнее о принципе работы модуля и о методике подсчета средней частоты см. ниже в этой главе. Дальнейшие сведения предполагают знание пользователем основ аппаратной структуры модуля и схемы селекции по уровню. Подробно об этом написано в гл. 10.3 документа «Крейтовая система LTR. Руководство пользователя».

Выявлением наличия или отсутствия активных перепадов на выходах триггеров занимается ПЛИС ALTREA ACEX, установленная на плате модуля. С определенной частотой, называемой частотой дискретизации, ПЛИС сканирует выходы триггеров для всех физических каналов и определяет уровень напряжения на выходе каждого триггера. Отсюда выведем определение:

Частотой дискретизации **Fs** называется частота сканирования уровней напряжения на выходах триггеров. Эта частота одинакова для всех физических каналов, и сканирование происходит согласованно.

Периодом дискретизации **τ** называется интервал времени между двумя ближайшими актами сканирования выходного напряжения триггеров. Очевидно, что частота дискретизации и период дискретизации являются взаимно обратными величинами: **τ=1/Fs.** Период дискретизации определяет *временное разрешение* частотомера.

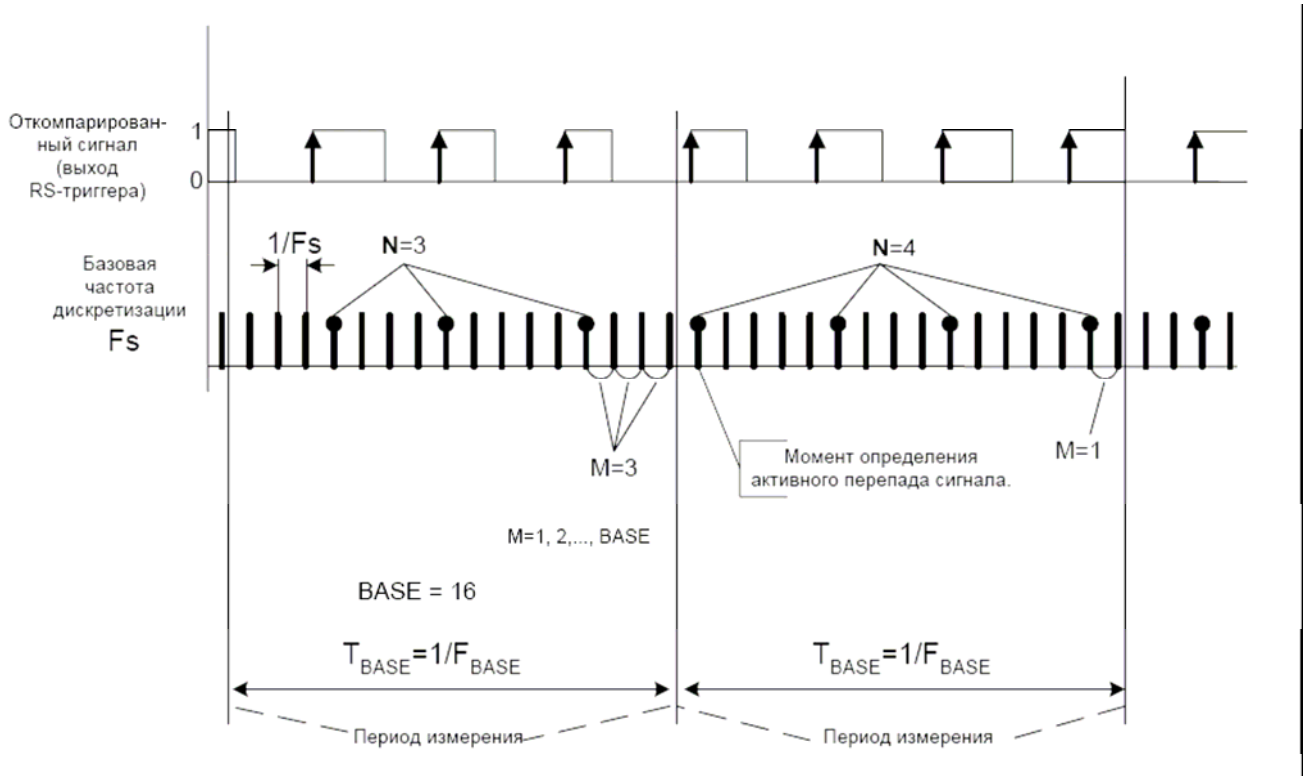


Рис. 1.

Используя поступающую с частотой **Fs** информацию об уровнях выходного напряжения триггеров, ПЛИС подсчитывает количество активных перепадов данного напряжения и делает это в течение временного интервала фиксированной длительности, называемого периодом измерения.

Периодом измерения $\mathbf{T_{BASE}}$ называется интервал времени, в течение которого ПЛИС подсчитывает количество активных перепадов выходного напряжения триггера.

Частотой измерения называется величина, обратная периоду измерения: $\mathbf{F_{BASE}=1/T_{BASE}}$.

Длительность периода измерения определяется значением специального декрементного перезагружаемого 16-битного счетчика **BASE**, значение которого задается пользователем. Значение счетчика уменьшается на единицу каждый период дискретизации $\boldsymbol{\tau}$ и перезагружается базовым значением **BASE** при достижении нуля. Диапазон значений: 70<=**BASE**<=65535.

Период измерения равен $\mathbf{T_{BASE}}$ **=BASE*** $\boldsymbol{\tau}$**= BASE/Fs.** Частота измерения соответственно $\mathbf{F_{BASE}}$=1/ $\mathbf{T_{BASE}}$=**Fs/BASE**. Например, если частота дискретизации **Fs**=500 кГц, а значение **BASE**=70, то величина периода измерения будет равна $\mathbf{T_{BASE}}$=70/500000=140 мкс. Частота измерения соответственно $\mathbf{F_{BASE}}$ =**Fs/BASE**=7142,86 Гц.

По истечении периода измерения $\mathbf{T_{BASE}}$ ПЛИС выдает в интерфейс крейта два слова данных, содержащих параметры частотного сигнала **N** и **M:**

N - количество активных перепадов входного сигнала за период измерения $\mathbf{T_{BASE}}$.

M – время, прошедшее от последнего активного перепада в данном периоде измерения до окончания этого периода, выраженное в количестве периодов дискретизации (т.е. в $\boldsymbol{\tau}$**)**.

Первое слово, выдаваемое ПЛИС, содержит значение параметра **M** для данного интервала измерения, второе слово – значение параметра **N**. Как **N**, так и **M** являются 16-битными целыми числами.

В каждом последующем периоде измерения $\mathbf{T_{BASE}}$ значения параметров **N** и **M** обнуляются, и их расчет выполняется снова. По окончании очередного периода измерения в интерфейс крейта вновь выдаются два слова данных со значениями **M** и **N**.

Например, имеем следующие установки: **Fs**=500 кГц, **BASE** =5000. С учетом того, что значение периода дискретизации $\boldsymbol{\tau}$=1/**Fs**=2 мкс**,** длительность периода измерения $\mathbf{T_{BASE}}$ равна 5000* $\boldsymbol{\tau}$=5000*2 мкс и составляет 10 мс.

Таким образом, при таких настройках каждые 10 мс модуль будет выдавать по два слова данных с каждого канала. Старшие 16 бит первого слова будут содержать значение параметра **M**, старшие 16 бит второго слова – значение параметра **N**.

ВНИМАНИЕ! *Модуль LTR51 выдает не значения частоты, а параметры частотного сигнала. Такой подход расширяет круг решаемых задач: например, становится возможной регистрация отдельных импульсов и подсчет импульсов.*

Возможны следующие частные случаи:

- Если в течение периода измерения не было зафиксировано ни одного активного перепада, то будут выданы следующие значения: **N=0**, **M=BASE**.
- **M=BASE, N=1**. Это соответствует приходу активного перепада в последнем периоде дискретизации предыдущего периода измерения, причем в текущем периоде измерения ни одного активного перепада зафиксировано не было.

Независимо от наличия субмодулей Н-51хх, данные при запущенном сборе приходят со всех 16-ти физических каналов. Поэтому в сумме по истечении каждого периода измерения в интерфейс крейта высылается модулем 32 слова данных: по два для каждого из 16-ти каналов. Помимо информации о частоте сигнала (параметры **N** и **M**), каждое слово данных содержит служебные поля: счетчики и номер физического канала, к которому относится рассматриваемое слово данных.

Исходя из параметров **N** и **M** несложно рассчитать среднюю частоту за какой-либо интервал времени, называемый *временем счета*.

Временем счета Tc называется промежуток времени набора данных, необходимый для корректного определения среднего значения частоты (периода) сигнала. Время счета состоит из конечного числа периодов измерения $\mathbf{T_{BASE}}$, причем в этот интервал должны попасть *по*

крайней мере два периода измерения (из их непрерывной последовательности), каждый из которых содержит *хотя бы по одному* активному перепаду. Очевидно, что время счета *всегда кратно* периоду измерения.

Однако собственно вычисление частоты базируется на использовании понятия расчетного интервала. Дальнейшее изложение методики иллюстрируется на Рис. 2.

Расчетный интервал Tw определяется как время между крайними активными перепадами сигнала внутри времени счета.

Принимая во внимание определения **Fs** и **BASE**, можем определить расчетный интервал **Tw**. При этом время счета **Tc** включает в себя **k** периодов измерения:

$Tw=(M_1+BASE(k-1)-Mk)*\tau=(M_1+BASE(k-1)-Mk)/Fs,$ где **BASE**- величина базового значения счетчика BASE, **M_1** – значение параметра **M** в первом периоде измерения для данного времени счета, **Mk** – значение параметра **M** в последнем (**k**-м) периоде измерения для данного времени счета.

Так как разрешение частотомера – один период дискретизации, то значение интервала **Tw** имеет точность **τ**. Это значит, что величина расчетного интервала **Tw** может отличаться от фактического времени между крайними активными перепадами **$Tw_{факт}$** не больше, чем на **τ.** **$Tw_{факт}$ - Tw <= ±τ.** Конечно, это действительно без учета иных факторов, влияющих на точность: погрешность частоты тактового генератора опорной частоты, погрешность установки порогов, погрешность запуска. Поскольку модуль может определить только измеренное, а не фактическое значение расчетного периода, то, очевидно, все расчеты производятся с использованием **Tw,** а не **$Tw_{факт.}$**

Вообще говоря, расчетный интервал **Tw** не равен времени счета **Tc**. Значение **Tw** может незначительно отличаться от **Tc**, а может быть много меньше. Это зависит от соотношения времени счета **T_c** и периода сигнала **T**. Как видно из методики расчета и рис. 2, ***время счета Tc должно состоять не менее чем из 2-х периодов измерения*** **T_{BASE}**. Подсчет средней частоты на одном периоде измерения **T_{BASE}** теряет смысл. Кроме того, для корректного подсчета периода и частоты сигнала, время счета должно содержать по крайней мере два активных перепада.

Итак, если имеем интервал некое время счета **Tc**, то общее число активных перепадов, зафиксированное в течение этого времени, составляет

$\sum_{i=2}^{k} N_i$, где k-количество периодов измерения **T_{BASE}** в данном времени счета **Tc**.

Отсюда нетрудно вычислить расчетное значение среднего периода сигнала: $T= Tw/\sum_{i=2}^{k} N_i$.

или $T= (M_1+BASE(k-1)-Mk)/\sum_{i=2}^{k} N_i$.

Соответственно значение средней частоты сигнала $f= 1/T = \sum_{i=2}^{k} N_i /(M_1+BASE(k-1)-Mk)$.

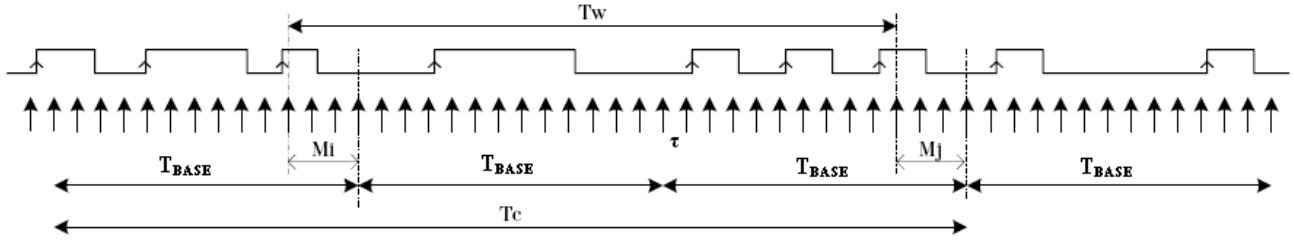


Рис. 2

Следует обратить внимание, что здесь суммирование **N** производится от второго периода измерения до последнего. Из первого периода измерения берется только значение $\mathbf{M_1}$. Значение $\mathbf{N_1}$ для вычисления среднего периода не требуется.

Конечно, фактическое значение периода сигнала $\mathbf{T_{факт}}$ будет отличаться от расчетного среднего периода **T**. Кроме того, фактический период вообще может быть величиной непостоянной. Поэтому напомним, что мы рассчитываем значение ***среднего периода (средней частоты) сигнала за время счета Tc.***

Fs, BASE и Tc являются основными параметрами сбора данных в модуле LTR51. Соотношение значений этих параметров определяют как точность полученных данных, так и загрузку интерфейса при движении потока слов от модуля к крейт-контроллеру и ПК.

Исходя из вышеизложенной методики расчета можно сделать следующие выводы и сформулировать некоторые правила:

- Поскольку время счета **Tc** состоит из конечного числа периодов измерения $\mathbf{T_{BASE}}$, то уменьшение значения параметра **BASE** приведет к возрастанию числа периодов измерения в течение времени счета и, соответственно, к большему потоку данных от модуля. Но, с другой стороны, увеличение **BASE** и, следовательно, уменьшение количества периодов измерения $\mathbf{T_{BASE}}$ за время счета **Tc** делают задание этого времени менее гибким. Это связано с тем, что возможные значения времени счета представляют собой дискретный ряд $\mathbf{Tc=n*\ T_{BASE}}$ (**n**>=2 и является целым числом). Если, например, значение **Fs**=10 кГц, **BASE**=65535 и, соответственно, $\mathbf{T_{BASE}}$ =65535/10000=6,55 с, то время счета будет принадлежать следующему ряду: 13,1 с; 26,2 с; 52,4 с и т.д. Как видно, задание времени счета в этом случае не является гибким и может не удовлетворять многим задачам. Но, поскольку данные выдаются в интерфейс каждые 6,55 с, то это заметно снижает загрузку интерфейса крейта.

  Если, например, значение частоты дискретизации **Fs**=10 кГц, но **BASE**=70 и, соответственно, $\mathbf{T_{BASE}}$ =70/ 10000=7 мс, то в этом случае время счета будет представлять следующий ряд: 14 мс, 21 мс, 42 мс и т.д. Очевидно, что при таких параметрах задавать время счета можно очень гибко, но, т.к. модуль будет выдавать данные каждые 7 мс, загрузка интерфейса крейта будет больше.

- Как указывалось выше, время счета **Tc** должно состоять минимум из 2-х периодов измерения $\mathbf{T_{BASE}}$. ***На времени счета, равном одному периоду измерения T_{BASE}, вычисления среднего периода и средней частоты сигнала теряют смысл.***

- Пределы допускаемой относительной погрешности при измерении периода непрерывных синусоидальных или импульсных сигналов вычисляются по формуле:

  $\mathbf{\delta = \pm | (| \delta_o | + | \delta_{зап} | + \tau / Tc) |}$, где

δ_o **-** относительная погрешность по частоте опорного генератора;

δзап - относительная погрешность запуска;

Tc – время счета;

τ – период дискретизации.

Погрешности **δo** и **δзап** являются аппаратными, и программно на них повлиять невозможно. Но отношение **τ/Tc,** очевидно, можно регулировать, изменяя значения периода (частоты) дискретизации и времени счета. Из вышеприведенной формулы видно, что ***значение относительной погрешности уменьшается при уменьшении периода дискретизации (увеличении частоты дискретизации) и при увеличении времени счета.*** Это обязательно надо учитывать при задании параметров сбора данных.

Резюмируя все вышеизложенное, можно сказать, что задание параметров сбора данных является достаточно гибким процессом, и пользователю необходимо выполнить оптимальные настройки для своей конкретной задачи. Однако при использовании библиотеки функций пользовательского интерфейса **ltr51api.dll** имеется возможность задавать только время счета, не изменяя вручную другие параметры. При этом модуль использует по умолчанию значения **Fs**=500 кГц и **BASE**=5000, т.е. период измерения $\mathbf{T_{BASE}}$=5000/500000=10 мс, частота измерения $\mathbf{F_{BASE}}$=100 Гц. Данный режим применим для большого класса задач, обеспечивает оптимальную загрузку интерфейса крейта и позволяет гибко задавать значение времени счета. Если же пользователя не устраивает этот режим, он может сам установить требуемые значения как частоты дискретизации, так и периода измерения. Подробно о программных возможностях при настройке модуля см. в следующей главе.

6.2 Установка параметров сбора данных

Как и в других модулях крейтовой системы LTR, конфигурация модуля выполняется при помощи присваивания полям структуры описания модуля требуемых значений с последующим вызовом функции **LTR51_Config().** В предыдущей главе указывалось, что при задании параметров сбора данных пользователю предоставляются две возможности:

- Задавать *только* время счета при оптимальных фиксированных значениях частоты дискретизации (500 кГц) и параметра **BASE**=5000 (период измерения 10 мс).
- Задавать все параметры: частоту дискретизации **Fs**, **BASE** и время счета **Tc**.

Программно выбор того или иного способа установки параметров выполняется при помощи изменения поля **SetUserPars** структуры описания модуля. Значению параметра **SetUserPars**=0 соответствует режим задания только времени счета. Ненулевому значению **SetUserPars** соответствует режим пользовательской настройки всех параметров.

После вызова функции **LTR51_Config()** значение поля **F_Base** примет значение частоты измерения в Герцах.

6.2.1 Режим настройки только времени счета

При значении поля **UserPars**=0 структуры описания модуля пользователю следует задать только время счета. Значение этого времени в миллисекундах должно быть присвоено полю **AcqTime** структуры описания модуля. Установка значений полей **Fs** и **Base** не даст эффекта, и после вызова функции **LTR51_Config()** эти поля автоматически примут следующие значения: **Fs**=500000, **Base**=5000. Таким образом, при данном режиме поле **AcqTime** является входным, а поля **Fs** и **Base** – выходными.

6.2.2 Режим настройки всех параметров

При ненулевом значении поля **UserPars** структуры описания модуля пользователю требуется самостоятельно задать три параметра: **Fs**, **Base** и **AcqTime**. В этом режиме три указанных поля являются входными. Полю **Fs** должно быть присвоено значение частоты

дискретизации в Герцах (от 306 Гц до 500 кГц). Полю **Base** – значение счетчика **BASE** (от 70 до 65535). Полю **AcqTime** – значение времени счета в миллисекундах. Следует отметить, что после работы функции **LTR51_Config()** значения полей **Fs** и **AcqTime** могут быть подкорректированы, т.к. величины и частоты дискретизации, и времени счета принадлежат к дискретным рядам. Поэтому после выполнения функции **LTR51_Config()** полезно считать значения этих полей структуры описания модуля, чтобы узнать их точные значения.

6.2.3 Количество периодов измерения

После работы функции **LTR51_Config()** изменится также значение поля **TbaseQnt** структуры описания модуля. Оно определяет количество периодов измерения, которое обеспечивает заданное время счета. Указанное значение можно использовать впоследствии при организации чтения данных с вызовом функции **LTR51_Recv(),** т.к. количество периодов измерения соответствует количеству точек сбора на один канал за время счета. При этом следует учесть, что данные приходят со всех 16 возможных физических каналов независимо от наличия субмодулей, и каждый отсчет представлен *двумя* словами данных. Таким образом, при получении данных следует запрашивать количество слов, равное 16*2***TbaseQnt**=32***TbaseQnt**.

В заключении этой главы приведем примеры конфигурирования модуля:

Пример 1. Настраиваем модуль только по значению времени счета, которое полагаем равным 1000 мс. Тогда конфигурирование модуля будет выглядеть следующим образом (предполагается, что экземпляр структуры описания модуля **conf** уже создан, заполнена таблица логических каналов, произведены инициализация и открытие модуля):

```
int err=0; // Переменная для хранения ошибки

conf.SetUserPars=0;  // Используем установку только времени счета
conf.AcqTime=1000;  // Устанавливаем значение времени счета 1000 мс

err=LTR51_Config(&conf); // Вызываем функцию конфигурирования модуля
if(err)
  return(err);
```

После работы функции поля структуры будут иметь следующие значения:

```
conf.Fs=500000;
conf.Base=5000;
conf.AcqTime=1000;
conf.TbaseQnt=100;
```

Пример 2. Задаем вручную как время счета, так и параметры **Fs** и **Base**. Пусть требуется установить режим с частотой дискретизации **Fs**=10 кГц, значение **Base** 10000 (отсюда период измерения равен $\mathbf{T_{BASE}}$=**Base**/**Fs**=1 с, частота измерения – 1 Гц). Время счета – 3 с. Код программы будет выглядеть следующим образом:

```
int err=0; // Переменная для хранения кода ошибки

conf.SetUserPars=1;  // Используем установку параметров сбора вручную
conf.Fs=10000;
conf.Base=10000;
conf.AcqTime=3000;
```

```
err=LTR51_Config(&conf); // Вызываем функцию конфигурирования модуля
if(err)
  return(err);
```

После работы функции поля структуры будут иметь следующие значения:

```
conf.Fs=10000;
conf.Base=10000;
conf.AcqTime=3000;
conf.TbaseQnt=3;
```

Предпочтительным является первый способ конфигурирования. Но если он неприемлем для решения какой-либо конкретной задачи пользователя, то тогда следует воспользоваться ручной настройкой параметров.

7 Особенности выполнения процесса сбора данных.

7.1 Формирование цикла сбора и получения данных

Запуск сбора данных и его остановка осуществляются при помощи функций **LTR51_Start()** и **LTR51_Stop(),** а чтение массивов данных, получаемых из модуля, производится функцией **LTR51_Recv()**. В последней функции параметр **hnd** – указатель на структуру описания модуля; параметр ***data** – указатель на массив, в который запишутся данные из модуля; **size** – количество слов данных, запрашиваемое из модуля. Параметр ***tmark** – указатель на массив синхрометок. Если синхрометки не используются, его можно положить NULL. Параметр **timeout** определяет максимальное время ожидания запрашиваемого числа слов данных в миллисекундах. *Значение таймаута рекомендуется выбирать на секунду дольше времени счета.* Функция возвращает число полученных слов. Если значение, возвращаемое функцией отрицательное, это говорит о том, что функция завершилась с ошибкой и вернула ее код. В этом случае необходимо вызвать функцию **LTR51_GetErrorString()**, чтобы получить описание ошибки, применив ее код в качестве параметра.

В полученном массиве данные располагаются по кадрам. **Кадр** в *этом массиве* представляет собой последовательность слов данных, полученных при выполнении одного цикла опроса всех 16-ти физических каналов. Данные располагаются в кадре в порядке *убывания* номеров физических каналов. Каждый кадр входного массива содержит число элементов, равное ***удвоенному*** числу физических каналов, т.к. для каждого канала модуль выдает два слова данных (подробно об этом в *гл. 6.1*). Поскольку независимо от количества подключенных субмодулей данные приходят со всех 16 возможных физических каналов, кадр входного массива всегда содержит 16*2=32 слова данных. Поэтому параметру ***size** следует присвоить значение, равное произведению количества периодов измерения на 32. Как указывалось в предыдущей главе, при задании требуемого временного интервала и последующего вызова функции **LTR51_Config()** в поле **TbaseQnt** структуры описания модуля возвращается количество периодов измерения, удовлетворяющее заданной продолжительности времени счета. Тогда при вызове функции **LTR51_Recv()** параметру **size** следует присвоить значение **size**=**TbaseQnt***32.

После получения из модуля массива данных следует удалить все служебные поля из его элементов, проверить правильность полученных данных и вычислить значения средней частоты за время счета для всех логических каналов. Для этого предназначена функция **LTR51_ProcessData(PTLTR51 hnd, DWORD *src, DWORD *dest, *Frequency, DWORD *size).** Параметры данной функции:

***src -** представляет собой указатель на исходный массив данных, предварительно полученный функцией **LTR51_Recv().** Это есть «сырые» данные, которые модуль LTR51 непосредственно высылает в крейт-контроллер и ПК**;**

dest** – указатель на выходной массив, который после работы функции будет содержать в своих элементах значения параметров **N** и **M** для всех *логических* каналов. Данные о физических каналах, не представленных в Таблице Логических Каналов, игнорируются и не включены в выходной массив **dest[]**. Элементы выходного массива **dest[]** расположены по кадрам, как и элементы входного массива **src[]**. Отличие в том, что *кадр этого массива содержит число элементов, равное количеству логических каналов, т.е. значению поля* ***LChQnt *структуры описания модуля.*

Располагаются элементы в кадре выходного массива **dest[]** в соответствии с порядком следования элементов в Таблице Логических Каналов. Если, например, мы определили только два логических канала, то кадр в массиве **dest[]** будет состоять из двух элементов. При этом если первый логический канал был связан, например, с физическим каналом 10 при определении Таблицы Логических Каналов, а второй – с физическим каналом 1, то и в кадре массива **dest[]** первый элемент будет содержать информацию **N** и **M** для физического канала 10, второй элемент кадра – для физического канала 1.

Количество кадров в массиве **dest[]** будет соответствовать количеству периодов измерения в требуемом времени счета, т.е. значению поля **TbaseQnt** структуры описания модуля. А общее количество элементов этого массива будет равно произведению **TbaseQnt*LChQnt**.

Параметр ***size** при входе в функцию указывает на размер исходного массива **src[]**, *при выходе из функции – на размер выходного массива* **dest[]**.

Внимание! В любом случае, при задании входного значения параметра size следует помнить, что оно должно быть кратно 32! В противном случае функция LTR51_ProcessData() вернет ошибку!

Если функция выявляет сбой в последовательности значений счетчика отправляемых слов-данных, то ее выполнение немедленно прерывается и возвращается код ошибки. Параметр ***size** в этом случае указывает на количество верных слов данных в массиве **dest[]**, которые удалось сформировать до сбоя.

Каждый элемент выходного массива **dest[]** представляет собой слово, старшие 16 бит которого соответствуют значению параметра **N**, младшие – параметра **M** для физического канала, связанного с данным логическим. Эти значения пользователь может использовать по своему усмотрению для решения каких-либо иных задач, кроме вычисления средней частоты.

***Frequency** – указатель на массив, в который записываются вычисленные значения средней частоты по всем логическим каналам для заданного времени счета. Количество элементов массива может быть от 1 до 16 и равно значению поля **LChQnt** структуры описания модуля. Значения средней частоты располагаются в соответствии с последовательностью каналов в Таблице Логических Каналов.

Приведем пример. Имеем один субмодуль Н-51FH в слоте 3 и определены 2 логических канала, соответствующие 5 и 6 физическим. Пусть значение частоты дискретизации **Fs**=500 кГц, параметра **Base**=5000, время счета **Tc**=20 мс. Соответственно, количество периодов измерения **TbaseQnt**=2, количество запрашиваемых данных - **size**=32***TbaseQnt**=32*2=64. Будем считать, что предварительно создана таблица логических каналов с правильно установленными порогами и состоящая из двух элементов, связанных с физическими каналами 5 и 6. Также будем считать, что выполнены инициализация, открытие и конфигурация модуля, **conf** – экземпляр структуры описания модуля.

Значение времени счета - 1 с **conf.AcqTime**=1000;
Количество запрашиваемых слов данных – **size**=32***conf.TbaseQnt**=32*100=3200. Предполагается, что создание таблицы логических каналов и конфигурирование модуля

произведены ранее. Выполним однократное получение данных. Для простоты в данном примере опускаем обработку ошибок выполнения функций.

DWORD size=32***conf. TbaseQnt**; // 32*2=64 слова
DWORD data[size]; // массив для приема слов данных от модуля
DWORD N_and_M[size]; // выходной массив для вывода параметров N и M для лог. каналов
int to=conf.AcqTime+1000; // Устанавливаем таймаут, равный времени счета + 1с.
double Frequency[4]; // Выходной массив для значений средних частот

LTR51_Start(&conf); // Запуск сбора данных

LTR51_Recv(&conf, data, NULL, size, to); // Запрашиваем требуемое количество данных

// Производим обработку полученных данных и расчет средней частоты
LTR51_ProcessData(&conf, data, N_and_M, Frequency, &size);

LTR51_Stop(&conf); // Останов сбора данных

После работы функции **LTR51_ProcessData()** массив **data[]** будет содержать 2 кадра (64 слова данных, по два для каждого физического канала). Например, массив data[] содержит следующие значения:

Массив src[], принятый от модуля. Кадр включает в себя данные со всех 16 каналов

0x1388000F
0x0000003F
0x1388004E
0x0000007E
0x1388008D
0x000000BD
0x138800CC
0x000000FC
0x1388000B
0x0000003B
0x1388004A
0x0000007A
0x13880089
0x000000B9
0x138800C8
0x000000F8
0x13880007
0x00000037
0x13880046
0x00000076
0x002300085
0x000A00B5
0x002500C4
0x000A00F4
0x13880003

0x00000033
0x13880042
0x00000072
0x13880081
0x000000B1
0x138800C0
0x000000F0
0x1388000F
0x0000003F
0x1388004E
0x0000007E
0x1388008D
0x000000BD
0x138800CC
0x000000FC
0x1388000B
0x0000003B
0x1388004A
0x0000007A
0x13880089
0x000000B9
0x138800C8
0x000000F8
0x13880007
0x00000037
0x13880046
0x00000076
0x00190085
0x000A00B5
0x001700C4
0x000A00F4
0x13880003
0x00000033
0x13880042
0x00000072
0x13880081
0x000000B1
0x138800C0
0x000000F0

Как видно, во всех неподключенных каналах значение M=5000, т.е. **BASE**, а значение N=0. Это говорит о том, что в этих каналах активных перепадов зафиксировано не было (см. *гл. 6.1.*). В физических каналах 5 и 6 значения **N** и **M** определяют частоту поданного сигнала, и для них функцией **LTR51_ProcessData()** будет произведена обработка с формированием выходных массивов **N_and_M** и **Frequency[]**. О формате массива **data[]** см. в Приложении 2.

.....

Выходной массив **N_and_M[]** будет выглядеть следующим образом:

0x000A0025
0x000A0023
0x000A0019
0x000A0013

Если во входном массиве **data[]** два кадра включали в себя 64 слова (в кадре 16 физических каналов, по 2 слова с каждого), то в выходном массиве **N_and_M[]** два кадра включают в себя только 4 элемента, т.к. было определено 2 логических канала.

Массив с рассчитанными значениями средней частоты **Frequency[]** будет включать количество элементов, равное количеству логических каналов, т.е. 2:

Frequency:

997.6057
998.8014

Параметр **size** при получении данных был равен **size**=64; после работы функции **LTR51_ProcessData()** его значение стало **size**=4.

8 Применение модуля для решения задач, отличных от определения средних периода и частоты сигнала

Модуль LTR51 может применяться не только как частотомер, но и для решения иных задач: например, измерение временных интервалов, регистрация отдельных импульсов и подсчет импульсов. Поскольку после работы функции **LTR51_ProcessData()** в массиве **dest[]** для всех логических каналов выдаются значения параметров **M** и **N** по каждому периоду измерения, то пользователь может, используя их и понимая их смысл, решать свои задачи по анализу импульсов или измерению временных интервалов. Решение этого спектра разнообразных задач ложится на пользователя. Можно применять также информацию, содержащуюся во входном массиве «сырых» данных, получаемом от модуля. Подробно формат данных, приходящих от модуля, представлен в Приложении 2.

9 Работа с ППЗУ микроконтроллера AVR.

Микроконтроллер **AVR ATmega 8515**, управляющей работой модуля LTR51, 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: **LTR51_WriteEEPROM()** и **LTR51_ReadEEPROM().**

Для чтения байта из ячейки ППЗУ с указанным адресом используется функция **LTR51_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:

LTR51_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 **LTR51_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:

LTR51_WriteEEPROM(&conf, 150, 0x3F),
где **conf –** экземпляр структуры описания модуля.

Функции записи/чтения в/из ППЗУ целесообразно использовать для хранения информации о положении джамперов для всех физических каналов, поскольку программной возможности считать эту информацию у модуля LTR51 нет. Формат хранения указанной информации в ППЗУ микроконтроллера определяется пользователем самостоятельно.

Приложение 1. Примеры конфигурации и приложения для модуля LTR51.

П1.1 Примеры конфигураций.

Прежде всего следует инициализировать и открыть канал связи с модулем, а также загрузить код прошивки в ПЛИС ALTERA ACEX. Это делается посредством вызова следующих функций: **LTR51_Init()** и **LTR51_Open().** Затем следует заполнить поля структуры описания модуля требуемыми значениями. Ниже приводятся примеры задания конфигурации модуля (определение полей структуры описания модуля):

1. Задействованы все физические каналы модуля. Будем использовать одинаковые значения порогов для всех физических каналов: диапазон +/-1,2 В, верхний порог + 0,7 В, нижний порог – 0.3 В. Режим выделения активных перепадов для всех физических каналов – по спадам. Будем задавать только величину времени счета 1 с. Пользовательские настройки **Fs** и **BASE** использовать не будем.

```
TLTR51 conf_1;          // Объявляем  структуру типа TLTR51
int ThresholdRange;     // Диапазон установки порогов
double HighThreshold;  // Значение верхнего порога
double LowThreshold;  // Значение нижнего порога
double EdgeMode;        // Режим выделения активных перепадов
int i;                  // Используется для счетчика


/* Формируем таблицу логических каналов */
conf_1.LChQnt=16;                    // Количество логических каналов – 16

ThresholdRange=1;    //  Диапазон установки порогов  +/-1,2 В
HighThreshold=0.7;    //  Значение верхнего порога + 0.7 В
LowThreshold= -0.3;   // Значение нижнего порога – 0.3 В
EdgeMode=1;             // Режим выделения активных перепадов – по спадам
for(i=0; i< conf_1.LChQnt; i++)
 {
  conf_1.LChTbl[i]=LTR51_CreateLChannel(i+1, &HighThreshold, &LowThreshold,
                                                ThresholdRange, EdgeMode)
}

/* Устанавливаем параметры сбора данных и вычисления средней частоты */
conf_1.SetUserPars=0;                // Задаем только время счета
conf_1.AcqTime=1000;                 // Интервал измерения устанавливаем равным 1000 мс
```

2. Имеем два субмодуля H-51FL, установленные в слотах 4 и 6 платы-носителя. Соответственно, подключены физические каналы: 7, 8, 11, 12. Диапазон установки порогов для всех каналов: +/-10В. Значения порогов гистерезиса:
   Для физ. канала 7: верхний +5 В, нижний +2 В;
   Для физ. канала 8: верхний +7.2 В, нижний –1,5 В;
   Для физ. канала 11: верхний –1 В, нижний – 7 В;
   Для физ. канала 12: верхний +6,7 В, нижний + 3 В;

Режим выделения активных перепадов для каналов 7 и 11 – по фронтам. Для каналов 8 и 12 – по спадам. Будем использовать пользовательскую настройку Fs, BASE и времени счета. Fs=100 к Гц, BASE=32768, Время счета AcqTime=5 с.

```
TLTR51 conf_2;       // Объявляем  структуру типа TLTR51
int ThresholdRange;     // Диапазон установки порогов
double HighThreshold;  // Значение верхнего порога
double LowThreshold;  // Значение нижнего порога
double EdgeMode;       // Режим выделения активных перепадов
int i;                          // Используется для счетчика

conf_2.size=sizeof(TLTR51);      // размер структуры описания модуля

/* Формируем таблицу логических каналов */
conf_2.LChQnt=4;      // Количество логических каналов – 4

ThresholdRange=0;    //  Диапазон установки порогов - +/-10 В

/* Определяем логический канал 0 */
HighThreshold=5.0;    //  Значение верхнего порога + 0.7 В
LowThreshold= 2.0;    // Значение нижнего порога – 0.3 В
EdgeMode=0;             // Режим выделения активных перепадов – по фронтам

conf_2.LChTbl[0]= LTR51_CreateLChannel(7, &HighThreshold, &LowThreshold,
                                                ThresholdRange, EdgeMode);

/* Определяем логический канал 1 */
HighThreshold=7.2;    //  Значение верхнего порога + 0.7 В
LowThreshold= -1.5;    // Значение нижнего порога – 0.3 В
EdgeMode=1;             // Режим выделения активных перепадов – по спадам

conf_2.LChTbl[0]= LTR51_CreateLChannel(7, &HighThreshold, &LowThreshold,
                                                ThresholdRange, EdgeMode);


/* Определяем логический канал 2 */
HighThreshold=-1;    //  Значение верхнего порога + 0.7 В
LowThreshold= -7;    // Значение нижнего порога – 0.3 В
EdgeMode=0;             // Режим выделения активных перепадов – по фронтам

conf_2.LChTbl[2]= LTR51_CreateLChannel(7, &HighThreshold, &LowThreshold,
                                                ThresholdRange, EdgeMode);


/* Определяем логический канал 3 */
```

```
HighThreshold=6.7;     //  Значение верхнего порога + 0.7 В
LowThreshold=3.0;    // Значение нижнего порога – 0.3 В
EdgeMode=1;              // Режим выделения активных перепадов – по спадам

conf_2.LChTbl[3]= LTR51_CreateLChannel(7, &HighThreshold, &LowThreshold,
                                                    ThresholdRange, EdgeMode);


/* Устанавливаем параметры сбора данных и вычисления средней частоты */
conf_2.SetUserPars=1;                  // Задаем вручную все параметры измерения частоты
conf_2.Fs=100000;                      // Частота дискретизации 100 кГц
conf_2.Base=32768;                     //  BASE=32768 (8000 hex)
conf_2.AcqTime=5000;                   //  Время счета – 5 с
conf_1.AcqTime=1000;                    // Время счета устанавливаем равным 1000 мс
```

П1.2. Пример приложения.

Данный пример представляет собой программу, написанную в среде MSVC++ 2005.

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

#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include "ltr\\include\\ltr51api.h"

void main()
{

TLTR51 conf; // Объявляем экземпляр структуру описания модуля
INT err; // Переменная для хранения кода ошибки
INT SlotNum; // Номер посадочного места
INT i; // Используется для счетчиков итераций в циклах
INT j; // используется для
DWORD samples_qnt; // Количество отсчетов , запрашиваемых из крейт-контроллера
INT samples_qnt_rcv; // Количество отсчетов, полученное из крейт-контроллера
INT ThresholdRange; // Диапазон установки порогов
double HighThreshold; // Значение верхнего порога
double LowThreshold; // Значение нижнего порога
INT EdgeMode; // Режим выделения активных перепадов
int to; // Таймаут для ожидания получения данных
DWORD *data; // Массив, в который запишутся данные, полученные от модуля
DWORD *output_array; //Выходной массив со значениями N и M
double * Frequency; // Выходной массив со значениями средней частоты
CHAR FirmwareVersion[255]; // В эту переменную запишем версию упр. программы AVR
CHAR FirmwareDate[255]; // В эту переменную запишем дату создания упр. программы
AVR
CHAR FPGA_Version[255]; // В эту переменную запишем версию прошивки ПЛИС ALTERA
// ACEX
CHAR SerialNum[255]; // В эту переменную запишем серийный номер модуля
CHAR ModuleName[255]; // В эту переменную запишем имя модуля
CHAR ErrorString[255]; // Строка для вывода описания ошибки
CHAR MsgString[255]; // Строка для вывода сообщений на консоль
/* Инициализируем канал связи с модулем. Поля структуры описания модуля
заполняются значениями по умолчанию. */



err=LTR51_Init(&conf);
```

```
if(err)
  {
      strcpy(ErrorString, (char *) LTR51_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      return;
   }

// Открываем канал связи с модулем. Серийный номер крейта и сетевой адрес - по
умолчанию

// Номер слота - 1;
SlotNum=7;
// Полный путь файла с прошивкой ПЛИС - C://ltr51.ttf
err=LTR51_Open(&conf, SADDR_DEFAULT, SPORT_DEFAULT,"", SlotNum, "C:\\ltr51.ttf");
if(err)
  {
      strcpy(ErrorString, (char *) LTR51_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      LTR51_Close(&conf);
      Sleep(3000);
      return;
   }
/* Заполнение полей для информации. Здесь сделано только для демонстрации */
strcpy(FirmwareVersion, conf.ModuleInfo.FirmwareVersion);
strcpy(MsgString,"Версия прошивки микроконтроллера: ");
strcat(MsgString, FirmwareVersion);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(FPGA_Version, conf.ModuleInfo. FPGA_Version);
strcpy(MsgString,"Версия прошивки ПЛИС: ");
strcat(MsgString, FPGA_Version);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(FirmwareDate, conf.ModuleInfo.FirmwareDate);
strcpy(MsgString,"Дата создания прошивки микроконтроллера: ");
strcat(MsgString, FirmwareDate);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(SerialNum, conf.ModuleInfo.Serial);
strcpy(MsgString,"Серийный номер модуля: ");
strcat(MsgString, SerialNum);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(ModuleName, conf.ModuleInfo.Name);
strcpy(MsgString,"Имя модуля: ");
strcat(MsgString, ModuleName);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

/* Заполняем поля структуры описания модуля */
/* Формируем таблицу логических каналов */
conf.LChQnt=16; // Количество логических каналов - 16
ThresholdRange=1; // Диапазон установки порогов +/-1,2 В
HighThreshold=0.7; // Значение верхнего порога + 0.7 В
LowThreshold=-0.3; // Значение нижнего порога - 0.3 В
EdgeMode=1; // Режим выделения активных перепадов - по спадам
for(i=0; i< conf.LChQnt; i++)
```

```
{
conf.LChTbl[i]=LTR51_CreateLChannel(i+1, &HighThreshold, &LowThreshold,
ThresholdRange, EdgeMode);
}
/* Устанавливаем параметры сбора данных и вычисления средней частоты */

conf.SetUserPars=1; // Параметры устанавливаются пользователем
conf.Fs=100000;     // Частота дискретизации 100 кГц
conf.Base=32768;    // Значение Base
conf.AcqTime=5000;  // Время счета 5 с.



err=LTR51_Config(&conf); // Передача параметров в модуль.
if(err)
  {
      strcpy(ErrorString, (char *) LTR51_GetErrorString(err));
      CharToOem(ErrorString,ErrorString);
      printf("%s",ErrorString);
      LTR51_Close;
      Sleep(3000);
      return;
   }

/* Выведем подкорректированные функцией значения параметров сбора данных */
printf("\n");

sprintf(MsgString,"Точное значение частоты дискретизации: %.3f Гц", conf.Fs);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Значение счетчика BASE: %d", conf.Base);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Точное значение времени счета: %d мс:", conf.AcqTime) ;
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Количество периодов измерения: %d", conf. TbaseQnt);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

to=conf.AcqTime+1000; // Устанавливаем таймаут на секунду больше, чем время счета

/* Выделим память для массива принимаемых данных */
samples_qnt=2*conf.LChQnt*conf. TbaseQnt;
data=(DWORD *) malloc(samples_qnt *sizeof(DWORD));
/* Выделим память для выходного массива */
output_array =(DWORD *) malloc(conf.LChQnt*conf. TbaseQnt *sizeof(DWORD));
/* Выделим память для выходного массива со значениями средней частоты */
Frequency =(double *) malloc(conf.LChQnt*sizeof(double));

// Запустим сбор данных
err=LTR51_Start(&conf); // Старт сбора данных
if(err)
 {
   strcpy(ErrorString, (char *) LTR51_GetErrorString(err));
   CharToOem(ErrorString,ErrorString);
   printf("%s",ErrorString);
   goto exit;
  }
```

```
j=0; // Обнуляем счетчик порций данных

printf("\n");
while(1)
{
/* Определим, сколько элементов должен содержать входной массив данных, чтобы
обеспечить требуемое время счета. Это надо делать при каждой итерации, т.к. ф-я
LTR51_ProcessData() изменит это значение */
samples_qnt=2*conf.LChQnt*conf. TbaseQnt;
/* Получаем отсчеты */
samples_qnt_rcv=LTR51_Recv (&conf, data, NULL, samples_qnt, to);

if(samples_qnt_rcv!=(int) samples_qnt)
{
if(samples_qnt_rcv < 0)
{
 sprintf(MsgString,"%s", "Ошибка при получении данных из крейт-контроллера!");
 CharToOem(MsgString,MsgString);
 printf("%s\n",MsgString);
}
else
{
 sprintf(MsgString, "Ошибка! Количество полученных сэмплов не равно количеству
запрашиваемых!");
 CharToOem(MsgString,MsgString);
 printf("%s\n",MsgString);
}

break; // в случае ошибки выходим из цикла сбора
}

err=LTR51_ProcessData(&conf, data, output_array, Frequency, &samples_qnt);
if(err) // В случае ошибки...
{
strcpy(ErrorString, (char *) LTR51_GetErrorString(err));
CharToOem(ErrorString,ErrorString);
printf( "%s", ErrorString); // Выводим сообщение об ошибке
break; // Выходим из цикла сбора
}
j++;
sprintf(MsgString,"Получена %d-я порция данных",j);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

//Выводим значения частоты
for(i=0;i<conf.LChQnt;i++)
{
 sprintf(MsgString,"Значение частоты для %d-го логического канала: %.3f",i,
Frequency[i]);
 CharToOem(MsgString,MsgString);
 printf("%s\n",MsgString);
}
printf("\n");
if(j==5)
 break;

}
// Окончание цикла сбора
err=LTR51_Stop(&conf); // Остановка сбора данных

exit:
LTR51_Close(&conf); // Закрытие канала связи с модулем
```

```
if(data!=NULL)
free(data);

if(output_array!=NULL)
free(output_array);

if(Frequency!=NULL)
free(Frequency);

sprintf(MsgString,"Сбор данных окончен. Нажмите любую клавишу для выхода.");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);
getchar();
return;

}
```

Приложение 2. Протокол обмена данными с модулем.

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 LTR51.

Все команды от крейт-контроллера к модулю и подтверждения этих команд представляют собой 4-байтные **командные слова**. Данные, содержащие параметры частотного сигнала и передаваемые из модуля в крейт-контроллер, представляют собой 4-байтные **слова данных**. Следует отметить, что указанный протокол является общим для всех модулей данной крейтовой системы.

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

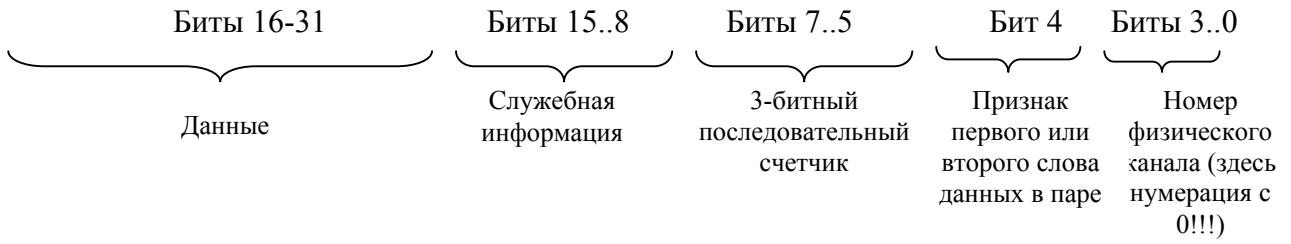
- **Код команды для AVR** – число, определяющее процедуру, которую следует выполнить процессору модуля.
- **Бит четности** – используется для контроля правильности передачи команд в микроконтроллер модуля и в обратном направлении.
- **Код команды RESET, STOP, PROGR или INSTR** – код команды общего интерфейса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

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 при его программировании (но не при сборе данных!!!).

Слова данных используются в рассматриваемом модуле только для передачи данных из модуля в крейт-контроллер и ПК во время сбора данных. При программировании модуля слова данных **не используются**.

Формат **слова данных** имеет следующий вид:



- **Данные** - непосредственно коды, выдаваемые ПЛИС и передаваемые из модуля в крейт-контроллер и ПК.
- **Служебная информация -** информация о номере посадочного места и бит-признак слова данных. Этот байт несет в себе информацию уровня крейт-контроллера. Информация о номере посадочного места не передается из модуля, а добавляется крейт-контроллером.
- **3-битный последовательный счетчик слов данных** – 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 пакетов из модуля. Внимание! Счетчик инкрементируется с каждым новым словом, а не с каждой новой парой слов данных!
- **Признак первого или второго слова данных в паре** – как говорилось ранее и как видно из Таблицы 3, для каждого физического канала модуль отправляет два слова данных. В первом слове содержится значение параметра **M**, во втором – **N**. Первому слову пары, несущему информацию о **M**, соответствует нулевое значение бита <4>. Второму слову пары, несущему информацию о **N**, соответствует единичное значение бита <4>.
- **Номер физического канала** – соответствует номеру физического канала, к которому относится слово данных. Обратите внимание, что здесь нумерация физических каналов начинается с нуля. Поскольку каждому физическому каналу соответствует пара слов данных, то для двух слов, составляющих эту пару, значение этого поля будет одинаковым.

Ниже, в Таблице 3, подробно указан формат слов данных, получаемых от модуля.

Таблица 3

Битовые поля	**<31.. 24>**	**<23..16>**	**<15..12>**	**<11..8>**	**<7..5>**	**<4>**	**<3..0>**
Последовательность пакетов	M <15..0> слот 16		0	№ п.м.	0	0	F
	N<15..0> слот 16		0	№ п.м.	1	1	F
	M <15..0> слот 15		0	№ п.м.	2	0	E
	N<15..0> слот 15		0	№ п.м.	3	1	E
	M <15..0> слот 14		0	№ п.м.	4	0	D
	N<15..0> слот 14		0	№ п.м.	5	1	D
	M <15..0> слот 13		0	№ п.м.	6	0	C
	N<15..0> слот 13		0	№ п.м.	7	1	C
	M <15..0> слот 12		0	№ п.м.	0	0	B
	N<15..0> слот 12		0	№ п.м.	1	1	B

	M <15..0> слот 11	0	№ п.м.	2	0	A
	N<15..0> слот 11	0	№ п.м.	3	1	A
	M <15..0> слот 10	0	№ п.м.	4	0	9
	N<15..0> слот 10	0	№ п.м.	5	1	9
	M <15..0> слот 9	0	№ п.м.	6	0	8
	N<15..0> слот 9	0	№ п.м.	7	1	8
	M <15..0> слот 8	0	№ п.м.	0	0	7
	N<15..0> слот 8	0	№ п.м.	1	1	7
	M <15..0> слот 7	0	№ п.м.	2	0	6
	N<15..0> слот 7	0	№ п.м.	3	1	6
	M <15..0> слот 6	0	№ п.м.	4	0	5
	N<15..0> слот 6	0	№ п.м.	5	1	5
	M <15..0> слот 5	0	№ п.м.	6	0	4
	N<15..0> слот 5	0	№ п.м.	7	1	4
	M <15..0> слот 4	0	№ п.м.	0	0	3
	N<15..0> слот 4	0	№ п.м.	1	1	3
	M <15..0> слот 3	0	№ п.м.	2	0	2
	N<15..0> слот 3	0	№ п.м.	3	1	2
	M <15..0> слот 2	0	№ п.м.	4	0	1
	N<15..0> слот 2	0	№ п.м.	5	1	1
	M <15..0> слот 1	0	№ п.м.	6	0	0
	N<15..0> слот 1	0	№ п.м.	7	1	0
	M <15..0> слот 16	0	№ п.м.	0	0	F
	N<15..0> слот 16	0	№ п.м.	1	1	F
. . . . .						