

L-CARD

Крейтовая система LTR

LTR-212

Модуль для тензометрических измерений

Руководство программиста



Москва. Ноябрь 2013 г.
Ревизия документа 1.0.4

ООО «Л КАРД»,

117105, г. Москва, Варшавское шоссе, д. 5, корп. 4, стр. 2.

тел. (495) 785-95-25

факс (495) 785-95-14

Адреса в Интернет:

WWW: www.lcard.ru

FTP: [ftp.lcard.ru](ftp://ftp.lcard.ru)

E-Mail:

Общие вопросы: lcard@lcard.ru
Отдел продаж: sale@lcard.ru
Техническая поддержка: support@lcard.ru
Отдел кадров: job@lcard.ru

Представители в регионах:

Украина:	HOLIT Data Systems	www.holit.com.ua	380 (44) 241-67-54
Санкт-Петербург:	Автэкс-СПБ (ООО "ЭМС")	www.autex.spb.ru	(812) 567-72-02
Санкт-Петербург:	"Ниеншанц-Автоматика"	www.nnz-ipc.ru	(812) 326-59-24
Новосибирск:	ООО "Сектор-Т"	www.sector-t.ru	(3832) 22-76-20
Екатеринбург:	ООО "Авеон"	www.aveon.ru	(343) 381-75-75
Казань:	ООО "Шатл"	shuttle@kai.ru	(8432) 38-16-00
Пенза:	ООО "Ньютон"	www.nwtm.ru/industry	(846)-998-29-01

LTR-212. Специализированный модуль для тензометрических измерений.

© Copyright 1989–2013, **ООО «Л Кард»**. Все права защищены.

История ревизий настоящего документа.

Ревизия	Дата	Примечания по внесенным изменениям
1.0.0	23.01.2006	Первая доступная для пользователя ревизия
1.0.1	07.03.2006	Изменена структура описания модуля
1.0.2	21.11.2006	Произведены изменения в формате структуры описания модуля и именах полей в целях унификации программного интерфейса всех модулей
1.0.3	1.03.2010	Добавлен новый режим калибровки. Добавлены функции сохранения и записи пользовательской калибровочной области ППЗУ.
1.0.4	1.09.2013	Добавлена информация о новых модификациях модуля. Изменена структура описания модуля (добавлено поле <code>Туре</code> структуры <code>TINFO_LTR212</code>). Изменён формат логического канала (добавлено поле ' <i>Тип мостового подключения</i> '). Введена новая функция <code>LTR212_CreateLChannel2()</code> .

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.

ООО "А-Кард" оставляет за собой право обновлять документацию без уведомления пользователей об изменениях.

Оглавление

1	Основные сведения о модуле LTR-212.	5
1.1	Что нового?	5
1.1.1	2013 год	5
2	Общие сведения о библиотеке LR212API.	6
2.1	Структура описания модуля.	6
2.2	Функции пользовательской библиотеки.	6
2.2.1	Классификация функций библиотеки.	6
2.2.2	Типичная последовательность написания программы.	8
3	Подробное описание библиотеки LTR212API.	9
3.1	Структура описания модуля.	9
3.1.1	Идентификационная информация модуля	11
3.1.2	Структура для хранения информации о пользовательской калибровке	12
3.2	Таблица логических каналов.	12
3.3	Описание функций библиотеки Ltr212api.	15
4	Особенности выполнения процесса сбора данных.	22
4.1	Формирование цикла сбора и получения данных	22
4.2	Особенности режимов сбора данных.	25
5	Калибровка модуля.	26
5.1	Режимы калибровки.	26
5.2	Особенности использования различных режимов калибровки.	28
6	Цифровая фильтрация.	30
6.1	Цифровые фильтры, используемые в модуле LTR212.	30
6.2	Применение программных фильтров	30
7	Проверка ППЗУ.	31
	Приложение 1. Примеры написания программ.	32

1 Основные сведения о модуле LTR-212.

Модуль для тензометрических измерений *LTR-212* имеет три основных режима работы: **4-канальный средней точности**, **4-канальный высокой точности** и **8-канальный высокой точности**. При этом дополнительно может быть установлена величина опорного напряжения (2,5 В или 5 В) и выбран тип опорного напряжения: *постоянное* или *знакопеременное*. При работе на каждом из режимов выполняется цифровая фильтрация полученных данных. Модуль позволяет осуществлять программную калибровку нуля и диапазона, компенсируя при этом ошибки смещения и усиления подключаемых сигналов. При калибровке нуля задействованы калибровочные регистры АЦП AD7730 и встроенный в эту микросхему тарировочный ЦАП. При калибровке диапазона используются только калибровочные регистры АЦП. Управление режимами сбора данных и калибровки, программирование АЦП AD7730, чтение полученных данных из регистров АЦП и отправка их в FIFO-буфер крейт-контроллера осуществляется Цифровым Сигнальным Процессором ADSP-2185M, который установлен на модуле. Этот процессор имеет свое собственное программное обеспечение низкого уровня, именуемое в данном описании *БИОС*.

1.1 Что нового?

Как правило, в данном параграфе будут приводиться только самые основные сведения об изменениях в модуле. За более подробной информацией следует обращаться к 'Главе 6' в книге "[Крейтовая система LTR. Руководство пользователя](#)"

1.1.1 2013 год

Компания ООО "А-Кард" (по результатам обобщения пожеланий пользователей за всю историю выпуска модулей *LC-212* и *LTR-212*) в 2013 г. осуществила выпуск новых модификаций модуля: *LTR-212M-1*, *LTR-212M-2* и *LTR-212M-3*.

При описании далее вводятся следующие сокращённые обозначения:

- *LTR-212* – старая модификация, выпускаемая до 2013 года;
- *LTR-212M* – все новые модификации *LTR212M-1*, *LTR212M-2* и *LTR212M-3*;
- *LTR-212(M)* – все модификации: *LTR212*, *LTR212M-1*, *LTR212M-2* и *LTR212M-3*.

2 Общие сведения о библиотеке LR212API.

С точки зрения программного обеспечения всё взаимодействие с модулем осуществляется при помощи библиотеки пользовательских функций *Ltr212api*.

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 режима сбора данных по умолчанию;
- Загрузка *БИОС* (низкоуровневого Программного Обеспечения) во внутреннюю память Цифрового Сигнального Процессора ADSP-2185M;
- Установка параметров работы (программирование каналов сбора данных, определение режима сбора данных или калибровки);
- Старт сбора данных;
- Получение и обработка данных (чаще всего этот процесс циклический);
- Остановка сбора данных;
- Закрытие интерфейсного канала связи с модулем.

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 **TLTR212**) предназначена для инициализации, хранения и изменения данных о конфигурации модуля в рамках создаваемой программы. При помощи полей этой структуры задается режим сбора данных, определяется использование калибровочных коэффициентов, формируется информация о задействованных каналах сбора данных и соответствующих им диапазонах напряжения входного сигнала. Эта структура также включает в себя подструктуру, содержащую информацию об используемых программных фильтрах. Перед вызовом функции конфигурации модуля (**LTR212_SetADC()**) следует обязательно заполнить поля структуры описания модуля. В противном случае модуль может выдавать неверные данные.

2.2 Функции пользовательской библиотеки.

Функции библиотеки пользователя *Ltr212api* предназначены для конфигурирования, программирования, сбора данных, калибровки и диагностики модуля.

2.2.1 Классификация функций библиотеки.

Функции, входящие в состав пользовательской библиотеки модуля *LTR-212(M)*, можно разделить на следующие группы:

- Функции инициализации и открытия;
- Функции конфигурирования;
- Функции сбора данных;
- Функция калибровки;
- Функция закрытия;
- Вспомогательные функции.

К **функциям инициализации и открытия** относятся функции **LTR212_Init()** и **LTR212_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, загрузить *БИОС* в DSP модуля и выполнить необходимые проверки. Без вызова этих функций дальнейшая работа с модулем невозможна.

Функции конфигурирования – это **LTR212_CreateLChannel()**, **LTR212_CreateLChannel2()** и **LTR212_SetADC()**. При помощи функции **LTR212_CreateLChannel()** выполняется создание таблицы логических каналов, т.е. связывание физических каналов модуля и соответствующих им диапазонов напряжения входного сигнала, а также определение, какие физические каналы будут задействованы при сборе и получении данных, а какие – нет. Также можно воспользоваться функ-

цией **LTR212_CreateLChannel2()**, которая дополнительно может устанавливать и тип мостовой схемы подключения датчиков (для *LTR-212M-1*). Функция **LTR212_SetADC()** передает все конфигурационные параметры, записанные в полях структуры описания модуля, в DSP модуля, который в свою очередь выполняет программирование АЦП AD7730, установленных на плате. После выполнения указанных функций устройство готово к сбору данных.

Функции сбора данных: LTR212_Start(), LTR212_Stop(), LTR212_ProcessData(). Предназначены для старта и остановки сбора данных, а также для проверки правильности полученных из модуля данных и пересчета их из кодов АЦП в значения напряжения (В).

Функция калибровки: LTR212_Calibrate(). Используется для выполнения различных калибровок модуля. Аргументы данной функции определяют режим калибровки и каналы, для которых следует ее произвести.

Вспомогательные функции: LTR212_GetErrorString() возвращает строку ошибки, соответствующую коду ошибки; **LTR212_CalcFS()** возвращает частоту сбора данных; **LTR212_TestEEPROM()** выполняет проверку целостности информации в ППЗУ модуля.

Функция закрытия: LTR212_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.

При программировании модуля следует придерживаться указанной ниже схемы.

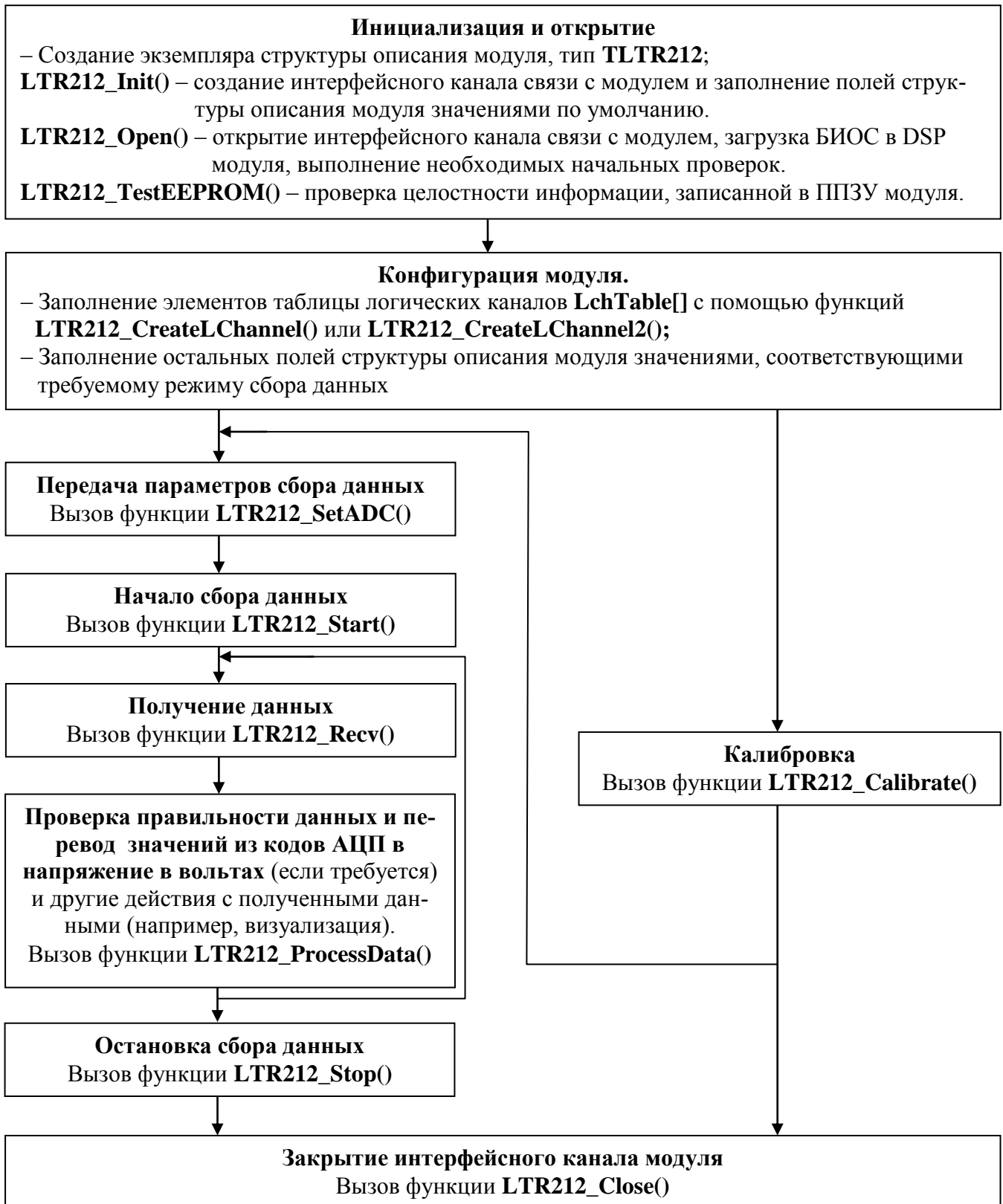


Рис. 1

3 Подробное описание библиотеки LTR212API.

В этом параграфе будут достаточно подробно рассмотрены компоненты пользовательской библиотеки *Ltr212api*.

Для получения возможности вызова интерфейсных функций библиотеки *Ltr212api* 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 **ltr212api.dll**.
- добавить в проект информацию о способе вызова интерфейсных функций dll-библиотеки и используемых типах данных. В различных средах разработки последовательность действий и приложенные усилия могут несколько отличаться:

Borland C++/Borland C++ Builder :

- подключить к проекту файлы **LTR\LIB\BORLAND\ltr212api.lib** и **LTR\INCLUDE\ltr212api.h**;

Microsoft Visual C++ :

- подключить к проекту файлы **LTR\LIB\MSVC\ltr212api.lib** и **LTR\INCLUDE\ltr212api.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 задействованных каналах модуля, режимах сбора данных или калибровки, об использовании калибровочных коэффициентов, подключении программных фильтров и некоторые другие сведения. Определение структуры приводится ниже:

```
typedef struct
{
    INT size; // Размер структуры
    TLTR channel; // Интерфейсный канал связи с модулем
    INT AcqMode; // Режим сбора данных
    INT UseClb; // Флаг использования калибровочных коэффициентов
    INT UseFabricClb; // Флаг использования заводских калибровочных коэффициентов
    INT LChQnt; // Количество логических каналов
    INT LChTbl[8]; // Таблица логических каналов
    INT REF; // Флаг выбора опорного напряжения (0->2.5В, 1->5В)
    INT AC; // Флаг использования знакопеременного опорного напряжения
    double Fs; // Частота выдачи данных

    struct
    {
        INT IIR; // Флаг использования программного БИХ-фильтра
        INT FIR; // Флаг использования программного КИХ-фильтра
        BYTE Decimation; // Коэффициент децимации программного КИХ-фильтра
        BYTE TAP; // Порядок (кол-во отводов) программного КИХ-фильтра
        CHAR IIR_Name[512+1]; // Полный путь программного БИХ-фильтра
        CHAR FIR_Name[512+1]; // Полный путь программного КИХ-фильтра
    } filter; // Структура, хранящая данные о программных фильтрах

    TINFO_LTR212 ModuleInfo // Идентификационная информация модуля
};
```

```

WORD CRC_PM; // для внутреннего использования
WORD CRC_Flash_Eval; // для внутреннего использования
WORD CRC_Flash_Read; // для внутреннего использования
} TLTR212, *PTLTR212; // структура описания модуля

```

Далее приводится описание полей этой структуры.

Таблица 1

Поле		Тип	Описание
size		INT	объем памяти, выделяемый под структуру
Channel		TLTR	структура, представляющая собой описание интерфейсного канала связи с крейт-контроллером;
AcqMode		INT	Определяет режим сбора данных. «0» - 4-канальный средней точности, «1» - 4-канальный высокой точности, «2» - 8-канальный высокой точности.
UseClb		INT	Определяет, следует ли использовать калибровочные коэффициенты, хранящиеся в калибровочной области ППЗУ модуля, при сборе данных. «0» - калибровочные коэффициенты не используются. «1» - калибровочные коэффициенты используются.
UseFabricClb		INT	Если установлен этот флаг, то при вызове функции загрузки конфигурации LTR212_SetADC() автоматически будут загружены в регистры АЦП AD7730 заводские калибровочные коэффициенты, соответствующие диапазонам каждого из задействованных каналов
LChQnt		INT	Количество используемых логических каналов. Разъяснение понятия “логический канал” см. ниже, в гл. 3.2 .
LChTbl[8]		INT	Таблица логических каналов. Разъяснение понятия “таблица логических каналов” см. ниже, в гл. 3.2 .
filter			Структура параметров программных фильтров
поля	IIR	INT	Определяет, задействован ли программный БИХ-фильтр “0” – фильтр отключен, “1” – фильтр включен
	FIR	INT	Определяет, задействован ли программный КИХ-фильтр. “0” – фильтр отключен, “1” – фильтр включен
	Decimation	BYTE	Коэффициент децимации, применяемый с используемым программным КИХ-фильтром. Этот коэффициент жестко привязан к конкретному фильтру.
	TAP	BYTE	Порядок (количество отводов) используемого программного фильтра
	IIR_Name[512+1]	CHAR	Строка, представляющая собой полное имя файла с коэффициентами программного БИХ-фильтра
	FIR_Name[512+1]	CHAR	Строка, представляющая собой полное имя файла с коэффициентами программного КИХ-фильтра

ModuleInfo	TINFO_LTR212	Идентификационная информация модуля. Данный тип представляет собой структуру, описание которой приводится ниже в этой главе
REF	INT	Определяет значение опорного напряжения: «0» - 2,5 В, «1» - 5.0 В.
AC	INT	Флаг использования знакопеременного опорного напряжения. «1» – используется знакопеременное опорное напряжение, «0» – используется постоянное опорное напряжение
Fs	INT	Частота выдачи данных в герцах. Это поле заполняется автоматически после вызова функции LTR212_SetADC()

Для задания режима работы модуля пользователю необходимо самостоятельно (вручную) определить следующие поля структуры: **AcqMode**, **UseClb**, **UseFabricClb**, **LChQnt**, **LChTbl**, **filter.FIR**, **filter.IIR**, **filter.IIR_Name**, **filter.FIR_Name**, **REF**, **AC**. Остальные поля заполняются автоматически при вызове различных функций. В конце данного руководства приводятся примеры задания конфигурации.

3.1.1 Идентификационная информация модуля

Данная структура содержит, после выполнения функции **LTR212_Open()**, всё необходимую служебную информацию об используемом модуле:

```
typedef struct
{
    CHAR Name[15]; // строка с именем модуля
    BYTE Type; // тип модификации модуля: LTR-212 (LTR-212M-3),
                // LTR-212M-1 или LTR-212M-2
    CHAR Serial[24]; // строка с серийным номером модуля
    CHAR BiosVersion[8]; // строка с версией БИОС'а
    CHAR BiosDate[16]; // строка с датой создания БИОС'а
} TINFO_LTR212, *PTINFO_LTR212;
```

Поле Type содержит текущий тип модификации используемого модуля:

Таблица 2

Модификация модуля	Тип	Мнемоника
LTR212 или LTR212M-3	0	LTR212_OLD
LTR212M-1	1	LTR212_M_1
LTR212M-2	2	LTR212_M_2

3.1.2 Структура для хранения информации о пользовательской калибровке

```
typedef struct
{
    DWORD Offset[MAX_212_CH]; // Коэффициенты смещения для 8-ми каналов
    DWORD Scale[MAX_212_CH]; // Коэффициенты масштаба для 8-ми каналов
    BYTE DAC_Value[MAX_212_CH]; // Значение тарировочного ЦАП кодеков
} TLTR212_Usr_Clb;
```

Калибровочная информация в этой структуре храниться во внутреннем формате кодека AD7730. Прямое изменение её нежелательно, без ознакомления с особенностями работы кодека.

3.2 Таблица логических каналов.

Одно из полей структуры описания модуля – **таблица логических каналов** – представляет собой массив, содержащий информацию о физических каналах модуля, соответствующих им диапазонах сбора данных и мостовых схемах подключения, а также определяет последовательность опроса каналов.

Логический канал представляет собой 32^x битное слово, формат которого представлен ниже:

Таблица 3

Биты	31÷28	27÷20	19÷16	15÷4	3÷0
Назначение	Тип мостового подключения	Зарезервировано	Номер физического канала	Зарезервировано	Диапазон входного напряжения

Тип мостового подключения задает мостовую схему подсоединения датчиков к модулю.

Таблица 4

Тип мостового подключения	Мнемоника	Значение
0	LTR212_FULL_OR_HALF_BRIDGE	Полно- или полу-мостовая схема подключения. Для модулей LTR-212(M)
1	LTR212_QUARTER_BRIDGE_WITH_200_Ohm	Четверть-мостовая схема подключения с использованием внутреннего резистора 200 Ом. Только для модулей LTR-212M-1.
2	LTR212_QUARTER_BRIDGE_WITH_350_Ohm	Четверть-мостовая схема подключения с использованием внутреннего резистора 350 Ом. Только для модулей LTR-212M-1.
3	LTR212_QUARTER_BRIDGE_WITH_CUSTOM_Ohm	Четверть-мостовая схема подключения с использованием внешнего пользовательского резистора 180÷1000 Ом на мезонине LTR212H. Только для модулей LTR-212M-1.

4	LTR212_UNBALANCED_QUARTER_BRIDGE_WITH_200_Ohm	Аналогично подключению 1, только с внесением нормируемого разбаланса четвертьмостов . Только для модулей <i>LTR-212M-1</i> .
5	LTR212_UNBALANCED_QUARTER_BRIDGE_WITH_350_Ohm	Аналогично подключению 2, только с внесением нормируемого разбаланса четвертьмостов . Только для модулей <i>LTR-212M-1</i> .
6	LTR212_UNBALANCED_QUARTER_BRIDGE_WITH_CUSTOM_Ohm	Аналогично подключению 3, только с внесением нормируемого разбаланса четвертьмостов . Только для модулей <i>LTR-212M-1</i> .

Следует подчеркнуть, что четверть-мостовую схему подключения датчиков можно использовать только на модуле *LTR-212M-1* и только на первых 4^x **физических каналах**. При этом во всех **логических каналах**, где применяются четвертьмосты, должны быть использованы одинаковые типы мостовых подключений. В этом смысле типы подключений 1 и 4 можно считать совместимыми, также как 2 и 5 или 3 и 6.

Физическим каналом называется аппаратный канал ввода данных, используемый для подключения электрического сигнала к модулю *LTR-212(M)*. Модуль имеет 8 физических каналов. Полная информация о них указана в книге *«Крейтовая система LTR. Руководство пользователя»*.

Коды диапазонов входного напряжения представлены ниже:

Таблица 5

Код входного диапазона	Диапазон
0	-10 mV/+10mV
1	-20 mV/+20mV
2	-40 mV/+40mV
3	-80 mV/+80mV
4	0mV/+10mV
5	0 mV/+20mV
6	0 mV/+40mV
7	0 mV/+80mV

Таблицей логических каналов называется массив, каждый элемент которого представляет собой логический канал. Все элементы этой таблицы определяют те физические каналы, с которых будут передаваться данные после старта, и соответствующие им диапазоны напряжений входных сигналов. Заполнение этого массива (таблицы логических каналов) осуществляется пользователем. Для создания логического канала следует вызвать функцию **LTR212_CreateLChannel()**, где в параметрах нужно указать номер физического канала, связываемого с данным логическим, и код соответствующего ему диапазона. Функция возвращает сформированное значение логического канала, которое следует записать в таблицу вручную. В таблице логических каналов элементы, соответствующие различным физическим каналам модуля, должна располагаться **последовательно**, в порядке возрастания номеров физических каналов. Элементы с более старшими индексами должны представлять собой элементы, соответствующие физическим каналам с более старшими номерами. Если это условие не выполнено, последующий вызов функции **LRT212_SetADC()** завершится с ошибкой. При работе модуля данные будут приходить только с тех каналов, которые

включены в рассматриваемую таблицу. Данные с других каналов будут игнорироваться. Поле структуры описания модуля **LChQnt** должно содержать число задействованных логических каналов (оно же – число элементов таблицы логических каналов). В четырехканальных режимах работы (**AcqMode=0** и **AcqMode=1**) максимальное число логических каналов равно 4. В восьмиканальном режиме (**AcqMode=2**) равно 8.

При работе модуля данные с каналов будут поступать в последовательности, определяемой таблицей логических каналов.

Пример верной конфигурации таблицы логических каналов в 8-канальном режиме:

LChQnt=6;

LChTbl[]:

Индекс элемента (номер логический канала)	Физический канал	Диапазон	Значение логического канала (Hex)
0	1	0 (-10 mV/+10mV)	0x00010000
1	2	0 (-10 mV/+10mV)	0x00020000
2	4	3 (-80 mV/+80mV)	0x00040003
3	5	6 (0 mV...+40mV)	0x00050006
4	7	4 (0 mV...+10mV)	0x00070004
5	8	3 (-80 mV/+80mV)	0x00080003

Т.е. массив **LChTbl** будет следующим:

0x00010000
0x00020000
0x00040003
0x00050006
0x00070004
0x00080003

Видно, что физические каналы 3 и 6 не включены в таблицу. Следовательно, никаких данных с этих каналов не будет содержаться в массиве, полученном от модуля при сборе данных. Эти каналы можно считать отключенными. Для текущего примера порядок опроса физических каналов и расположения данных во входном массиве при получении данных приведен ниже, цифры здесь указывают номер физического канала, с которого получен каждый элемент входного массива (пакет данных):

1 – 2 – 4 – 5 – 7 – 8 – 1 – 2 – 4 – 5 – 7 – 8 – 1 – 2 – 4 – 5

└──────────┘ └──────────┘ └──────────┘

Более подробно о входном массиве и о расположении данных в нем будет рассказано ниже, в [гл. 4.1](#).

3.3 Описание функций библиотеки Ltr212api.

Формат: INT LTR212_Init(PTLTR212 hnd)

Параметр: hnd – указатель на структуру описания модуля (тип PTLTR212)

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

```
hnd->size=sizeof(TLTR212);           // Размер структуры
hnd->AcqMode=1;                       // 4-канальный режим сбора данных высокой точности
hnd->UseClb=0;                         // Не задействуем тонирующие коэффициенты
hnd->UseFabricClb=0;                  // Не задействуем режим прямого использования
                                        // заводских калибровочных коэффициентов
hnd->LChQnt=4;                        // Количество логических каналов равно 4

// Инициализация таблицы логических каналов. По умолчанию задаются четыре канала
// с диапазоном напряжения входного сигнала –80 мВ/+80 мВ
hnd->LChTbl[0]=LTR212_CreateLChannel(1, 3);
hnd->LChTbl[1]=LTR212_CreateLChannel(2, 3);
hnd->LChTbl[2]=LTR212_CreateLChannel(3, 3);
hnd->LChTbl[3]=LTR212_CreateLChannel(4, 3);

hnd->filter.IIR=0;                    // Программные фильтры отключены
hnd->filter.FIR=0;
hnd->filter.Decimation=0;            // отключим децимацию фильтра
hnd->filter.TAP=0;                   // Порядок фильтра – 0
hnd->Fs=150.15;                      // Частота сбора для данного режима

// До вызова функции LTR212_Open() следующие поля являются пустыми строками
strcpy((CHAR *)hnd->ModuleInfo.Serial, "\0"); // Серийный номер по умолчанию
strcpy((CHAR *)hnd->ModuleInfo.BiosVersion, "\0"); // Версия БИОСа по умолчанию
strcpy((CHAR *)hnd->ModuleInfo.BiosDate, "\0"); // Дата БИОСа по умолчанию
strcpy((CHAR *)hnd->ModuleInfo.Name, "\0"); // Имя модуля по умолчанию
```

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

Формат: INT LTR212_IsOpened(PTLTR212 hnd)

Параметр: **hnd** – указатель на структуру описания модуля, тип TLTR212

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 LTR212_Open(PTLTR212 hnd, DWORD net_addr, WORD net_port, CHAR *crate_sn, INT slot_num, CHAR *biosname)**Параметры:**

- **hnd** – указатель на структуру описания модуля (тип **PTLTR212**)
- **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.
- **slot_num** – номер слота, в котором расположен модуль (нумерация *с единицы!*)
- ***biosname** – полный путь к файлу, содержащему программу для DSP (*БИОС*). Поставляемый фирмой файл *БИОС*'а имеет имя **ltr212.bio**. В этой функции необходимо указать полный путь к этому файлу в файловой системе пользователя. Следует помнить, что, согласно синтаксису языка “C”, Символ ‘\’ в строках должен быть заменен на “\\”.

Описание: Функция открывает интерфейсный канал связи с модулем, считывает из файла и загружает в DSP модуля программу *БИОС*, выполняет необходимые проверки, а также считывает из ППЗУ модуля его идентификационную запись. После работы функции в соответствующих полях структуры описания модуля будут находиться: версия *БИОС*'а, дата создания *БИОС*'а, имя модуля, тип модуля и серийный номер модуля.

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

Формат: INT LTR212_CreateLChannel(INT PhysChannel, INT Scale)

Параметры:

- **PhysChannel** – номер физического канала, соответствующего создаваемому логическому каналу. Нумерация начинается с единицы;
- **Scale** – диапазон напряжения входного сигнала для данного физического канала.

Описание: Функция создает логический канал, т.е. формирует 32^x битное слово, которое следует записать в соответствующую ячейку таблицы логических каналов. Подробнее о формате слова см. [гл. 3.2](#) этого руководства.

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

Формат: INT LTR212_CreateLChannel2(INT PhysChannel, INT Scale, INT BridgeType)

Параметры:

- **PhysChannel** – номер физического канала, соответствующего создаваемому логическому каналу. Нумерация начинается с единицы;
- **Scale** – диапазон напряжения входного сигнала для данного физического канала;
- **BridgeType** – тип мостового подключения.

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. [гл. 3.2](#) этого руководства.

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

Формат: INT LTR212_SetADC(PTLTR212 hnd)

Параметр: hnd – указатель на структуру описания модуля **PTLTR212**

Описание: функция передачи модулю информации о каналах и сведений о режиме сбора данных. Информация об используемых физических каналах, диапазонах, режиме фильтрации загружается в память DSP и в регистры АЦП AD7730, установленных на плате модуля. Если применяются калибровочные коэффициенты, функция загружает их в калибровочные регистры АЦП AD7730, соответствующие нужным каналам. При использовании программных фильтров их коэффициенты загружаются во внутреннюю память DSP модуля. Перед использованием этой функции все поля структуры описания модуля должны быть заполнены требуемыми значениями. Запуск сбора данных следует осуществлять *только после выполнения этой функции*.

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

Формат: INT LTR212_Start(PTLTR212 hnd)

Параметр: hnd – указатель на структуру описания модуля PTLTR212

Описание: Выполняет запуск сбора данных. Перед вызовом этой функции необходимо выполнить функцию LTR212_SetADC().

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

Формат: INT LTR212_Stop(PTLTR212 hnd)

Параметр: hnd – указатель на структуру описания модуля PTLTR212

Описание: Выполняет остановку сбора данных.

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

Формат: INT LTR212_Recv(PTLTR212 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 в 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 завершится. 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 LTR212_GetErrorString().

Примечание: Описание этой функции соответствует описанию функции LTR_Recv() библиотеки драйвера-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 LTR212_ProcessData(PTLTR212 hnd, DWORD *src, double *dest, DWORD *size, BOOL volt)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**
- ***src** – указатель на массив с данными, полученными из модуля предварительно вызванной функцией **LTRRcv()**.
- ***dest** – указатель на массив, в который записываются выходные данные, формируемые рассматриваемой функцией.
- **volt** – флаг перевода полученных значений кода АЦП в значение напряжения (В).
- ***size** – указатель на переменную, представляющую размер массива данных

Описание: Функция выполняет проверку полученных из модуля данных и трансформацию их из внутреннего формата системы LTR в непосредственно код, полученный от АЦП. Также эта функция способна выполнить перевод полученного кода в значение напряжения в вольтах в соответствии с диапазонами, выбранными для используемых каналов.

src[] – массив с данными, полученными из модуля предварительно вызванной функцией **LTR_Rcv()**. Функция сравнивает последовательность каналов, определенную при создании таблицы логических каналов, с последовательностью, полученной непосредственно от модуля. В случае несоответствия функция пропускает кадр, где это несоответствие было обнаружено, и на его место в выходном массиве **dest[]** записывает следующий кадр, если он верен. «Сбойный» кадр стирается целиком, независимо от того, в каком месте кадра произошло нарушение нужной последовательности каналов. Однако при этом **функция возвращает ошибку** несмотря на то, что массив подкорректирован и дальнейшая работа, в принципе, возможна. Функция также отслеживает состояние счетчика отправленных пакетов данных. В случае его сбоя функция не прерывает свое выполнение, выходной массив формируется по указанным выше правилам, но возвращает ошибку. В данную функцию в качестве параметра **src** можно передавать массив любой длины и заказывать обработку любого количества точек (определяется параметром **size**) но следует помнить, что количество обрабатываемых точек должно быть кратно количеству используемых каналов. В противном случае функция завершится с ошибкой.

dest[] – массив, в который записываются выходные данные, формируемые рассматриваемой функцией. Функция при этом удаляет все служебные поля из элементов входного массива. При значении “0” параметра **volt** выходной массив представляет собой последовательность кодов, полученных от АЦП модуля, которые следуют по кадрам, т.е. в порядке, соответствующем таблице логических каналов. При ненулевом значении параметра **volt** выходной массив будет содержать значения сигнала в вольтах, вычисленные в соответствии с диапазонами каналов, заданными в таблице логических каналов. Необходимо отметить, что в обоих случаях выходной массив **dest[]** имеет тип **double**.

size – На входе функции это значение соответствует количеству элементов входного массива **src[]**. Если в процессе проверки данных рассматриваемой функцией не было выявлено нарушений последовательности отсчетов, то после завершения работы функции значение параметра **size** уменьшится вдвое (т.к. в исходном массиве каждый сэмпл определялся двумя соседними элементами). Если были обнаружены и удалены сбойные кадры, то значение **size** будет меньше, т.к. количество элементов выходного массива сократится за счет удаления сбойных кадров. Параметр **size** в этом случае не будет совпадать с половиной размера входного массива. Подробнее о применении этой функции см. ниже, в [гл. 4.1](#).

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

Формат: INT LTR212_Calibrate(PTLTR212 hnd, BYTE *Lchannel_Mask, INT mode, INT reset)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**;
- ***LChannel_Mask** – указатель на маску логических каналов, подлежащих калибровке. Представляет собой байт, где номера единичных разрядов соответствуют номерам логических каналов, подлежащих калибровке. Например, **LChannel_Mask=0x93** (10010011 в двоичной системе) указывает, что калибровке подлежат 0, 1, 4, 7 логические каналы. Калибровка этих каналов выполняется одновременно. По возвращении из функции при успешной калибровке всех каналов значение маски не изменяется. Этот параметр может измениться только при возникновении ошибок внешней калибровки нуля в случае, когда не удалось подобрать значение встроенного ЦАП. Тогда биты маски, соответствующие некалиброванным каналам, остаются единицами. Биты, соответствующие, калиброванным каналам, равны 0. Таким образом, при ошибочном завершении функции можно увидеть, какие каналы не удалось откалибровать. Следует отметить, что при первой же ошибке калибровки функция завершается, и остальные каналы не калибруются. В случае успешной калибровки всех каналов значение маски не изменяется.
- **mode** – режим калибровки;
- **reset** – флаг сброса всех АЦП AD7730 перед калибровкой. “1» - все АЦП сбрасываются, “0” - не сбрасываются.

Описание: Выполняет калибровку указанных логических каналов с последующей записью калибровочных коэффициентов в ППЗУ модуля. Подробнее о режимах калибровки см. ниже, в главе 5, «*Особенности калибровки модуля*». Перед калибровкой необходимо определить маску логических каналов, подлежащих калибровке.

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

Формат: LPCSTR LTR212_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

INT LTR212_CalcFS(PTLTR212 hnd, double *fsBase, double *fs)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**;
- ***fsBase** – указатель на значение частоты дискретизации АЦП при сборе данных;
- ***fs** – указатель на значение частоты выдачи данных

Описание: Возвращает частоту выдачи данных. При использовании 4-канального режима средней точности без программных фильтров и 4-канального режима высокой точности данная частота совпадает с частотой дискретизации АЦП. При использовании программного КИХ-фильтра **fsBase** – частота дискретизации АЦП, а **fs** меньше **fsBase** в число раз, равное коэффициенту **десимации** ($fs=fsBase/hnd->filter.Decimation$). В восьмиканальном режиме **fsBase** – это частота дискретизации АЦП при заполнении отводов аппаратных цифровых фильтров, **fsBase=150,15** Гц, **fs** – частота выдачи данных, равная примерно 3,4 Гц.

Возвращаемое значение: -----

INT LTR212_CalcTimeout(PTLTR212 hnd, DWORD n)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**;
- **n** – количество точек на канал.

Описание: Возвращает максимальное время в миллисекундах, в течение которого функция **LTR212_Recv()** должна ожидать получения порции данных, если *количество точек на канал* равно **n**. Функция рассчитывает таймаут исходя из режима, установленного в структуре описания модуля.

Возвращаемое значение: максимальное время в миллисекундах, в течение которого функция **LTR212_Recv()** должна ожидать получения порции данных.

Формат: INT LTR212_TestEEPROM(PTLTR212 hnd)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**

Описание: Функция выполняет контроль целостности данных, записанных в ППЗУ модуля.

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

Формат: INT LTR212_ReadUSR_Clb (PTLTR212 hnd, TLTR212_Usr_Clb *CALLIBR)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**;
- **CALLIBR** – указатель на структуру для сохранения пользовательских калибровок

Описание: Функция, считывающая пользовательские калибровки из ППЗУ модуля.

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

Формат: INT LTR212_WriteUSR_Clb(PTLTR212 hnd, TLTR212_Usr_Clb *CALLIBR)

Параметры:

- **hnd** – указатель на структуру описания модуля **PTLTR212**;
- **CALLIBR** – указатель на структуру, содержащую пользовательские калибровки

Описание: Функция, записывающая пользовательские калибровки в ППЗУ модуля.

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

Внимание!!! Следует не путать типы данных: структура описания модуля (тип **TLTR212**) и **УКАЗАТЕЛЬ** на структуру описания модуля (тип **PTLTR212**).

4 Особенности выполнения процесса сбора данных.

4.1 Формирование цикла сбора и получения данных

Запуск сбора данных и его остановка осуществляются при помощи функций **LTR212_Start()** и **LTR212_Stop()**. Чтение массивов данных, получаемых из модуля, производится функцией **LTR212_Recv()**. В этой 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 получения запрашиваемого числа слов данных в миллисекундах. 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 **LTR212_GetErrorString()**, чтобы определить ошибку, применив код ошибки в качестве параметра.

*Необходимо помнить, что, поскольку в данном модуле используется 24-битный АЦП, то для передачи каждого сэмпла требуется 2 слова данных, первое из которых содержит старший байт 24-битного сэмпла, а второе – средний и младший байты. Поэтому в функции **LTR212_Recv()** следует указывать удвоенное количество получаемых слов по отношению к количеству точек сбора, с которых запрашивается информация.*

В полученном массиве данные располагаются по кадрам. **Кадр** представляет собой последовательность слов данных, полученных при выполнении одного цикла опроса всех задействованных каналов. Данные располагаются в кадре в порядке опроса физических каналов, т.е. в порядке возрастания их номеров. Каждый кадр входного массива содержит число элементов, равное **удвоенному** числу задействованных каналов. Состав кадра определяется таблицей логических каналов, т.к. именно она указывает, какие каналы включены, а какие – нет.

После получения из модуля массива данных функцией **LTR212_Rcv()** следует удалить все служебные поля из его элементов, проверить правильность полученных данных и, если требуется, перевести значения кода АЦП в значения напряжения сигнала в вольтах в соответствии с диапазонами входного напряжения каждого из каналов. Для этого предназначена функция **LTR212_ProcessData(*hnd, *src, *dest, *size, volt)**. Ее параметр ***src** представляет собой указатель на исходный массив данных, предварительно полученный функцией **LTR212_Recv()**; ***dest** – указатель на выходной массив, который может включать в себя или непосредственно коды АЦП, или значения напряжения в вольтах. Это определяется параметром **volt**. Если **volt=0**, то выходной массив содержит коды АЦП; если **volt=1**, то значения, пересчитанные в напряжение в вольтах. Но в любом случае этот массив должен иметь тип **double**. В выходном массиве **dest[]** данные также располагаются по кадрам, но в случае успешного завершения функции количество элементов в нем будет равно количеству точек сбора, т.е. вдвое меньше числа элементов входного массива **src[]**. Поэтому при вызове функции **LTR212_ProcessData()** параметр **size** следует установить равным удвоенному числу точек сбора, т.е. равному количеству элементов входного массива **src[]**. Функция изменит значение этого параметра, и при выходе из нее **size** будет равен уже половине исходного значения. Как правило, процесс сбора данных является циклическим (см. Рис. 1.). Для эффективной работы цикл сбора данных следует организовать в отдельном потоке. Далее следует небольшой пример получения и обработки данных при помощи функций **LTR212_Recv()** и **LTR212_ProcessData()**.

Например, мы хотим получить данные по двум точкам сбора с каждого канала. Пусть переменная **hltr212** – экземпляр структуры описания модуля типа **TLTR212**. В этом примере будем использовать ту же таблицу логических каналов, что и в [гл. 3.2](#), только диапазон входного сигнала у всех каналов положим **+/-80 мВ**. Тогда таблица логических каналов будет выглядеть следующим образом:

hltr212.LChTbl:

0x00010003
0x00020003
0x00040003
0x00050003
0x00070003
0x00080003

Имеем задействованные физические каналы: 1, 2, 4, 5, 7 и 8. После конфигурации модуля и запуска сбора данных, чтобы получить массив данных с двух точек для каждого канала, применяем функцию **LTR212_Recv()**. Поскольку число задействованных каналов 6 и мы хотим получить данные с двух точек сбора для каждого, то всего требуется получить информацию с $6*2=12$ точек. Как указывалось выше, параметр **size** должен содержать удвоенное количество точек сбора, т.е. **size=12*2=24**. Параметр **timeout** положим равным $0,5с=500мс$. **timeout= 500**. **src[]**- массив, куда будут записаны слова данных из модуля. После инициализации, открытия, конфигурации модуля и старта сбора данных применяем функцию с указанными выше параметрами: **LTR212_Recv(&hltr212, src, NULL, 24, 500)**. По возвращении из функции массив «сырых» данных **src[]** будет выглядеть, например, следующим образом:

src[]:

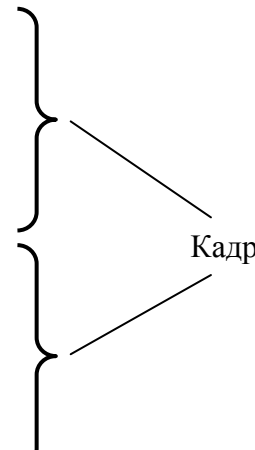
0x007F0800	}	Кадр
0xFF220810		
0x00840821		
0x03790831		
0x008C0843		
0x0E560853		
0x00820864		
0x03010874		
0x008A0886		
0x05140896		
0x 008E08A7		
0x 0D1508B7		
0x 007F08C0	}	Кадр
0x FF2708D0		
0x 008408E1		
0x 035508F1		
0x 008C0803		
0x 0E5D0813		
0x 00820824		
0x 02DB0834		
0x 008A0846		
0x 05110856		
0x 008E0867		
0x D200877		

Как видно из примера, сэмплы в кадрах располагаются циклически в порядке возрастания номеров физических каналов. Это «сырые данные», пришедшие непосредственно из крейт-контроллера. Номер физического канала представлен младшими четырьмя битами в каждом пакете данных. В этом счетчике каналов нумерация производится с нуля. Каждому сэмплу соответствуют два элемента массива. Первый содержит старший байт полученного сэмпла, второй –

средний и младший байты (более подробно о формате пакетов данных см. Приложение 2). Затем эти данные необходимо проверить, удалить служебные поля и, для нашего примера, пересчитать в значение напряжения входного сигнала. Для этого вызываем функцию **LTR212_ProcessData()**. Параметр **src[]** – это полученный ранее функцией **LTR212_Recv()** массив данных. Параметр **size** равен количеству его элементов, **size=24**. **dest[]** – выходной массив, в нашем случае содержащий значения напряжения в вольтах для входных сигналов. Параметр **volt=1**, т.к. мы хотим пересчитать данные из кода АЦП в значение напряжения входного сигнала. Вызов функции **LTR212_ProcessData()** будет выглядеть следующим образом: **LTR212_ProcessData(&hltr212, src, dest, &size, 1)**, где **size=24**. Функция, выполнив проверки, удалив все служебные поля и объединив все байты каждого сэмпла, получила значения кодов АЦП, а затем перевела эти значения в вольты. Пересчет в значения напряжения производится исходя из условия, что диапазон входного сигнала для всех каналов +/- 80 мВ, как было определено выше в таблице логических каналов. Выходной массив **dest[]** после работы функции **LTR212_ProcessData()** будет иметь следующий вид:

dest[]:

-2.117156982421875 E-6
2.508478164672852 E-3
7.534999847412110 E-3
1.257333755493164 E-3
6.262397766113282 E-3
8.781938552856446 E-3
-2.069473266701562 E-6
2.508134841918946 E-3
7.535066704614258 E-3
1.256971359252930 E-3
6.262369155883789 E-3
8.782043457031249 E-3



Как видно из этого примера, после работы функции **LTR212_ProcessData()** выходной массив содержит в два раза меньше элементов, чем исходный. Параметр **size** теперь также равен 12. Теперь эти данные можно использовать, например, для визуализации. Всегда следует помнить, что кадр, т.е. порядок следования данных в выходном массиве, соответствует порядку следования физических каналов, определенному в таблице логических каналов. Номер канала, которому соответствует данный элемент выходного массива, также определяется таблицей логических каналов. Это всегда нужно использовать при написании программного обеспечения. Резюмируя вышеизложенное, приведем небольшой фрагмент кода программы, иллюстрирующий получение данных из модуля исходя из условий рассматриваемого примера.

```

DWORD src[24];
DWORD dest[12];
DWORD size;
TLTR212 hltr212;

/*****
Инициализация, открытие, конфигурация и старт сбора данных
.....
*****/
// Получение данных:
// Если требуется, условия цикла
size=24;
LTR212_Recv(&hltr212, src, NULL, size, 500);
LTR212_ProcessData(src, dest, &size, 1);
.....
/*****
Завершение цикла сбора, останов сбора данных
*****/

```


Кроме вышеуказанных действий, функция осуществляет проверку правильности следования слов данных путем контроля специальных счетчиков (счетчик отправляемых слов и счетчик каналов), содержащихся в каждом слове. Функция сравнивает последовательность каналов, определенную при создании таблицы логических каналов, с последовательностью, полученной непосредственно от модуля. В случае несоответствия функция пропускает кадр, где это несоответствие было обнаружено, и на его место в выходном массиве `dest[]` записывает следующий кадр, если он верен. «Сбойный» кадр стирается целиком, независимо от того, в каком месте кадра произошло нарушение нужной последовательности каналов. Однако при этом **функция возвращает ошибку** несмотря на то, что массив подкорректирован и дальнейшая работа, в принципе, возможна. Функция также отслеживает состояние счетчика отправленных пакетов данных. В случае его сбоя продолжает дальнейшую обработку массива, но завершается с ошибкой. Независимо от наличия или отсутствия ошибки в счетчике отправленных пакетов данных счетчик каналов проверяется так, как это было описано выше, и сбойные кадры, если есть, удаляются. Подробнее об этих счетчиках см. [Приложение 2](#). Если были обнаружены «сбойные» кадры, то на выходе функции параметр `size` будет равен уже не половине исходного значения, а станет меньше этой половины, т.к. из выходного массива `dest[]` будут удалены кадры, в которых обнаружен сбой.

4.2 Особенности режимов сбора данных.

Модуль имеет три режима сбора данных, два из которых **четырёхканальные** и один - **восьмиканальный**.

В **четырёхканальных** режимах все четыре АЦП AD7730, установленные на плате модуля, работают синхронно и коммутации каналов не происходит. Сбор данных ведется непрерывно.

В **восьмиканальном** режиме происходит переключение каналов на каждом из АЦП AD7730. Схема сбора данных в восьмиканальном режиме следующая: на каждом из четырех АЦП AD7730, установленных на модуле, активизируется первый канал сбора данных (физические каналы 1, 2, 3, 4). Все АЦП запускаются синхронно на однократный сбор данных. Полученные отсчеты принимаются из регистров всех АЦП и отправляются в крейт-контроллер и ПК. Затем на каждом из АЦП активизируется второй канал (физические каналы 5, 6, 7, 8) и происходят те же действия. После этого опять включается первая группа каналов, и процесс повторяется циклически. В данном режиме во время заполнения аппаратных фильтров сбор данных ведется на частоте установленного режима фильтрации (150.1 Гц), однако частота выдачи данных в ПК будет значительно меньше (примерно 3,4 Гц), т.к. при коммутации каналов требуется время на перепрограммирование АЦП, сбор данных другой группой каналов и передачу данных в крейт-контроллер и ПК. Для определения частоты выдачи отсчетов как в четырехканальном, так и в восьмиканальном режимах, используется функция `LTR212_CalcFS()`.

Следует помнить, что в восьмиканальном режиме частота выдачи данных не является точной, т.к. процесс сбора данных не абсолютно синхронный! В частности, эта частота зависит от количества задействованных каналов. В связи с этим не следует использовать 8-канальный режим в приложениях, требовательных к точности частоты и временных интервалов.

В **четырёх- и восьмиканальном** режимах высокой точности возможно использование знакопеременного опорного напряжения. В этом случае питание мостов осуществляется переменным напряжением и дополнительно компенсируется влияние возможных термоЭДС, возникающих при подключении измерительных мостов к модулю. Частота переключения напряжения питания равна половине частоты сбора данных. Следует помнить, что **в четырёхканальном режиме средней точности знакопеременное опорное напряжение не применяется.**

5 Калибровка модуля.

5.1 Режимы калибровки.

Каждый из 4^x АЦП AD7730, установленных на плате модуля, имеет встроенные функции калибровки, специальные калибровочные регистры (смещения и усиления), а также встроенный Цифро-Аналоговый Преобразователь (ЦАП). Эти средства позволяют произвести внутреннюю или внешнюю калибровку каналов модуля.

С точки зрения данного программного обеспечения, калибровка представляет собой определение калибровочных коэффициентов и помещение их в специальную область ППЗУ модуля, именуемую в дальнейшем калибровочной областью. При сборе данных, если пользователь желает задействовать калибровочные коэффициенты, они будут загружаться из этой области в калибровочные регистры требуемых каналов АЦП AD7730.

Под **внутренней калибровкой** понимается запись заводских калибровочных коэффициентов (поставляемых фирмой ООО “А-Кард”) в калибровочную область ППЗУ модуля. Заводские калибровочные коэффициенты хранятся в другой области ППЗУ, которая используется только для хранения, и данные в ней не могут быть перезаписаны.

Под **внешней калибровкой** понимается определение калибровочных коэффициентов исходя из пользовательских условий измерения и запись этих коэффициентов в калибровочную область ППЗУ.

Калибровка нуля подразумевает вычисление (если требуется) и запись в калибровочную область ППЗУ калибровочных коэффициентов смещения и кода встроенного ЦАП.

При сборе данных с применением коэффициентов смещения они переписываются из калибровочной области ППЗУ в **калибровочные регистры смещения** требуемых каналов, а также в регистр ЦАПа. Это позволяет скорректировать смещение нулевого уровня входного сигнала.

Калибровка диапазона подразумевает вычисление (если требуется) и запись в калибровочную область ППЗУ калибровочных коэффициентов усиления. При сборе данных с применением коэффициентов усиления они переписываются из калибровочной области ППЗУ в **калибровочные регистры усиления** требуемых каналов. Это позволяет скорректировать соответствие фактического полного диапазона измеряемого сигнала заявленному диапазону для данного канала.

Полная калибровка представляет собой выполнение как калибровки нуля, так и калибровки диапазона. При этом заполняются нужными калибровочными коэффициентами как ячейки ППЗУ, соответствующие коэффициентам смещения, так и ячейки, соответствующие коэффициентам усиления.

Перед выполнением калибровки необходимо установить режим сбора данных, при работе на котором должны применяться определяемые калибровочные коэффициенты. Т.е. перед калибровкой необходимо заполнить поля структуры описания модуля. Калибровка выполняется посредством вызова функции **LTR212_Calibrate()**. Описание этой функции см. в [главе 3.3](#) данного Руководства.

В случае внутренней калибровки данная функция выполняет считывание коэффициентов из области хранения заводских коэффициентов в калибровочную область ППЗУ модуля, а в случае внешней калибровки – расчет коэффициентов и запись их в калибровочную область ППЗУ. После выполнения калибровки при установке поля **UseCalibration=1** структуры описания модуля и последующем вызове функции **LTR212_SetADC()** калибровочные коэффициенты будут считаны из этой области ППЗУ и записаны в **калибровочные регистры** соответствующих каналов АЦП. Параметр **mode** функции **LTR212_Calibrate()** определяет режим калибровки. При выполнении внутренней калибровки пользователю не обязательно подавать какие-либо сигналы на входы модуля, т.к. в этом случае процесс калибровки представляет собой лишь перезапись коэффициентов из одной области ППЗУ в другую. В случае **внешней калибровки нуля** на входы модуля следует подать напряжение, соответствующее нулевому значению. При **внешней калибровке диапазона** на входы каналов модуля следует подать напряжение, соответствующее максимальному положительному значению напряжения для данного диапазона. При полной внешней калибровке следует

сначала выполнить калибровку нуля, затем – диапазона. Ниже приводится описание режимов калибровки модуля.

Таблица 6

mode	Режим	Описание
0	Внутренняя калибровка нуля	Записывает заводские калибровочные коэффициенты смещения в соответствующие ячейки калибровочной области ППЗУ модуля.
1	Внутренняя калибровка диапазона	Записывает заводские калибровочные коэффициенты усиления в соответствующие ячейки калибровочной области ППЗУ модуля.
2	Полная внутренняя калибровка	Записывает заводские калибровочные коэффициенты смещения и усиления в соответствующие ячейки калибровочной области ППЗУ модуля.
3	Внешняя калибровка нуля	Рассчитывает калибровочные коэффициенты смещения и записывает их в соответствующие ячейки калибровочной области ППЗУ модуля. Во время калибровки на входы модуля следует подать постоянное напряжение, соответствующее нулевому сигналу
4	Внешняя калибровка диапазона	Рассчитывает калибровочные коэффициенты усиления и записывает в соответствующие ячейки калибровочной области ППЗУ модуля. Во время калибровки на входы модуля следует подать постоянное напряжение, соответствующее максимальному положительному напряжению диапазона входного сигнала.
5	Внутренний диапазон + внешний ноль	Записывает заводские калибровочные коэффициенты усиления в соответствующие ячейки калибровочной области ППЗУ модуля, а затем рассчитывает и записывает в ППЗУ калибровочные коэффициенты смещения. Во время калибровки на входы модуля следует подать постоянное напряжение, соответствующее нулевому сигналу.
6	Внешняя калибровка диапазона (вторая стадия при полной внешней калибровке)	Используется ТОЛЬКО при полной внешней калибровке ПОСЛЕ выполненной внешней калибровки нуля. Во время калибровки на входы модуля следует подать постоянное напряжение, соответствующее максимальному положительному напряжению сигнала.
7	Внешняя калибровка нуля, с сохранением масштабных коэффициентов	Аналог «Внешней калибровки нуля», однако в процессе выполнения сохраняются определённые ранее при полной внешней калибровке коэффициенты масштаба.

Параметр **reset** функции **LTR212_Calibrate()** позволяет указать, нужно ли очищать калибровочные регистры АЦП AD7730 перед проведением калибровки. Например, если нужно подкорректировать калибровку нуля, не изменяя при этом коэффициенты усиления, то следует откалибровать требуемые каналы без предварительной очистки регистров. Чтобы очистить регистры перед калибровкой, следует установить параметр **reset=1**. Если этого не требуется, то **reset=0**.

Программное обеспечение также предполагает возможность автоматического использования заводских калибровочных коэффициентов. При установке флага **UseFabricClb=1** при каждом вызове функции **LTR212_SetADC()** при смене диапазона каналов автоматически будут загружаться в область ППЗУ и соответствующие этому диапазону заводские калибровочные коэффициенты, как смещения, так и усиления.

В таблице ниже указаны максимальные значения напряжения смещения, которые можно компенсировать при помощи внешней калибровки.

Таблица 7

Компенсруемое смещение в диапазонах:	
10mV, ±10mV	±78,7mV *
19mV, ±19mV	±78,7mV*
40mV, ±40mV	±79,5mV*
80mV, ±80mV	±81,5mV*

* При использовании опорного напряжения 2,5 В указанные значения вдвое меньше.

5.2 Особенности использования различных режимов калибровки.

Калибровка модуля *LTR-212(M)* имеет ряд важных особенностей, которые следует учитывать при программировании:

- **Заводские калибровочные** коэффициенты:
 - для *LTR212* и *LTR212M-3* эти коэффициенты используются **ТОЛЬКО** при опорном напряжении 5.0 В. При опорном напряжении 2.5 В функции внутренней калибровки будут завершаться с ошибкой!!! При работе с использованием опорного напряжения 2.5 В применяйте внешнюю калибровку.
 - для *LTR212M-1* и *LTR212M-2* эти коэффициенты используются как при опорном напряжении 5.0 В, так и при 2.5 В.
- При **внутренней калибровке** в **четырёхканальном** режиме **средней** точности возможно использование **ТОЛЬКО** заводских коэффициентов усиления. Т.е. в этом режиме допускается выполнение лишь внутренней калибровки диапазона. При попытке внутренней калибровки нуля или полной внутренней калибровки функция **LTR212_Calibrate()** завершится с ошибкой. Внешнюю калибровку можно использовать без ограничений, как и режим “Внутренняя калибровка диапазона + внешняя калибровка нуля”.
- Программное обеспечение предусматривает режим **прямого использования заводских калибровочных коэффициентов**. Он включается путем установки поля **UseFabricClb** структуры описания модуля в единицу (**UseFabricClb=1**). Если включен этот режим, то заводские калибровочные коэффициенты, как смещения, так и усиления, напрямую загружаются из области хранения заводских калибровочных коэффициентов ППЗУ в калибровочные регистры АЦП AD7730. Выбор нужных коэффициентов осуществляется **автоматически** согласно выбранным каналам и соответствующим им диапазонам входного сигнала. При этом калибровочная область ППЗУ не изменяется. Этот режим позволяет автоматически загружать верные заводские калибровочные коэффициенты при переходе на разные диапазоны измерений. А также дает возможность использовать заводскую калибровку, при этом не стирая пользовательские калибровочные коэффициенты определенные ранее и находящиеся в калибровочной области ППЗУ. Таким образом, у пользователя имеется возможность легко переключиться с собственных калибровочных коэффициентов на заводские и обратно. Этот режим не работает в четырехканальном режиме средней точности. Также этот режим не работает для *LTR212* и *LTR212M-3* при опорном напряжении 2.5 В. А вот для *LTR212M-1* и *LTR212M-2* данный режим функционирует при обоих опорных напряжениях: 2.5 В и 5.0 В.

Таблица поясняет сказанное вышесказанное для *LTR212* и *LTR212M-3*.

Таблица 8

Режим внутренней калибровки	Внутренняя калибровка нуля	Внутренняя калибровка диапазона	Полная внутренняя калибровка	Внутренняя калибровка диапазона + внешняя калибровка нуля	Режим прямого использования заводских калибровочных коэффициентов
Четырехканальный режим средней точности	-	+	-	+	-
Четырехканальный режим высокой точности	+	+	+	+	+
Восьмиканальный режим высокой точности	+	+	+	+	+
Любой режим при опорном напряжении 2,5 В	-	-	-	-	-

Еще раз следует подчеркнуть, что **внешняя калибровка может использоваться без каких-либо ограничений.**

При полной внешней калибровке следует сначала выполнить внешнюю калибровку нуля (вызов функции **LTR212_Calibrate()** с параметром **mode=3**), а затем выполнить внешнюю калибровку диапазона, но как вторую стадию полной внешней калибровки (вызов функции **LTR212_Calibrate()** с параметром **mode=6**. Но не **mode=4!!!**).

Внимание!!! Будьте внимательны при подключении сигналов во время калибровки! Если какие-то каналы, подлежащие калибровке, «висят в воздухе» или сигнал на них выходит за пределы диапазона напряжения входного канала, то калибровка завершится с ошибкой. Всегда в случае ошибочного выполнения функции калибровки проанализируйте значение маски логических каналов. Если она изменилась, то для некоторых каналов не удалось подобрать значения встроенного ЦАП. Это может указывать либо на то, что подаваемый сигнал вне диапазона значений, компенсируемых ЦАПом, либо входные сигналы «висят в воздухе».

6 Цифровая фильтрация.

6.1 Цифровые фильтры, используемые в модуле LTR212.

Все данные, собираемые модулем, подвергаются процедуре цифровой фильтрации. Каждый из АЦП AD7730, установленный на плате модуля, имеет два встроенных аппаратных цифровых фильтра: фильтр первой ступени и фильтр второй ступени. Фильтр первой ступени – это фильтр низкой частоты типа sinc³, основная цель которого - устранение шумов квантования, возникающих в модуляторе. Фильтр второй ступени – это цифровой КИХ-фильтр с 22 отводами, который обрабатывает сигнал с выхода фильтра первой ступени.

Аппаратный цифровой фильтр первой ступени задействован на всех трех режимах работы модуля LTR212. Аппаратный цифровой фильтр второй ступени – только в 4-канальном режиме высокой точности и в 8-канальном режиме высокой точности. В 4-канальном режиме средней точности этот фильтр ОТКЛЮЧЕН!

Однако в этом режиме возможно использование программных цифровых фильтров, разработанных фирмой Л-КАРД. Об аппаратных фильтрах 1-й и 2-й ступени можно подробнее прочитать в документе AD7730 Data Sheet (www.analog.com/en/prod/0,2877,AD7730,00.html). О программных цифровых фильтрах более подробно будет рассказано в следующей главе.

АЧХ работы модуля с применением программных и аппаратных фильтров приведены в документе «Крейтовая система LTR. Руководство пользователя», Приложение А.2.2.1.

Напомним, что при включённом программном КИХ-фильтре перед получением первого фильтрованного отсчета необходимо заполнить сэмплами буфер в памяти DSP модуля, равный порядку фильтра. Например, если порядок фильтра 225, то после старта АЦП сначала будет заполнен буфер фильтра (225 слов для каждого канала), и только после это модуль начнет выдавать данные.

Если включен программный БИХ-фильтр, то первые 30 отсчетов каждого канала используются для заполнения линии задержки и для обеспечения стабилизации фильтра. Эти данные модуль не передает в крейт-контроллер и ПК.

6.2 Применение программных фильтров

Фирмой Л-КАРД разработан набор программных цифровых фильтров, реализуемых на уровне DSP модуля. Они позволяют производить цифровую фильтрацию данных на максимальной частоте сбора (7680 Гц), что невозможно сделать с использованием встроенных аппаратных фильтров 2-й ступени. Следует помнить, что в данной версии программного обеспечения **при включенных программных фильтрах АЦП работают в 16-битном режиме!** Однако 16-битные данные передаются в крейт-контроллер двумя пакетами, как и 24-битные. Поэтому функция **LTR212_ProcessData()** применяется так же, как и при иных режимах работы. Набор программных фильтров включает в себя БИХ-фильтр 2-го порядка, выравнивающий исходную АЧХ в заданной полосе частот с точностью $\pm 0,02$ дБ, и 5 КИХ-фильтров различного порядка и с разными значениями частот начала полосы задержания. Имеющиеся в системе программные фильтры должны работать **только в 4-канальном режиме средней точности** на частоте сбора данных 7680 Гц. Коэффициенты фильтров хранятся в файлах, которые считываются функцией **LTR212_SetADC()**, затем они передаются в DSP модуля. Полные имена файлов с коэффициентами программных фильтров должны быть предварительно указаны в элементах структуры описания модуля **filter.IIR_Name** и **filter.FIR_Name**. Остальные поля подструктуры **filter** будут заполнены автоматически значениями, считанными из файла.

Файл БИХ-фильтра содержит только коэффициенты. Для КИХ-фильтра первые строки файла должны содержать сведения о порядке фильтра и коэффициенте децимации в следующем формате:

```
FS=XXXX HZ  
DECIMATION= XX
```

Например,

```
FS= 7680 HZ
DECIMATION= 24
0001
0001
.....
Коэффициенты
.....
```

Внимание! Порядок КИХ-фильтра должен быть не более 255!

Функция **LTR212_SetADC()** «понимает» только такой формат файла КИХ-фильтра. Пользователь может самостоятельно рассчитать цифровой фильтр и записать его коэффициенты в файл. Функция **LTR212_SetADC()** прочитает коэффициенты и загрузит их во внутреннюю память DSP модуля. Фильтр можно разработать, например, в среде QED. Полученные коэффициенты следует умножить на 32768 и результат округлить до целочисленного двухбайтного значения. Затем нужно сформировать текстовый файл указанного выше формата, куда внести коэффициенты. Именно в таком виде программные цифровые фильтры смогут их применить.

7 Проверка ППЗУ.

Библиотека пользовательского интерфейса содержит полезную функцию проверки целостности данных в ППЗУ модуля **LTR212_TestEEPROM()**. Если функция выполнится без ошибки, это значит, контрольная сумма данных в ППЗУ модуля верна. Если же функция вернет значение '-2031' (Неверная контрольная сумма ППЗУ), то данные в ППЗУ повреждены. Несмотря на это, можно продолжать работу с модулем. Однако следует иметь в виду, что данные в каких-то ячейках ППЗУ могут быть неверными. Эту функцию полезно применять, например, каждый раз после открытия модуля (после выполнения функции **LTR212_Open()**) для контроля целостности калибровочных коэффициентов и идентификационной записи.

Приложение 1. Примеры написания программ.

П1.1 Примеры конфигураций.

Перед заданием конфигурации следует инициализировать и открыть канал связи с модулем. Это делается посредством вызова следующих функций: **LTR212_Init()** и **LTR212_Open()**. Затем следует заполнить поля структуры описания модуля требуемыми значениями. Ниже приводятся примеры задания конфигурации модуля (определение полей структуры описания модуля):

1. **Четырехканальный режим средней точности**, частота сбора 7600 Гц, включены программные фильтры. Выбран программный КИХ-фильтр с частотой начала полосы задержки 345 Гц. Задействованы все 4 канала. Диапазон напряжения входного сигнала всех каналов ± 80 мВ. Используются калибровочные коэффициенты. Опорное напряжение постоянное 5 В.

```
TLTR212 conf_1;           // Объявляем структуру типа TLTR212
...
conf_1.AcqMode=0;         // 4-канальный режим средней точности
conf_1.UseClb=1;          // Используем калибровочные коэффициенты
conf_1.UseFabricClb=0;    // Не используем заводские калибр. коэфф.
conf_1.LChQnt=4;          // Количество логических каналов – 4

// Заполняем таблицу логических каналов. Для каждого канала устанавливаем
// диапазон входного сигнала  $\pm 80$  мВ.

for(i=0; i< conf_1.LChQnt; i++)
    conf_1.LChTbl[i]=LTR212_CreateLChannel(i, 3);

// Задаем параметры программных фильтров
conf_1.filter.IIR=1;       // БИХ-фильтр включен
conf_1.filter.FIR=1;       // КИХ-фильтр включен
conf_1.filter.IIRName="C://Filter// D212_IIR.flt"; // Файлы фильтров
conf_1.filter.FIRName="C://Filter// D212_345.flt";

// Устанавливаем полярность и величину опорного напряжения
conf_2.AC=0;               // Постоянное опорное напряжение
conf_1.REF=1;              // Опорное напряжение 5 В.
```

2. **4-канальный режим высокой точности**, частота сбора 150.1 Гц, программные фильтры отключены. Включены аппаратные фильтры на АЦП AD7730. Задействованы 2 логических канала, соответствующие физическим каналам 1 и 4. Диапазоны ± 10 мВ и $+40$ мВ соответственно. Используются калибровочные коэффициенты. Опорное напряжение постоянное 2,5 В.

```
TLTR212 conf_2;           // Объявляем структуру типа TLTR212
...
conf_2.AcqMode=1;         // 4-канальный режим высокой точности
conf_2.UseClb=1;          // Используем калибровочные коэффициенты
conf_1.UseFabricClb=0;    // Не используем заводские калибр. коэфф.
conf_2.LChQnt=2;          // Количество логических каналов – 2

// Заполняем таблицу логических каналов
conf_2.LChTbl[0]=LTR212_CreateLChannel(1, 0); // физ.канал 1, диапазон  $\pm 10$  мВ
conf_2.LChTbl[1]=LTR212_CreateLChannel(4, 6); // физ.канал 4, диапазон  $+40$  мВ

// Задаем параметры программных фильтров
conf_2.filter.IIR=0;       // БИХ-фильтр отключен
```



```
conf_2.filter.FIR=0; //КИХ-фильтр отключен
```

```
// Опорное напряжение
```

```
conf_2.AC=0; // Постоянное опорное напряжение  
conf_2.REF=0; // Опорное напряжение 2,5 В.
```

3. **Четырехканальный режим высокой точности**, частота сбора 150.1 Гц, программные фильтры отключены. Задействованы 3 логических канала, соответствующие физическим каналам 1, 3 и 4. Диапазоны каналов ± 10 мВ, +20 мВ и +80 мВ соответственно. Используются заводские калибровочные коэффициенты. Опорное напряжение знакопеременное 5 В.

```
TLTR212 conf_3; // Объявляем структуру типа TLTR212
```

```
conf_3.Mode=1; // 4-канальный режим высокой точности  
conf_3.UseClb=0; // Не используем калибровочные коэффициенты  
conf_3.UseFabricClb=1; // Используем заводские калибр. коэфф.  
conf_3.LChQnt=3; // Количество логических каналов – 3
```

```
// Заполняем таблицу логических каналов
```

```
conf_3.LChTbl[0]=LTR212_CreateLChannel(1,0); // физ.канал 1, диапазон  $\pm 10$ мВ  
conf_3.LChTbl[1]=LTR212_CreateLChannel(3,5); // физ.канал 3, диапазон +20мВ  
conf_3.LChTbl[2]=LTR212_CreateLChannel(4,7); // физ.канал 4, диапазон +80мВ
```

```
// Задаем параметры программных фильтров
```

```
conf_3.filter.IIR=0; // БИХ-фильтр отключен  
conf_3.filter.FIR=0; // КИХ-фильтр отключен
```

```
// Опорное напряжение
```

```
conf_3.AC=1; // Знакопеременное опорное напряжение  
conf_3.codec.REF=1; // Опорное напряжение 5 В.
```

4. **8-канальный режим высокой точности**, приближенная частота сбора 3.4 Гц, программные фильтры отключены. Режим с прерыванием отключен. Задействованы 6 логических каналов, соответствующие физическим каналам: 1, 2, 3, 4, 5, 8. Диапазоны всех каналов ± 20 мВ. Используются калибровочные коэффициенты. Опорное напряжение постоянное 5 В.

```
TLTR212 conf_4; // Объявляем экземпляр структуры описания модуля
```

```
conf_4.size=sizeof(PTLTR212); // Размер структуры  
conf_4.Mode=1; // 8-канальный режим высокой точности  
conf_4.UseClb=1; // Используем калибровочные коэффициенты  
conf_4.UseFabricClb=0; // Не используем заводские калибр. коэфф.  
conf_4.LChQnt=6; // Количество виртуальных каналов – 6
```

```
// Заполняем таблицу логических каналов
```

```
conf_4.LChTbl[0]=LTR212_CreateLChannel(1,1); // физ.канал 1, диапазон  $\pm 20$ мВ  
conf_4.LChTbl[1]=LTR212_CreateLChannel(2,1); // физ.канал 2, диапазон  $\pm 20$ мВ  
conf_4.LChTbl[2]=LTR212_CreateLChannel(3,1); // физ.канал 3, диапазон  $\pm 20$ мВ  
conf_4.LChTbl[3]=LTR212_CreateLChannel(4,1); // физ.канал 4, диапазон  $\pm 20$ мВ  
conf_4.LChTbl[4]=LTR212_CreateLChannel(5,1); // физ.канал 5, диапазон  $\pm 20$ мВ  
conf_4.LChTbl[5]=LTR212_CreateLChannel(8,1); // физ.канал 8, диапазон  $\pm 20$ мВ
```

```
// Задаем параметры программных фильтров
```

```

conf_4.filter.IIR=0;           // БИХ-фильтр отключен
conf_4.filter.FIR=0;         //КИХ-фильтр отключен

// Опорное напряжение
conf_4.codec.AC=0;           // Постоянное опорное напряжение
conf_4.codec.REF=1;         // Опорное напряжение 5 В.

```

П1.2. Пример приложения.

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

```

#pragma hdrstop

#include "ltr\include\ltr212api.h"
/* остальные заголовочные файлы */
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <conio.h>

#define ACQ_BLOCKS_QNT (2)
#define TOTAL_BLOCKS_QNT (10)
#define POINTS_PER_CHANNEL (256)
#define CHANNELS_QNT (4)

TLTR212 hltr212;           // Экземпляр структуры описания модуля
HANDLE AcqThreadHnd;      // Поток сбора данных
CHAR ErrorString[255];    // Строка для вывода описания ошибки
CHAR MsgString[255];     // Строка для вывода сообщений на консоль

double voltage[ACQ_BLOCKS_QNT][CHANNELS_QNT*POINTS_PER_CHANNEL];
static volatile int RunFlag = 0; // Флаг, указывающий, что идет сбор данных
static volatile int BlockReady[ACQ_BLOCKS_QNT];

static DWORD WINAPI AcqThread (LPVOID param);

static DWORD WINAPI AcquireThread(LPVOID param)
{
    int i;
    int err=0;
    INT DataCntr=0;
    DWORD data[2*POINTS_PER_CHANNEL*CHANNELS_QNT];
    DWORD to; // Таймаут
    INT AcqBlockCntr=0;
    int BlockCntr=0;
    DWORD size; // Размер порции данных
    DWORD ExitCode;

    /* Функция, запускаемая в качестве потока сбора данных. */
    for(i=0;i<ACQ_BLOCKS_QNT;i++)
        BlockReady[i]=0;

    err=LTR212_Start(&hltr212);
    if(err)
    {
        strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
        CharToOem(ErrorString,ErrorString);
        printf("%s",ErrorString);
        LTR212_Stop(&hltr212);
        RunFlag=0;
    }
}

```

```

}

size=2*POINTS_PER_CHANNEL*CHANNELS_QNT; // Размер получаемой порции данных
to=LTR212_CalcTimeout(&hltr212, POINTS_PER_CHANNEL);

while(RunFlag) // Пока запущен сбор данных...
{
    size=2*POINTS_PER_CHANNEL*CHANNELS_QNT;

    DataCntr=LTR212_Recv(&hltr212, data, NULL, size, to);
    if(DataCntr!=size)
    {
        RunFlag=0;
        if(DataCntr>=0)
        {
            sprintf(MsgString,"%s", "Ошибка! Получено меньше сэмплов, чем было
заказано!\n");
            CharToOem(MsgString,MsgString);
            printf("%s\n",MsgString);
        }
        else
        {
            err=DataCntr;
            strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
            CharToOem(ErrorString,ErrorString);
            printf("%s",ErrorString);
        }
        break;
    } // к if(DataCntr!=size)

    err=LTR212_ProcessData(&hltr212, data, voltage[BlockCntr], &size, 1);
    if(err)
    {
        strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
        CharToOem(ErrorString,ErrorString);
        printf("%s",ErrorString);
        RunFlag=0;
        break;
    }

    AcqBlockCntr++;

    sprintf(MsgString,"Получена %d-я порция данных", AcqBlockCntr);
    CharToOem(MsgString,MsgString);
    printf("%s\n",MsgString);

    /* Для синхронизации с записью в файл, производимой в основном потоке,
используем чередование блоков */

    BlockReady[BlockCntr]=1;
    BlockCntr++;

    if(BlockCntr>=ACQ_BLOCKS_QNT)
        BlockCntr=0;
} // к while(RunFlag)

RunFlag=0;
err=LTR212_Stop(&hltr212);

```

```

if(err)
{
    strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
    CharToOem(ErrorString,ErrorString);
    printf("%s",ErrorString);
}

ExitThread(0);
return 0;
}

void main()
{

DWORD AcqThreadId;
DWORD ThreadSatus;

INT TotalBlockCntr; // Счетчик блоков, записанных в файл
INT err; // Переменная для хранения кода ошибки
INT i; // Используется для счетчиков итераций в циклах
DWORD data[2*5000]; /* Массив, в который запишутся данные, полученные от
                    модуля*/
FILE *DataFile;
const char FileName[] = "ltr212_data.bin";

int BlockCntr=0;
int BlockNumber;

/* Инициализируем канал связи с модулем. Поля структуры описания модуля
заполняются значениями по умолчанию. */

sprintf(MsgString,"%s", "Инициализация структуры описания модуля\n");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

err=LTR212_Init(&hltr212);
if(err)
{
    strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
    CharToOem(ErrorString,ErrorString);
    printf("%s",ErrorString);
    LTR212_Close(&hltr212);
    Sleep(3000);
    return;
}

// Открываем канал связи с модулем. Сетевой адрес и номер порта по умолчанию
// Серийный номер первого найденного крейта;
// Номер слота - 0;

err=LTR212_Open(&hltr212, SADDR_DEFAULT, SPORT_DEFAULT, "", CC_MODULE1,
"ltr212.bio");
if(err)
{
    if(err==LTR_WARNING_MODULE_IN_USE)
    {
        strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
        CharToOem(ErrorString,ErrorString);
        printf("%s",ErrorString);
    }
}

```

```

else
{
    strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
    CharToOem(ErrorString,ErrorString);
    printf("%s",ErrorString);
    Sleep(3000);
    return;
}
}

/* Заполнение полей для информации. Здесь сделано только для демонстрации */
sprintf(MsgString,"Имя модуля: %s", hltr212.ModuleInfo.Name);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Версия BIOSа: %s", hltr212.ModuleInfo.BiosVersion);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Дата создания BIOSа: %s", hltr212.ModuleInfo.BiosDate);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

sprintf(MsgString,"Серийный номер модуля: %s", hltr212.ModuleInfo.Serial);
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

// Приводим заполнение полей структуры описания модуля требуемыми значениями

hltr212.size=sizeof(TLTR212); // Размер структуры
hltr212.AcqMode=1;           // 4-канальный режим высокой точности
hltr212.UseClb=0;           // Не используем пользовательские калибровочные коэфф.
hltr212.UseFabricClb=1;     // Используем заводские калибр. коэфф.
hltr212.LChQnt=4;           // Количество логических каналов - 4

hltr212.REF=1;              // Опорное напряжение 5 В.
hltr212.AC=0;               // Постоянное опорное напряжение

for(i=0; i<hltr212.LChQnt; i++)
    hltr212.LChTbl[i]=LTR212_CreateLChannel(i+1,3);

// Передаем параметры АЦП управляющей программе модуля

err=LTR212_SetADC(&hltr212);
if(err)
    if(err)
    {
        strcpy(ErrorString, (char *) LTR212_GetErrorString(err));
        CharToOem(ErrorString,ErrorString);
        printf("%s",ErrorString);
        LTR212_Close;
        Sleep(3000);
        return;
    }

DataFile = fopen(FileName, "wb");

```

```

if(DataFile==NULL)
{
    strcpy(MsgString, "Невозможно создать файл!");
    CharToOem(MsgString,MsgString);
    printf("%s\n",MsgString);
    LTR212_Close(&hltr212);
    Sleep(3000);
    return;
}

RunFlag = 1;

// Создаем поток сбора данных
AcqThreadHnd=CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)AcquireThread,
                          NULL, CREATE_SUSPENDED, (LPDWORD) &AcqThreadId);

if(AcqThreadHnd==NULL)
{
    strcpy(MsgString, "Невозможно создать поток сбора данных!");
    CharToOem(MsgString,MsgString);
    printf("%s\n",MsgString);
    LTR212_Close(&hltr212);
    Sleep(3000);
    return;
}

// Запускаем поток
ResumeThread(AcqThreadHnd);

strcpy(MsgString, "Поток сбора данных запущен!\n");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

TotalBlockCntr = 0;

while(RunFlag)
{
    if(BlockReady[BlockCntr]) // Если очередной блок считан
    {
        fwrite(voltage[BlockCntr], sizeof voltage[0][0],
              CHANNELS_QNT*POINTS_PER_CHANNEL, DataFile);

        TotalBlockCntr++; // Счетчик блоков

        sprintf(MsgString, "%d-я %s\n", TotalBlockCntr, "порция записана в файл");
        CharToOem(MsgString,MsgString);
        printf("%s\n",MsgString);

        BlockReady[BlockCntr]=0;
        BlockCntr++;

        if(BlockCntr>=2)
            BlockCntr=0;

        if (TotalBlockCntr >= TOTAL_BLOCKS_QNT)
        {
            strcpy(MsgString, ">> Сбор данных успешно завершен.\n");
            CharToOem(MsgString,MsgString);
            printf("%s\n",MsgString);
        }
    }
}

```

```

    RunFlag = 0;
}
} // κ if(BlockReady[BlockCntr]...)
} // κ while(RunFlag)

//Отслеживаем завершение потока
WaitForSingleObject(AcqThreadHnd, INFINITE);
GetExitCodeThread(AcqThreadHnd, &ThreadSatus);

CloseHandle(AcqThreadHnd);

strcpy(MsgString, "Поток сбора данных удален.\n");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

fclose(DataFile);
strcpy(MsgString, "Файл для записи данных закрыт.\n");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);

strcpy(MsgString, ">> Для выхода нажмите любую клавишу\n");
CharToOem(MsgString,MsgString);
printf("%s\n",MsgString);
while(!kbhit())
    continue;
return;
}

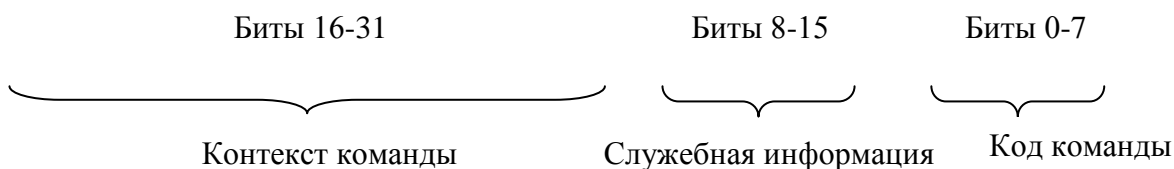
```

Приложение 2. Протокол обмена данных 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 LTR212.

Все команды от крейт-контроллера к модулю и подтверждения этих команд представляют собой 4-байтные **командные слова**. Данные, собранные АЦП модуля 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 LTR212 имеет следующий вид:

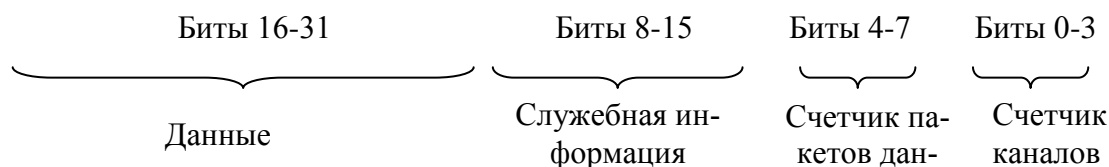


- **Код команды** – число, определяющее процедуру *БИОС*'а, которую следует выполнить процессору модуля.
- **Служебная информация** - информация о номере слота, бит-признак командного слова, временная метка, секундная метка. Этот байт несет в себе информацию уровня крейт-контроллера. Кроме бита-признака команды или данных, эти биты не передаются в модуль и обратно.
- **Контекст команды** – значения, передающиеся вместе с командой и используемые процессором при ее выполнении. Например, параметры калибровки, содержимое регистров АЦП при их программировании и т.д.

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!!!). Например, содержимое считанного регистра АЦП AD7730.

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 АЦП, передаваемые из модуля в крейт-контроллер и ПК.
- **Служебная информация** - информация о номере слота, бит-признак слова данных, временная метка, секундная метка. Этот байт несет в себе информацию уровня крейт-контроллера. Кроме бита-признака команды или данных, эти биты не передаются в модуль и обратно.
- **Счетчик пакетов данных** – 4-битный счетчик пакетов данных, отправляемых модулем в крейт-контроллер. При отправке каждого нового пакета процессор инкрементирует значение этого счетчика, которое впоследствии используется для проверки правильности следования пакетов из модуля.
- **Счетчик каналов** – в это поле для каждого пакета данных процессор записывает номер физического канала, с которого были получены отправляемые данные. Используется для проверки правильности следования пакетов из модуля. Нумерация каналов начинается с нуля.