Современные устройства сбора данных

L-502

Руководство программиста

Ревизия 1.0.2

Февраль 2013

Автор руководства:
Борисов А.В.

ООО "Л КАРД"

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

тел.: (495) 785-95-25
факс: (495) 785-95-14

Адреса в Интернет:

www.lcard.ru

E-Mail:

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

Представители в регионах:

Украина: ХОЛИТ Дэйта Системс,www.holit.com.ua, (044) 241-6754
Санкт-Петербург: Autex Spb Ltd., www.autex.spb.ru, (812) 567-7202
Новосибирск: Сектор-Т, www.sector-t.ru, (383-2) 396-592
Казань: ООО 'Шатл', shuttle@kai.ru, (8432) 38-1600
Екатеринбург: Авеон, aveon@aveon.ru, +7(343) 381-75-75
Пенза: НПП Технолинк, http://www.tl.ru/ru/departments/industry (8412) 49-10-59

Таблица 1: Ревизии текущего документа

Ревизия	**Дата**	**Описание**
1.0.0	27.06.2012	Первая ревизия данного документа
1.0.1	22.11.2012	Добавлено описание использования библиотеки с программами на C# и в LabView, добавлено описание установки для ОС Linux, а также описание функций для циклического вывода
1.0.2	20.02.2012	Добавлена ссылка на исходные коды SDK. Описание установки пакетов для Linux вынесено в отдельный документ

Оглавление

Глава 1

О чем этот документ

Данный документ предназначен в первую очередь для программистов, которые собираются писать свои программы для работы с модулями L502 с использованием предоставляемой фирмой "Л Кард" библиотеки l502api.

В данном документе рассматривается вопрос подключения библиотеки к проекту пользователя, дается подробное описание интерфейсных функций, предоставляемых библиотекой, а также дается описание основных подходов к использованию этих интерфейсных функций.

Сама библиотека написана на языке *C* и все объявления функций и типов, а также примеры приводятся на языке *C*, однако все привязки к другим языкам программирования являются лишь обертками над библиотекой C и все функции, типы и параметры сохраняют свое значения и для других языков программирования. Поэтому этот документ необходимо прочитать и пользователям, пишущим на других языках программирования. Примеры для других языков можно установить вместе с ПО разработчика для модулей PCI-Express фирмы "Л Кард".

В настоящем документе не рассматриваются какие-либо вопросы, касающиеся подключения сигналов и характеристик модуля, а также лишь в общем затрагиваются принципы работы самого модуля. Эти вопросы рассматриваются в "Руководстве пользователя", с которым рекомендуется ознакомится перед прочтением данного документа.

Также в данном документе не рассматривается задача написания своей прошивки для сигнального процессора модуля и работа с модулем без использования предоставляемой "Л Кард" библиотеки. Эти вопросы рассматриваются в "Низкоуровневом описании".

Глава 2

Установка и подключение библиотеки к проекту

Для написания собственного программного обеспечения, работающего с модулями L502, необходимо выполнить следующее:

1. Установить драйвер для модулей с интерфейсом PCI Express фирмы "Л Кард"
2. Установить динамическую библиотеку (l502api.dll для Windows или l502api.so для Linux) в директорию, присутствующую в соответствующей переменной окружения, либо в директорию с проектом. Динамическая библиотека необходима при написании программ на любом языке программирования, так как все привязки к языкам работают через библиотеку.
3. Подключить библиотеку к проекту.

Для ОС Windows предоставляется установщик программного обеспечения для разработчика "L-Card PCI Express SDK", который позволяет установить драйвер для устройства, установить динамическую библиотеку в системную папку, а также установить все файлы, необходимые для подключения библиотеки к проекту и примеры в указанную папку. В дальнейшем в данной главе с помощью LPCIE_SDK_DIR будет обозначаться указанная папка при установке.

Подключение к проекту зависит от используемого языка и среды.

2.1 Подключение библиотеки при написании программы на языках C/C++.

При написании на C/C++ при подключении необходимо выполнить следующее:

1. Включить в проект заголовочный файл "l502api.h", добавив при этом в проекте к путям для заголовочных файлов директорию LPCIE_SDK_DIR/inc.
2. Добавить в проект файл линкера для используемого компилятора:
   - **Microsoft Visual C++** : LPCIE_SDK_DIR/lib/msvc
   - **Microsoft Visual C++** 64-битный компилятор (подробнее о 64-битной версии описано ниже): LPCIE_SDK_DIR/lib/msvc64
   - **Borland C++/Borland C++ Builder** : LPCIE_SDK_DIR/lib/borland

- **MinGW** : LPCIE_SDK_DIR/lib/mingw
- **MinGW** 64-битный компилятор: LPCIE_SDK_DIR/lib/mingw64

2.2 Использование библиотеки в проекте на Delphi

Для написания программ на Delphi с использования API для работы с модулем L502, необходимо включить в проект программы два файла: LPCIE_SDK_DIR/pas/lpcieapi.pas и LPCIE_SDK_DIR/pas/l502api, которые представляют собой обертку над библиотекой l502api.dll. В файлах, использующих типы и функции из этого документа, необходимо подключить модуль l502api с помощью `uses l502api;`. При этом для 64-битного компилятора используются те же файлы, что и для 32-битного (см. 64-битная версия библиотеки).

Все функции, типы и константы библиотеки отображаются в Delphi один к одному, за исключением следующих моментов:

- все строки (серийные номера, строки с описанием кодов ошибок)преобразуются оберткой в тип `sting`, который используется стандартно для представления строк в Delphi (следует не забывать, что в последних версиях среды этот тип представляет собой юникодную строку). Исключением является структура t_l502_info с информацией о модуле, в которой строки представлены массивом `AnsiChar` фиксированной длины.

- все функции работающие с массивами принимают в качестве параметра открытый массив (open array parameter), что означает, что в эти функции можно передать как статический массив, так и динамический (установив предварительно его длину с помощью `SetLength()`). При этом, так как массивы Delphi содержат в себе длину, то в функцию L502_GetSerialList() отдельно передавать размер массива не требуется. Однако в функциях для работы с данными (например, L502_Recv()) длина передается так же как и в функциях C, чтобы можно было использовать для приема не обязательно весь массив. При этом дополнительно проверяется, что переданная отдельным параметром длина не превышает реальную длину массива. В случае превышения будет возвращена ошибка L502_ERR_INSUFFICIENT_ARRAY_SIZE.

2.3 Использование библиотеки в проекте на C#

Для написания программ, работающих с платой L502, на языке C# (или на любом другом, поддерживаемым NetFramework), реализована специальная библиотека-обертка lpcieNet.dll. Она использует библиотеку C l502api, в которой реализована вся логика работы с устройством. Установщик позволяет установить lpcieNet.dll в системный кэш (GAC), что позволяет не копировать библиотеку вместе с Вашим проектом. Однако, к сожалению, Visual Studio не позволяет добавлять в проект ссылки из системного кеша и Вам все равно придется саму ссылку делать на локальную копию (просто ее не обязательно будет распространять вместе с проектом). Библиотека в кеше имеет преимущество перед локальной и будет всегда использованна именно она, если установлена.

Для использования библиотеки достаточно добавить в проект ссылку на lpcieNet.dll и в исходниках подключить нужные пространства имен:

```
using l502api;
using lpcieapi;
```

По сравнению с функциями языка C в обертке C# сделаны следующие изменения:

- Так как C# — объектно ориентированный язык, то для управления модулем L502 создан специальный класс L502.
- Все функции, которые принимают описатель модуля первым параметром реализованы методами класса L502, при этом сам префикс L502_ не используется. Например, вместо `L502_Open(hnd, serial)`, используется `hnd.Open(serial)`.
- Функции L502_Create() и L502_Free() вызывается в конструкторе и деструкторе класса L502, отдельными функциями не реализованы.
- Функции, которые не требуют экземпляра модуля (не принимают описатель первым параметром), реализованы как статические функции класса L502. Например не `L502_GetErrorString(err)`, а `L502.GetErrorString(err)`.
- Функции типа Get/Set, которые принимают описатель модуля и один параметр, реализованы в виде свойств (properties) класса L502. Например, вместо `L502_SetLChannelCount(hnd, value)`, используется `hnd.LChannelCount = value`. Однако следует быть внимательным, так как неправильно устанавливаемое значение вызовет исключение `L502.Exception`.
- Константы объявлены внутри класса L502 и без префикса L502_
- Перечисления также объявлены как перечисления внутри класса L502 и без префикса `L502_ПЕРЕЧИСЛЕНИЕ`. Например не L502_SYNC_INTERNAL, а `L502.Sync.INTERNAL`.
- Коды ошибок, так как планируется использовать общие коды ошибок и для будущих модулей на шине PCI-Express, вынесены в перечисление ERR в классе lpcie.
- Все функции, использующие в C строки в виде `char *`, используют в обертке строки типа `String`.
- L502.GetSerialList() возвращает созданный внутри функции динамический массив строк (а не заполняет переданный), который уже содержит длину. Поэтому дополнительный параметр размера массива не требуется.
- Так же как и в Delphi при работе с массивами данных передается длина дополнительным параметром, так как можно принимать меньше данных чем в выделенном массиве.
- Функции, принимающие указатели в C, принимают параметры со спецификаторами out или ref, в зависимости от того, должна ли быть переменная проинициализирована перед вызовом функции или является выходным параметром.

Пример программы на C# можно посмотреть в examples/cs "L-Card PCI Express SDK".

2.4 Использование библиотеки в проекте LabView

Вы можете управлять модулем L502 из LabView используя тот факт, что LabView поддерживает управляемые библиотеки NetFramework. Соответственно Вы имеете доступ ко всем функциям, которые реализует оболочка C# lpcieNet.dll (т.е. все доступные функции). В отличие от Visual Studio LabView автоматически подхватывает .Net библиотеки из системного кэша (GAC) и Вы можете ссылаться на нее, а не хранить локальную копию вместе с программой.

Для работы с классами .Net в LabView есть специальная панель **Connectivity -> .Net** .

Вам необходимо использовать следующие блоки:

- **Constructor Node** - создает объект. Должен быть создан для каждого модуля L502, с которым будете работать. При создании LabView предложит выбрать библиотеку и класс (нужно выбрать lpcieNet.dll и L502). Одним из выходов этого блока является ссылка на объект, которая используются как вход для остальных блоков для работы с модулем. Также с помощью конструктора создается объект логического канала со всеми настройками.

- **Close Reference** - закрывает и удаляет объект. Должен вызываться для каждого созданного объекта L502 по завершению работы.

- **Invoke Node** - вызов функции (метода класса). При работе с объектом на вход подается ссылка и входные параметры, а на выходе — выходные параметры и обновленная ссылка (которая должна использоваться для блоков, которые будут вызваны после текущего). Входная ссылка и определяет, методы какого объекта используются (после заведения ссылки на вход имя класса появится в верхней строке, а при нажатии на вторую будет предложен на выбор его метод). Для функций, которые не работают с конкретным объектом (GetErrorString, GetSerialList — эти функции статические и при выборе помечены [S] в начале) ссылку на вход подавать не обязательно. Однако они все равно принадлежат классу, который надо выбрать нажав правой кнопкой на блок и далее **Select Class/.Net**

- **Property Node** - используется для установки или получения свойств. Через свойства устанавливается часть параметров (логическая таблица, режим синхронизации) и также можно получить информацию о модуле (в виде класса, каждое поле которого является также отдельным свойством). Можно устанавливать несколько свойств одним блоком, расширив его вниз. Также с помощью свойств можно задавать константы из перечислений (что может быть более понятно, чем просто подавать на вход числа) - в этом случае надо выбрать класс перечисление, а каждому значению соответствует свое свойство.

Пример программы на LabView находится в examples/LabView `"L-Card PCI Express SDK"`.

2.5 64-битная версия библиотеки

На 64-биной версии Windows могут выполняться программы, как собранные 32-битным, так и 64-битным компилятором, поэтому большинство программ для Windows

существует только в 32-битном варианте. 64-битный компилятор используют как правило для программ, работающих с большим количеством данных, так как это позволяет иметь процессу виртуальное пространство больше 4 Гбайт.

Для 64-битной Windows установщик "L-Card PCI Express SDK" ставит как 32-битную версию библиотеки, так и 64-битную. Первая ставится в директорию Windows/SysWOW64, где хранятся 32-битные библиотеки, а вторую в Windows/-Sysnative, где соответственно лежат системные 64-битные библиотеки. При этом директория Windows/system32 указывает на одну из этих директорий в зависимости от разрядности самого приложения, которое обращается по указанному пути.

Таким образом, выбор библиотеки происходит автоматически, в зависимости от разрядности приложения. Единственным отличием при написании программ на C/C++ является необходимость подключить lib-файл в соответствии с разрядностью используемого компилятора.

Для программ на Delphi необходимо только указать для какой платформы будет собираться проект (win32 или win64) и собранная программа будет использовать библиотеку l502api.dll для которой программа была скомпилирована.

Программы на языке C# компилируются в машинный код при выполнении. При этом один раз созданная программа может выполняться как на 32-битной версии, так и на 64-битной версии виртуальной машины NetFramework (в проекте можно указать явно для какой разрядности NetFramework предназначена программа). Таким образом, одна и та же программа в 32-битной версии Windows будет выполняться поверх 32-битной версии l502api.dll, а в 64-битной - поверх 64-битной версии l502api.dll. Для библиотеки .Net разрядность определяется разрядностью использующего его приложения.

Соответственно, в проекте на LabView, который использует .Net библиотеку, разрядность используемой библиотеки определяется разрядностью используемой Вами среды.

2.6 Установка библиотеки и драйвера для ОС Linux

Для установки драйвера и библиотеки под ОС Linux у Вас есть два варианта:

- Воспользоваться готовыми собранными пакетами, предоставляемыми "Л Кард". Это рекомендованный способ для дистрибутивов, для которых предоставляются собранные пакеты. Список поддерживаемых дистрибутивов можно посмотреть в документе "Использование внешних репозиториев L Card для дистрибутивов Linux"

- Скачать исходники SDK и собрать их самостоятельно (подробнее в следующем разделе).

Для примеров работы с модулем L502 на C под Linux Вы можете скачать архив с исходниками SDK и посмотреть на примеры в директории examples/msvc. Несмотря на название, они могут быть собраны GCC под ОС Linux. Для каждого примера есть makefile (с комментариями), а также файл CMakeList.txt для предпочитающих сборку с использованием cmake .

О том, как подключить внешний репозиторий, установить собранные пакеты и о преимуществах данного метода установки описано в документе "Использование внешних репозиториев L Card для дистрибутивов Linux". Здесь же будет приведен список самих пакетов, использующихся при работе с модулем L502.

Для работы с L502 используются пакеты:

- **libl502api1-dev** или **libl502api1-devel** — Пакет с файлами l502api для разработчика: заголовочные файлы и линк на библиотеку нужной версии. Нужен при написании своих программ с использованием l502api (зависит от libl502api1)

- **libl502api1** — Пакет непосредственно с библиотекой l502api нужной версии. Если вы распространяете свою программу, которая использует l502api, то Вам достаточно включить в зависимости только этот пакет (без пакета с файлами для разработчика)

- **lpcie-dkms** — Пакет с исходниками драйвера (модуля ядра), использующий систему сборки внешних модулей dkms (подробнее описано ниже).

- **lpcie-firm-update** — Утилита для обновления прошивки ПЛИС модуля L502. Пакет включает в себя последнюю версию прошивки ПЛИС и скрипт lpcie-firm-update-all.sh для обновления прошивок всех найденных устройств L502.

Так как драйвер должен быть собран под конкретную версию ядра (а ядро может обновляться в одной версии дистрибутива или даже могут использоваться разные варианты ядра), то драйвер не может распространятся в уже собранном виде. Его сборка выполняется непосредственно при установке пакета. При этом необходимо предварительно поставить пакет с заголовочными файлами текущего ядра (обычно в пакетах с именами linux-headers или kernel-devel). Для некоторых дистрибутивов может быть несколько вариантов ядра (и соответственно несколько пакетов), более того Вы можете использовать свое ядро. Именно по этой причине пакет не задан зависимостью для lpcie-dkms, в отличие от других зависимостей. Узнать текущую версию ядра можно командой: `uname -r`. Убедиться, что нужные файлы установлены можно проверив наличие файлов в директории /lib/modules/‘uname -r‘/build (обычно это ссылка на заголовки ядра в /usr/src/linux-<version> или /usr/src/kernels/<version>).

Если заголовки текущего ядра установлены, то при установке пакета lpcie-dkms будет выполнена сборка драйверов с использованием DKMS (пакет dkms входит в зависимости lpcie-dkms, как и make и gcc, необходимые для сборки). DKMS это достаточно широко распространенная система сборки и управления внешними модулями ядра (она находится в основном репозитории большинства дистрибутивов Linux, хотя ее нет в openSuse, но для нее распространяется пакет dkms через тот же Open Build System).

DKMS позволяет:

- централизованно отслеживать, какие сторонние модули ядра, какие их версии и для каких версий ядра установлены (dkms status)

- хранит всегда исходники драйвера разных версий в централизованном месте (/usr/src/lpcie-<version>)

- позволяет автоматически пересобирать драйвер при переходе на новое ядро

- позволяет в любой момент удалить драйвер или любую его версию и все связанные с ним файлы (dkms remove -m lpcie -v <version> –all)

Таким образом, хотя в пакете находятся исходники драйвера, а не собранный драйвер, установка мало чем отличается от установки других пакетов, кроме того, что требует дополнительной установки заголовочных файлов ядра и установка пакета требует значительного времени на сборку.

При установки нового ядра модуль будет пересобран под него автоматически либо при установке пакета, либо при первом входе в систему с новым ядром.

2.7 Исходные коды SDK

Исходные коды всех составных компонентов SDK открыты. Пользователю предоставляется доступ на чтение к репозиторию системы контроля версий Mercurial, расположенному по адресу https://bitbucket.org/lcard/lpcie_sdk. Подробнее об использовании открытых репозиториев исходных кодов "L Card" на bitbucket.org Вы можете прочитать в документе "Использование открытых репозиториев исходных кодов L Card на bitbucket.org".

Вы также можете скачать архив lpcie_sdk_src.zip со всеми исходными кодами из прикрепленных файлов проекта репозитория.

Инструкции по сборке находятся в файле исходников INSTALL.txt.

Глава 3

Общий подход к работе с библиотекой

3.1 Общий алгоритм для работы с модулем.

Данный раздел описывает типичную последовательность вызова функций для работы с модулем L502. Более подробно каждый шаг будет описан в последующих разделах.

Типичная последовательность вызовов имеет следующий вид:

1. Получение списка серийных номеров модулей L502, присутствующих в системе, с помощью функции L502_GetSerialList().
2. Если в системе присутствует нужный модуль L502, создаем описатель модуля с помощью L502_Create().
3. Установить соединение с модулем, передав созданный описатель и серийный номер выбранного устройства в функцию L502_Open().
4. При необходимости, получить дополнительную информацию о устройстве с помощью L502_GetDevInfo() (в частности для проверки наличия сигнально процессора BlackFin).
5. При наличии сигнального процессора (и желании работать с его использованием) загрузить прошивку сигнального процессора с помощью L502_BfLoadFirmware().
6. Установка параметров модуля с помощью набора функций для изменения настроек модуля (названия функций начинаются с L502_Set)
7. Передача установленных параметров в модуль с помощью L502_Configure().
8. Работа с модулем в синхронном и/или асинхронном режиме (описана в последующих подразделах).
9. Закрытие модуля L502_Close().
10. Освобождение описателя модуля функцией L502_Free().

3.1.1 Работа с модулем при синхронном вводе

Типичная работа с модулем при синхронном вводе состоит из следующих шагов:

1. Разрешение нужных синхронных потоков (АЦП и/или цифровых данных) с помощью L502_StreamsEnable().

2. Запуск синхронных потоков L502_StreamsStart().
3. Чтение принятых данных из модуля с помощью L502_Recv().
4. Обработка прочитанных данных с помощью L502_ProcessData().
5. При необходимости приема и обработки следующего блока, переход к пункту 3.
6. Останов синхронных потоков с помощью L502_StreamsStop().

3.1.2 Работа с модулем при синхронном потоковом выводе

Типичная работа с модулем при синхронном выводе состоит из следующих шагов:

1. Разрешение нужных синхронных потоков (каналы ЦАП, цифровые выходы) с помощью L502_StreamsEnable().
2. Запуск предварительной загрузки данных на вывод с помощью L502_PreloadStart().
3. Подготовка блока данных на запись с помощью L502_PrepareData().
4. Запись подготовленного блока в модуль с помощью L502_Send().
5. Запуск синхронных потоков вызовом L502_StreamsStart().
6. Выполнение пунктов 4. и 5. при необходимости подгрузить новые данные.
7. Останов синхронных потоков с помощью L502_StreamsStop().

3.1.3 Работа с модулем при циклическом выводе выводе

Для выставления циклического сигнала без подкачки типичная последовательность выглядит так:

1. Разрешение нужных синхронных потоков (каналы ЦАП, цифровые выходы) с помощью L502_StreamsEnable().
2. Выделение циклического буфера указанного размера с помощью L502_OutCycleLoadStart().
3. Загрузка данных указанного размера для циклического вывода с помощью одного или нескольких вызовов L502_Send().
4. Сделать загруженный сигнал активным с помощью L502_OutCycleSetup().
5. Запустить синхронный ввод-вывод через L502_StreamsStart().
6. При необходимости вывести новый сигнал выполнить шаги 2.-4.
7. По завершению работы остановить синхронный ввод-вывод с помощью L502_StreamsStop() или только циклический вывод через L502_OutCycleStop().

3.1.4 Работа с модулем при асинхронном вводе-выводе

Типичная работа при асинхронном вводе-выводе состоит из вызова одной из функций:

- L502_AsyncInDig() - асинхронный ввод значений цифровых линий
- L502_AsyncOutDig() - асинхронный вывод значений на цифровые линии
- L502_AsyncOutDac() - асинхронный вывод значения на один из каналов ЦАП
- L502_AsyncGetAdcFrame() - асинхронный прием одного кадра АЦП

3.2 Создание и освобождение описателя модуля.

Вся работа с модулем L502 осуществляется через описатель модуля типа t_l502-_hnd. Описатель модуля представляет собой непрозрачный указатель на структуру, которая хранит всю информацию о модуле и состоянии соединения с ним. Пользователь не имеет прямого доступа к полям структуры и все действия с модулем выполняются посредством вызова соответствующих функций библиотеки, которые принимают описатель модуля в качестве первого параметра.

Перед попыткой установить связь с модулем необходимо создать описатель, вызвав функцию L502_Create(), которая выделяет память под структуру, инициализирует ее поля значениями по-умолчанию и возвращает указатель на нее - описатель модуля.

После того как работа с модулем завершена, выделенная функцией L502_Create() память должна быть освобождена посредством вызова L502_Free(). После освобождения описатель уже не может использоваться.

3.3 Открытие связи с модулем.

Для начала работы с модулем необходимо установить с ним связь с помощью функции L502_Open(). Для различия модулей используется их серийные номера.

Получить список серийных номеров всех найденных модулей L502 можно с помощью функции L502_GetSerialList(). Данная функция принимает плоский массив, в который будут сохранены найденные серийные номера, и максимальное количество серийных номеров, которое можно сохранить в переданный массив.

В простейшем случае можно задать максимальное значение модулей и для серийных номеров использовать статически выделенный массив:

```
#define MAX_MODULES_CNT  16

char serial_list[MAX_MODULES_CNT][L502_SERIAL_SIZE];
int32_t get_list_res = L502_GetSerialList(serial_list, MAX_MODULES_CNT, 0, NULL);
if (get_list_res<0)
{
    /* Ошибка получения списка серийных номеров */
}
else if (get_list_res==0)
{
     /* Не найдено ни одного модуля */
```

```
}
else
{
     /* Найдено get_list_res модулей */
}
```

В общем случае для работы с произвольным максимальным количеством модулей можно воспользоваться третьим параметром функции для получения количества найденных модулей в системе. При этом в качестве массива серийных номеров можно передать нулевой указатель и указать нулевой размер массива. После этого можно динамически выделить массив под полученное количество серийных номеров и повторно вызвать L502_GetSerialList() для получения серийных номеров всех модулей L502:

```
uint32_t dev_cnt;
int32_t res;

/* Получаем количество модулей в системе */
res  = L502_GetSerialList(NULL, 0, 0, &dev_cnt);
if (res<0)
{
     /* Ошибка получения списка серийных номеров */
}
else if (dev_cnt==0)
{
     /* Не найдено ни одного модуля */
}
else
{
    /* Выделяем плоский массив под dev_cnt серийных номеров размером
       dev_cnt*L502_SERIAL_SIZE */
    t_l502_serial_list serial_list=
        (t_l502_serial_list)
         malloc(dev_cnt*L502_SERIAL_SIZE);
    if (serial_list==NULL)
    {
         /* Ошибка выделения памяти */
    }
    else
    {
         res = L502_GetSerialList(serial_list, dev_cnt, 0,
                                  NULL);
         if (res>0)
         {
             /* Получено res серийных номеров */
         }
         /* Освобождаем выделенный массив под серийные номера */
         free(serial_list);
    }
}
```

Следует отметить, что с одним модулем одновременно может быть установлено только одно соединение. При попытке открыть модуль, с которым уже установлено соединение через другой описатель (возможно, в другой программе) L502_Open()

вернет L502_ERR_DEVICE_ACCESS_DENIED. При этом L502_GetSerialList() по-умолчанию возвращает список всех серийных номеров модуля, включая те, с которыми уже установлено соединение. Если нужно получить список только тех устройств, с которыми еще не установлено соединения, то в L502_GetSerialList() можно передать флаг L502_GETDEVS_FLAGS_ONLY_NOT_OPENED.

Если в качестве серийного номера в L502_Open() передать нулевой указатель или пустую строку, то будет предпринята попытка открыть первый модуль, с которым удастся успешно установить соединение. Если ни с одним модулем установить соединение не удалось, то будет возвращена ошибка, полученная при попытки открыть последний модуль. То есть, при наличии двух модулей L502 в системе, первый вызов L502_Open() установит соединение с первым модулем L502, второй вызов - со вторым, а третий вернет уже ошибку доступа L502_ERR_DEVICE_ACCESS_DENIED.

3.4 Режимы работы с сигнальным процессором и без него

Модуль L502 может работать в двух режимах:

- В штатном режиме (L502_MODE_FPGA) вся обработка данных выполняется аппаратно в ПЛИС модуля, а управление модулем осуществляется путем прямой записи значений в регистры ПЛИС. В этом режиме доступны все штатные функции сбора данных, однако у пользователя нет возможности расширить функциональные возможности самого модуля. Этот режим доступен для всех модификаций L502.

- В режим работы с сигнальным процессором (L502_MODE_DSP) все управление сбором данных выполняется сигнальным процессором BlackFin и все потоки данных на ввод и вывод идут через него. Таким образом, пользователь может путем создания модифицированной прошивки BlackFin реализовать дополнительные возможности (например, обратную связь в режиме реального времени). Этот режим доступен только для модификаций L-502-P-G м L-502-P-G-D.

При включении питания, модуль всегда находится в штатном режиме работы без использования сигнального процессора. Для работы сигнального процессора необходимо предварительно загрузить в него программу (прошивку). Это можно сделать из файла формата ldr с помощью функции L502_BfLoadFirmware(). После этого, будет автоматически переведена в режим с сигнальным процессором.

При необходимости, можно специально переключить режим работы с помощью L502_SetMode(). Это может потребоваться, например, если прошивка загружена в BlackFin по интерфейсу JTAG. Кроме того, следует иметь ввиду, что при открытии связи с устройством, не производится изменение режима работы модуля. Т.е. если одна программа установила режим L502_MODE_DSP, то при последующем открытии модуля из другой программы этот режим сохранится. В этой связи может понадобиться явно перевести модуль в штатный режим с помощью L502_SetMode().

Все настройки модуля и работа с синхронным вводом-выводом должны выполняться после установки нужного режима.

В любой момент можно узнать текущий режим работы с помощью L502_GetMode().

Программно проверить наличие сигнального процессора можно, получив информацию о модуле с помощью L502_GetDevInfo() и проверив, установлен ли флаг L502_DEVFLAGS_BF_PRESENT.

3.5 Установка настроек модуля

Перед использованием модуля, как правило, необходимо выполнить настройку его параметров. Сперва все настройки записываются в поля структуры описателя модуля с помощью функций, начинающихся с L502_Set, которые будут описаны в последующих подразделах, после чего установленные параметры передаются в модуль с помощью L502_Configure().

3.5.1 Настройка последовательности опроса каналов АЦП

L502 является модулем с последовательной коммутацией каналов. То есть измерение нескольких каналов происходит последовательно, путем переключения входного коммутатора АЦП. Как и в большинстве моделей L-Card последовательность опроса каналов задается с помощью управляющей таблицы логических каналов АЦП. Всего таблица может содержать от одного до L502_LTABLE_MAX_CH_CNT логических каналов.

Каждый логический канал задается следующие параметры:

- номер физического канала, с которого производится измерение. Номер физического канала задается, считая от 0, то есть 0 означает первый канал, 1 – второй и т.д. Таким образом, в дифференциальном режиме номер канала может быть от 0 до 15, а в режиме измерения с общей землей – от 0 до 31.
- режим измерения АЦП из t_l502_lch_mode.
- используемый диапазон измерения (из t_l502_adc_range).
- коэффициент усреднения по заданному логическому каналу (см. раздел Коэффициент усреднения для логического канала).

Задать параметры логического канала с нужным номером можно с помощью функции L502_SetLChannel(), а количество логических каналов в управляющей таблице - с помощью L502_SetLChannelCount().

Например, необходимо измерить сперва напряжение входа X1 относительно общей земли для диапазона +/-10В, затем измерить значение на 16 канале в дифференциальном режиме (между входами X16 и Y16) с диапазоном +/-1В, а затем измерить напряжение между Y1 и общей землей (17 канал в режиме с общей землей) с диапазоном +/-0.2В (Назначение выводов сигнального разъема и подключение сигналов к модулю описано в "Руководстве пользователя"). В этом случае настройка логической таблицы будет выглядеть следующим образом:

```
/* устанавливаем 3 логических канала */
int32_t err = L502_SetLChannelCount(hnd, 3);
if (!err)
{
    /* первый логический канал соответствует измерению 1 канала
       относительно общей земли */
    err = L502_SetLChannel(hnd,0,0,L502_LCH_MODE_COMM,
      L502_ADC_RANGE_10,0);
}
if (!err)
{
```

```
    /* второй логический канал соответствует измерению 16 канала
       в диф. режиме */
    err = L502_SetLChannel(hnd,1,15,L502_LCH_MODE_DIFF,
     L502_ADC_RANGE_1, 0);
}
if (!err)
{
    /* третий логический канал - измерение 17-го канала
       относительно общей земли */
    err = L502_SetLChannel(hnd,2,16,L502_LCH_MODE_COMM,
      L502_ADC_RANGE_02, 0);
}

if (!err)
{
    /* установка других настроек */

}
if (!err)
{
    /* передаем настройки в модуль */
    err = L502_Configure(hnd,0);
}
if (err)
{
    /* произошла ошибка при настройке параметров... */
}
```

После завершения измерения с настройками, соответствующими последнему логическому каналу, следует измерение соответствующее снова первому (с нулевым номером) логическому каналу. Последовательность измерений соответствующая одному проходу логической таблицы называется кадром.

При желании, между завершением измерения соответствующего последнему логическому каналу кадра и первому логическому каналу следующего кадра может быть вставлена межкадровая задержка.

3.5.2 Настройка частоты синхронного ввода/вывода

Все частоты потокового сбора и выдачи данных основываются на опорной частоте синхронизации. В качестве опорной частоты может использоваться внутренний источник частоты или внешний. В первом случае, опорная частота может быть 2МГц либо 1.5МГц. По умолчанию используется 2МГц. Изменить ее можно с помощью функции L502_SetRefFreq().

Частота сбора АЦП получается с помощью деления значения опорной частоты на установленный коэффициент, который может быть в диапазоне от 1 до L502_ADC_FREQ_DIV_MAX. Кроме того, как уже упоминалось в предыдущем разделе, между измерением последнего логического канала одного кадра и началом следующего кадра, может быть добавлена межкадровая задержка. Межкадровая задержка задается в виде количества периодов опорной частоты синхронизации.

Делитель частоты сбора АЦП и количество периодов опорной частоты для межкадровой задержки можно задать явно с помощью функций L502_SetAdcFreqDivider()

и L502_SetAdcInterframeDelay() соответственно. Вместо этих функций для удобства можно использовать функцию L502_SetAdcFreq(), которой можно передать значения частоты сбора АЦП и частоты кадров в Герцах, а функция сама рассчитает нужный делитель и значение межкадровой задержки, чтобы полученные частоты были наиболее близки к указанным. При этом функция вернет реально установившиеся значения частот.

Под частотой сбора АЦП (f_acq) понимается величина, обратная времени одного преобразования, соответствующего одному логическому каналу. Под частотой кадров (f_frame) понимается величина, обратная времени от начала измерения первого логического канала одного кадра до начала измерения первого логического канала следующего кадра. Это частота соответствует частоте сбора для одного логического канала.

Ниже приведена диаграмма, иллюстрирующая на примере сбора при заданных трех логических каналах, как определяются упомянутые выше частоты.

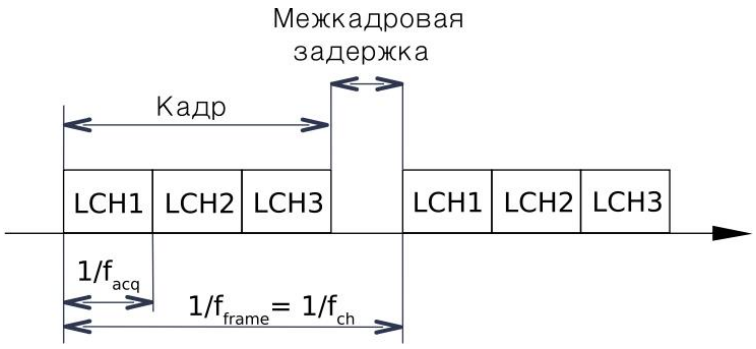

Рис. 3.1: Диаграмма сбора АЦП для трех логических каналов

Если межкадровая задержка не нужна, то можно передать нулевой указатель в качестве второго параметра L502_SetAdcFreq(), тогда будет использоваться всегда нулевая межкадровая задержка (т.е. измерение следующего кадра начнется сразу после завершения предыдущего).

Помимо синхронного ввода с АЦП модуль L502 позволяет осуществлять синхронный ввод с 18 цифровых входов. Так же как и для синхронного сбора данных АЦП, частота синхронного цифрового ввода определяется как опорная частота, разделенная на коэффициент, который можно установить с помощью L502_SetDinFreqDivider(). Так же можно вызывать функцию L502_SetDinFreq(), чтобы она рассчитала этот коэффициент для получения наиболее близкой частоты к указанной. Для синхронного ввода цифровых линий нет ни логической таблицы (так как каждый раз вводятся все 18 линий), ни межкадровой задержки - при запуске синхронного ввода все измерения выполняются через одинаковые промежутки времени.

Плата L502 так же позволяет осуществить синхронный вывод на два канала ЦАП и цифровые выводы. При этом вывод каждого канала осуществляется с частотой в два раза меньшей значения опорной частоты. Соответственно, частоты вывода специально настраивать не нужно.

3.5.3 Коэффициент усреднения для логического канала

Реально микросхема АЦП всегда работает на частоте равной опорной частоте синхронизации. В случае, если частота сбора АЦП установлена меньше, чем опорная частота синхронизации, то на одно измерение значения логического канала приходится

n измерений АЦП (n = f_acq/f_ref - отношение установленной частоты сбора АЦП к опорной частоте дискретизации).

При отключенном усреднении, первые n-1 измерений отбрасывается, что увеличивает время установления сигнала. При необходимости можно использовать несколько последних отсчетов (navg) для получения результирующего значения. Тогда результирующего значение будет являться средним между navg последними измерениями, однако это сокращает соответственно время на установления сигнала. Естественно navg всегда меньше либо равно n. Кроме того navg не может превышать максимального значения, равного L502_LCH_AVG_SIZE_MAX.

Значение navg задается последним параметром функции L502_SetLChannel(). Значение равное 1 означает отсутствие усреднения. Значения равное 0, означает что коэффициент усреднения может быть выбран по-усмотрению библиотеки. В текущей реализации значение 0 аналогично значению 1, но это может быть изменено в последующих версиях.

3.5.4 Настройка режимов синхронизации

По умолчанию в качестве опорной частоты синхронизации используется внутренняя частота модуля, а запуск всех синхронных измерений осуществляется при выполнении функции L502_StreamsStart().

Однако, при необходимости, возможно задать как внешний источник опорной частоты, так и внешний сигнал запуска синхронного сбора/выдачи данных.

Для этого могут быть использованы входы цифрового разъема DI_SYN1 и DI_SYN2 (может использоваться как фронт, так и спад одного из этих сигналов), либо также может использоваться разъем синхронизации для организации синхронного сбора данных по принципу ведущий-ведомые.

Выбор внешнего сигнала для задания опорной частоты синхронизации задается с помощью L502_SetSyncMode(), а условие запуска с помощью функции L502_SetSyncStartMode(). Следует отметить, что если задано внешнее событие запуска, то для того, чтобы модуль перешел в режим ожидания этого события, необходимо вызвать L502_StreamsStart().

Останов синхронного сбора/выдачи данных всегда осуществляется программно с помощью L502_StreamsStop().

При использовании разъема синхронизации для организации сбора данных по принципу ведущий-ведомые, для ведущего модуля источником опорной частоты синхронизации остается внутренняя частота (режим L502_SYNC_INTERNAL), а каждый ведомый модуль использует опорную частоту и/или признак запуска сбора от внешнего мастера, т.е. для каждого ведомого модуля должен быть установлен режим L502_SYNC_EXTERNAL_MASTER.

3.6 Синхронный и асинхронный режимы работы.

Для модуля L502 доступны следующие данные на ввод:

- отсчеты с АЦП
- значения цифровых входов

Также модуль может быть использована для вывода:

- отсчетов на первый канал ЦАП
- отсчетов на второй канал ЦАП
- значений на цифровые выходы

Таким образом, имеется 2 канала на ввод и 3 канала на вывод.

Каждый из этих каналов может работать как в синхронном режиме, так и асинхронно. При этом каждый канал может быть настроен индивидуально, то есть можно выполнять, например, асинхронный ввод цифровых линий на фоне синхронного потокового сбора с АЦП или выводить на один канал ЦАП сигнал в синхронном потоковом режиме, в то время как в другой выставлять значения асинхронно. Единственное исключение - невозможно осуществить асинхронный ввод с АЦП на фоне синхронного сбора данных с цифровых входов.

3.6.1 Асинхронный режим работы

При включении питания все каналы находятся в асинхронном режиме. В асинхронном режимы при вызове функции асинхронного ввода/вывода производится однократный ввод или вывод указанной информации. При этом задержка от вызова функции до непосредственно момента измерения данных для ввода или выставления указанного значения на выходе для вывода точно не определена. Также точно не может быть определена задержка между двумя последовательными операциями ввода/вывода.

Плюсом асинхронного режима является простота его использования - достаточно одного вызова требуемой функции:

- L502_AsyncInDig() - асинхронный ввод значений цифровых линий
- L502_AsyncOutDig() - асинхронный вывод значений на цифровые линии
- L502_AsyncOutDac() - асинхронный вывод значения на один из каналов ЦАП

Для однократного ввода данных с АЦП используется функция L502_AsyncGetAdcFrame(), которая выполняет ввод одного кадра данных АЦП. В отличии от других функций асинхронного ввода-вывода, перед ее вызовом данной функции необходимо выполнить настройку модуля: необходимо задать управляющую таблицу АЦП (см. Настройка последовательности опроса каналов АЦП). Измерение логических каналов внутри кадра происходит синхронно с заданной частотой сбора АЦП. Асинхронным является ввод самих кадров, то есть задержка между измерением кадров при последовательном вызове L502_AsyncGetAdcFrame() не определена.

Например, код для выполнения однократного измерения с 7-го физического канала в дифференциальным режиме с диапазоном +/-0.5В может выглядеть следующим образом:

```
/* устанавливаем 1 логический канал в управляющей таблице */
int32_t err = L502_SetLChannelCount(hnd, 1);
if (!err)
{
    /* логический канал соответствует измерению 7 канала
        в диф. режиме */
    err = L502_SetLChannel(hnd,0,6,L502_LCH_MODE_DIFF,
      L502_ADC_RANGE_05,0);
```

```
}
if (!err)
{
    /* передаем настройки в модуль */
    err = L502_Configure(hnd,0);
}
if (!err)
{
    /* Считываем кадр данных АЦП из одного отсчета */
    double val;
    err = L502_AsyncGetAdcFrame(hnd,
            L502_PROC_FLAGS_VOLT, 1000, &val);
    if (!err)
    {
        /* верно считали значение val */
    }
}
```

3.6.2 Синхронный режим работы

В синхронном режиме ввод или вывод данных осуществляется с заданной частотой, то есть время между соседними измерениями или выводом соседний отсчетов определено. Частоты сбора для каждого канала задаются относительно общей опорной частоты синхронизации (подробнее см. главу "Настройка частоты синхронного ввода/вывода") и запуск синхронного ввода-вывода для всех каналов осуществляется одновременно.

Для запуска синхронного режима, необходимо сперва с помощью функции L502_StreamsEnable() разрешить синхронный режим по требуемым каналам, а затем запустить синхронный ввод/вывод по всем разрешенным каналам с помощью L502_StreamsStart().

При синхронном вводе модуль производит измерения с заданной частотой и сам передает данные по интерфейсу PCI Express в буфер драйвера (с использованием Bus-Master DMA). Принятые в буфер данные могут быть прочитаны программой с помощью L502_Recv().

Аналогично, для синхронного вывода, модуль сам по мере необходимости считывает данные из буфера драйвера и выводит считанные отсчеты с заданной частотой. Данные в буфер драйвера должны быть предварительно записаны с помощью L502_Send(). При этом, если к моменту вывода очередного данные в буфер драйвера не поступили, то будет выведено предыдущее значение.

В драйвере выделяется всего два буфера — один на прием и один на передачу. То есть значения для синхронного ввода с цифровых линий и отсчеты АЦП передаются одним потоком, также одним потоком передаются все данные на вывод. Каждый отсчет передается в виде 32-битного слова, содержащего дополнительную информацию, включающую в себя признак, к какому типу данных относится данный отсчет.

Разбор принятых данных на отсчеты АЦП и значения цифровых выводов осуществляется с помощью L502_ProcessData(). Помимо этого, данная функция также может осуществить перевод отсчетов АЦП в Вольты. Следует учесть, что в отличие от других изделий L-Card, применение калибровочных коэффициентов осуществляется аппаратно и значения уже приходят в виде 24-битных отсчетов с уже примененными коэффициентами.

В 32-битном слове соответствующем отсчету АЦП передается дополнительно режим

измерения и номер физического канала. L502_ProcessData() сравнивает эти значения теми которые были заданы при настройке управляющей таблицы, чтобы убедиться в корректности принимаемых данных. При этом L502_ProcessData() ожидает что с ее помощью будут обрабатываться все принятые данные.

Данные от АЦП приходят в том порядке, в котором производятся измерения, т.-е. сперва измерения соответствующие всем логическим каналам первого кадра, затем второго и т.д. В этом же порядке L502_ProcessData() возвращает и преобразованные отсчеты АЦП. При этом в L502_ProcessData() можно передавать и нецелое количество кадров (например, если запущен синхронный ввод с цифровых линий, то заранее сложно определить, сколько в принятом блоке данных содержится отсчетов с цифровых линий, а сколько отсчетов с АЦП), в этом случае L502_ProcessData() обработает и выдает все отсчеты, включая отсчеты нецелого кадра, а при следующем ее вызове проверяет, что отсчеты АЦП начинаются с логического канала, следующего за последним обработанным до этого каналом. Какой логический канал ожидается следующим для обработки можно узнать с помощью функции L502_GetNextExpectedLchNum().

Например, пусть в управляющей таблице АЦП установлено 7 логических каналов. Если был принят блок данных от модуля содержащий 5 отсчетов АЦП (и произвольное количество значений цифровых входов, если включен синхронный ввод с цифровых линий) и обработан с помощью L502_ProcessData(), то L502_ProcessData() вернет преобразованные 5 отсчетов АЦП, соответствующие логическим каналом с индексами 0,1,2,3,4. Следующий логический канал, который ожидается для обработки - с индексом 5, поэтому L502_GetNextExpectedLchNum() вернет значение 5. Если обработать следующий блок данных, содержащий 5 следующих отсчетов, то L502_ProcessData() вернет отсчеты соответствующие каналам с индексами 5,6,0,1,2. Т.е. с помощью вызова L502_GetNextExpectedLchNum() можно узнать, какому логическому отсчету будет соответствовать первый элемент выходного массива при последующем вызове L502_-ProcessData().

Следует учесть, что по-умолчанию буфер в драйвере рассчитан на количество отсчетов, которое будет введено за 4с непрерывного сбора. Если вовремя не считывать данные с помощью L502_Recv(), то произойдет переполнение буфера в драйвере и часть данных, для которых не нашлось в буфере места будет потеряна. При последующем появлении места в буфере, в то место в потоке, где произошел разрыв непрерывного потока данных будет вставлено слово, представляющее собой сообщение о переполнении буфера. Если L502_ProcessData() во входном массиве обнаружит это слово, то функция вернет ошибку L502_ERR_STREAM_OVERFLOW. При этом, как и в случае возникновения других ошибок обработки, все отсчеты, которые были до возникновения ошибки будут обработаны и возвращены в выходных массивах (размеры которых будут соответственно обновлены).

Аналогично, для формирования общего потока на вывод в требуемом формате используется функция L502_PrepareData(), принимающая данные из трех массивов и сохраняющая их во внешний массив. Если какой-либо из источников не должен использоваться, то в качестве массива передается нулевой указатель. В частности, нулевой указатель должен быть передан для каналов, которые не были настроены на синхронный режим с помощью L502_StreamsEnable().

Следует отметить, что если для синхронного ввода инициализация каналов DMA происходит по L502_StreamsStart(), так как данные начнут поступать только после запуска синхронного ввода, то с синхронным выводом дело обстоит несколько иначе. Так как по L502_StreamsStart() уже должна начаться выдача синхронных данных, то часть данных уже должна быть загружена в модуль. Таким образом, после разреше-

ния синхронного вывода по нужным каналам с помощью L502_StreamsEnable() и до запуска синхронного вывода с помощью L502_StreamsStart() необходимо осуществить предзагрузку части данных синхронного потока. Для этого следует вызвать функцию L502_PreloadStart(), по которой в драйвере будет выделен буфер на передачу и инициализирован канал DMA на передачу, а затем записать часть синхронных данных в буфер драйвера с помощью L502_PrepareData() и L502_Send(). Если этого не сделать, то синхронный вывод начнется лишь после того как данные будут записаны в модуль и не будет привязан к началу синхронного сбора/выдачи данных.

3.6.3 Циклический вывод

Начиная с версии 1.0.4 драйвера и библиотеки введена поддержка циклического вывода на ЦАП и цифровые выводы. Этот режим позволяет загрузить сигнал полностью в буфер внутри драйвера, содержимое которого будет циклически передаваться в модуль L502 без необходимости дальнейшей подкачки.

Данные для загрузки в циклический буфер подготавливаются также как и для потокового вывода с помощью L502_PrepareData() и записываются с помощью L502_Send() и могут содержать комбинацию данных на оба канала ЦАП и на цифровые выводы. Циклический вывод является вариантом синхронного вывода и для его работы нужно разрешить нужные потоки на вывод через L502_StreamsEnable() и должен быть запущен синхронный ввод-вывод через L502_StreamsStart(). Также как и с обычным потоковым выводом, часть каналов может использоваться для вывода циклического сигнала, а часть — асинхронно. Однако нельзя часть каналов вывода использовать в циклическом режиме, а часть в потоковом режиме с подкачкой (естественно, циклический режим на вывод можно использовать с потоковым на ввод).

Для вывода циклического сигнала в драйвере используется двойная буферизация - то есть может быть выделено два буфера, пока из одного сигнал циклически выводится, в другой может подгружаться следующий сигнал. Смена сигнала происходит по концу периода предыдущего. При этом после записи одного сигнала необходимо, чтобы успела пройти смена сигналов перед тем как можно будет загружать следующий.

Для загрузки сигнала сперва вызывается функция L502_OutCycleLoadStart(), в которой задается размер циклического буфера, который будет использован для хранения отсчетов всех используемых каналов вывода. Например, если нужно использовать два канала ЦАП, на каждый из которых вывести сигнал из 1000 точек, то размер должен быть указан 2000. После этого отсчеты загружаются как и при потоковом выводе с подкачкой с помощью функций L502_PrepareData()/L502_Send(). При этом суммарно должно быть записано ровно столько же отсчетов, сколько было указано при вызове L502_OutCycleLoadStart(). После загрузки по вызову L502_OutCycleSetup() происходит переключение на загруженный буфер, при этом в зависимости от текущего состояния это приводит к следующему:

- если синхронный ввод-вывод не запущен (не было вызова L502_StreamsStart()) то начинается предзагрузка циклического сигнала в плату, однако реально выдача сигнала начнется только при вызове L502_StreamsStart() (или по внешнему условию запуска). Это позволяет привязать начало вывода первого отсчета циклического сигнала к началу ввода.

- если синхронный ввод-вывод запущен, но не было выведено ни одного циклического сигнала до этого, то по L502_OutCycleSetup() начинается вывод циклического сигнала.

- если синхронный ввод-вывод запущен и уже выводится предыдущий циклический сигнал, то после вызова L502_OutCycleSetup() в драйвере будет взведен флаг о необходимости переключить сигналы. После этого события по достижению конца предыдущего циклического буфера будет произведена смена буферов вывода. Таким образом это позволяет производить смену сигнала всегда в известной точке. При смене сигнала происходит освобождение старого буфера и только после реальной смены можно будет вызвать следующий раз L502_OutCycleLoadStart() для загрузки следующего сигнала.

Останов циклического вывода можно осуществить одним из следующий способов:

- L502_OutCycleStop() останавливает циклический вывод после вывода последней точки на границе циклического буфера. То есть эта функция используется чтобы циклический вывод был завершен точно в известной точке и на выходах остались значения соответствующие последним отсчетам в циклическом сигнале.

- L502_StreamsDisable() с указанием всех используемых каналов вывода приводит к немедленному завершению вывода и освобождением всех буферов. На выходах останутся значения которые были в момент вызова. После этого можно будет заново разрешить и проинициализировать вывод как в циклическом так и в потоковом режиме с подкачкой.

- L502_StreamsStop() приводит к немедленному останову всех потоков и останову генерации опорной частоты синхронизации. На выходе остаются те значения которые были в момент вызова функции.

3.6.4 Размер буфера и шаг прерываний для синхронного режима

В данном разделе приводится дополнительная информация о том, как можно настроить дополнительные параметры, управляющие передачей данных по DMA между платой и драйвером. Эти параметры по-умолчанию настраиваются библиотекой автоматически. Предполагается, что большинству пользователей должны подойти автоматически настраиваемые параметры и данный раздел не является обязательным. Однако для случаев, когда автоматически определенные параметры не подходят, пользователь может задать их самостоятельно. Для этого в этом разделе приводится описание как выбирается размер буфера и шаг прерывания библиотекой, что означают эти параметры и как их можно настроить вручную.

Как уже было сказано в предыдущем разделе прием и передача синхронных данных осуществляется через буфера в драйвере - один буфер на прием, один на передачу.

Выделение буфера на ввод осуществляется по L502_StreamsStart(), если был разрешен хотя бы один источник для синхронного ввода. Выделение буфера на вывод осуществляется по L502_PreloadStart().

При этом размер буфера определяется автоматически библиотекой в зависимости от установленной частоты передачи данных. Размер буфера рассчитывается так, чтобы его хватило на 4 секунды при синхронном вводе и на 3 секунды при синхронном выводе.

Передача данных между буфером драйвера и модулем осуществляется непосредственно самим модулем по DMA. При этом, чтобы драйвер мог узнать о том, что данные были записаны в буфер или прочитаны из него, при передаче определенного количества отсчетов модуль генерирует прерывание. То есть реально драйвер "узнает" о том,

что были переданы данные только после того как будет передано заданное количество отсчетов, называемое в данной главе шагом прерываний (Точнее сказать, не позже, чем будет передано заданное количество отсчетов, так как драйвер может прочитать значение счетчика переданных данных из модуля и по другим условиям).

Таким образом, малый шаг прерываний позволяет драйверу раньше узнает о принятых или переданных данных, но приводит к большей загрузке системы. Библиотека рассчитывает шаг прерываний так, чтобы прерывания происходили с частотой 64 раза в секунду.

Если пользователя по каким-либо причинам не устраивают эти значения он может настроить их вручную с помощью функций L502_SetDmaBufSize() и L502_DmaSetIrqStep(). Эти функции должны быть вызваны до инициализации DMA-канала (до L502_StreamsStart() или L502_PreloadStart()).

В частности, случаями когда значения библиотеки могут не устроить, могут быть следующие:

- Пользователь использует свою прошивку BlackFin и использует каналы синхронных данных для передачи пользовательских данных, которые сильно изменяют скорость передачи данных. В этом случае библиотека не может правильно определить частоту передачи, так как не знает скорости передачи пользовательских данных.

- Пользователь изменяет каналы, которые используются в синхронном режиме, на лету (после L502_StreamsStart()) и при этом скорости передачи по этим каналам существенно отличаются. Так как расчет размера буфера выполняется при инициализации канала, то он осуществляется только по тем каналам, которые были разрешены на тот момент. Если, например, был разрешен только синхронный сбор с АЦП на относительно небольшой частоте, то буфер будет выделен также небольшой. При этом, если после запуска сбора данных будет разрешен синхронный ввод с цифровых линий на частоте 2МГц, то вероятнее всего этот буфер окажется недостаточного размера, и с большой вероятностью произойдет его переполнение. Если же оба этих потока были разрешены изначально, а потом синхронный ввод цифровых линий будет запрещен, то рассчитанный изначально шаг прерывания будет слишком большим и данные от медленного потока АЦП будут обновляться с большими задержками. Если же частоты каналов соизмеримы, то включение/отключение одного из них не приведет к существенному изменению параметров. Изменение шага прерывания и размера буфера при запущенном сборе данных на текущий момент невозможно.

Глава 4

Константы, типы данных и функции библиотеки

4.1 Константы и перечисления.

4.1.1 Константы и макроопределения.

Константа	Значение	Описание
L502_LTABLE_MAX_CH_CNT	256	Максимальное количество логических каналов в таблице
L502_ADC_RANGE_CNT	6	Количество диапазонов для измерения напряжений
L502_LCH_AVG_SIZE_MAX	128	Максимальное значение для аппаратного усреднения по логическому каналу
L502_ADC_FREQ_DIV_MAX	(1024*1024)	Максимальное значения делителя частоты АЦП
L502_DIN_FREQ_DIV_MAX	(1024*1024)	Максимальное значение делителя частоты синхронного цифрового ввода
L502_ADC_INTERFRAME_DELAY_MAX	(0x1FFFFF)	Максимальное значение межкадровой задержки для АЦП
L502_BF_CMD_DEFAULT_TOUT	500	Таймаут по умолчанию для выполнения команды к BlackFin
L502_ADC_SCALE_CODE_MAX	6000000	Код АЦП, соответствующий максимальному значению шкалы
L502_DAC_SCALE_CODE_MAX	30000	Код ЦАП, соответствующий максимальному значению шкалы
L502_DEVNAME_SIZE	32	Максимальное количество символов в строке с названием устройства

L502_SERIAL_SIZE	32	Максимальное количество символов в строке с серийным номером
L502_EXT_REF_FREQ_MAX	2000000	Максимально возможное значение внешней опорной частоты
L502_FLASH_USER_SIZE	0x100000	Размер пользовательской области Flash-памяти
L502_BF_REQ_TOUT	500	Стандартный таймаут на выполнение запроса к BlackFin в мс
L502_DAC_RANGE	5.	Диапазон ЦАП в вольтах
L502_DAC_CH_CNT	2	Количество каналов ЦАП
L502_STREAM_IN_MSG_OVERFLOW	0x01010000	слово в потоке, означающее, что произошло переполнение

4.1.2 Коды ошибок библиотеки

Тип: t_lpcie_errs		
Описание: Коды ошибок библиотеки		
Константа	**Значение**	**Описание**
L502_ERR_OK	= 0	Функция выполнена без ошибок
L502_ERR_INVALID_HANDLE	= -1	В функцию передан недействительный описатель модуля
L502_ERR_MEMORY_ALLOC	= -2	Ошибка выделения памяти
L502_ERR_ALREADY_OPENED	= -3	Попытка открыть уже открытое устройство
L502_ERR_DEVICE_NOT_FOUND	= -4	Устройство с заданными параметрами не найдено в системе
L502_ERR_DEVICE_ACCESS_DENIED	= -5	Доступ к устройству запрещен (Как правило из-за того, что устройство уже открыто в другой программе)
L502_ERR_DEVICE_OPEN	= -6	Ошибка открытия устройства
L502_ERR_INVALID_POINTER	= -7	В функцию передан недействительный указатель

L502_ERR_STREAM_IS_RUNNING	= -8	Функция не может быть выполнена при запущенном потоке сбора данных
L502_ERR_RECV	= -9	Ошибка чтения данных синхронного ввода
L502_ERR_SEND	= -10	Ошибка записи данных для синхронного вывода
L502_ERR_STREAM_OVERFLOW	= -11	Произошло переполнение внутреннего буфера для потока синхронного ввода
L502_ERR_UNSUP_STREAM_MSG	= -12	Неизвестное сообщение в потоке синхронного ввода
L502_ERR_MUTEX_CREATE	= -13	Ошибка создания системного мьютекса
L502_ERR_MUTEX_INVALID_HANDLE	= -14	Неверный описатель мьютекса
L502_ERR_MUTEX_LOCK_TOUT	= -15	Истекло время ожидания освобождения мьютекса
L502_ERR_MUTEX_RELEASE	= -16	Ошибка освобождения мьютекса
L502_ERR_INSUFFICIENT_SYSTEM_RESOURCES	= -17	Недостаточно системных ресурсов
L502_ERR_NOT_IMPLEMENTED	= -18	Данная возможность еще не реализована
L502_ERR_INSUFFICIENT_ARRAY_SIZE	= -19	Недостаточный размер массива
L502_ERR_FPGA_REG_READ	= -20	Ошибка чтения регистра FPGA
L502_ERR_FPGA_REG_WRITE	= -21	Ошибка записи регистра FPGA
L502_ERR_STREAM_IS_NOT_RUNNING	= -22	Сбор данных уже остановлен
L502_ERR_INVALID_LTABLE_SIZE	= -102	Задан неверный размер логической таблицы
L502_ERR_INVALID_LCH_NUMBER	= -103	Задан неверный номер логического канала
L502_ERR_INVALID_LCH_RANGE	= -104	Неверно задано значение диапазона АЦП
L502_ERR_INVALID_LCH_MODE	= -105	Неверно задан режим измерения для логического канала
L502_ERR_INVALID_LCH_PHY_NUMBER	= -106	Неверно задан номер физического канала при настройке логического

L502_ERR_INVALID_LCH_AVG_SIZE	= -107	Неверно задан размер усреднения для логического канала
L502_ERR_INVALID_ADC_FREQ_DIV	= -108	Неверно задан делитель частоты сбора данных АЦП
L502_ERR_INVALID_DIN_FREQ_DIV	= -108	Неверно задан делитель частоты синхронного ввода цифровых линий
L502_ERR_INVALID_MODE	= -109	Неверно задан режим работы модуля L502
L502_ERR_INVALID_DAC_CHANNEL	= -110	Неверный номер канала ЦАП
L502_ERR_INVALID_REF_FREQ	= -111	Неверный код выбора опорной частоты синхронизации
L502_ERR_INVALID_INTERFRAME_DELAY	= -112	Неверно задано значение межкадровой задержки
L502_ERR_INVALID_SYNC_MODE	= -113	Неверно задан режим синхронизации
L502_ERR_INVALID_DMA_CH	= -114	Неверно задан номер канала DMA
L502_ERR_REF_FREQ_NOT_LOCKED	= -131	Ошибка захвата опорной частоты синхронизации
L502_ERR_IOCTL_FAILD	= -132	Управляющий запрос к драйверу завершен с ошибкой
L502_ERR_IOCTL_TIMEOUT	= -133	Истек таймаут ожидания завершения выполнения управляющего запроса к драйверу
L502_ERR_GET_INFO	= -134	Ошибка получения информации о устройстве от драйвера
L502_ERR_DIG_IN_NOT_RDY	= -135	За время ожидания не было считано новое слово с цифровых линий
L502_ERR_RECV_INSUFFICIENT_WORDS	= -136	Принято недостаточно слов от модуля
L502_ERR_DAC_NOT_PRESENT	= -137	Попытка выполнить операцию, требующую наличие ЦАП, при его отсутствии
L502_ERR_PROC_INVALID_CH_NUM	= -140	Неверный номер канала в обрабатываемом потоке синхронного ввода

L502_ERR_PROC_INVALID_CH_RANGE	= -141	Неверный код диапазона в обрабатываемом потоке синхронного ввода
L502_ERR_FLASH_INVALID_ADDR	= -142	Задан неверный адрес во Flash-памяти
L502_ERR_FLASH_INVALID_SIZE	= -143	Задан неверный размер блока данных при работе с Flash-памятью
L502_ERR_FLASH_WRITE_TOUT	= -144	Истек таймаут ожидания завершения записи во Flash-память
L502_ERR_FLASH_ERASE_TOUT	= -145	Истек таймаут ожидания завершения стирания блока Flash-памяти
L502_ERR_FLASH_SECTOR_BOUNDARY	= -146	Заданная область для стирания Flash-памяти нарушает границу блока в 4 Кбайт
L502_ERR_LDR_FILE_OPEN	= -180	Не удалось открыть файл прошивки BlackFin
L502_ERR_LDR_FILE_READ	= -181	Ошибка чтения из фала прошивки BlackFin
L502_ERR_LDR_FILE_FORMAT	= -182	Неверный формат файла прошивки BlackFin
L502_ERR_LDR_FILE_UNSUP_FEATURE	= -183	Используются возможность LDR-файла, недоступные при записи прошивки BlackFin по HDMA
L502_ERR_LDR_FILE_UNSUP_STARTUP_ADDR	= -184	Неверный стартовый адрес программы в прошивке BlackFin
L502_ERR_BF_REQ_TIMEOUT	= -185	Истек таймаут выполнения запроса на чтения/запись памяти BlackFin
L502_ERR_BF_CMD_IN_PROGRESS	= -186	Команда для BlackFin все еще находится в процессе обработки
L502_ERR_BF_CMD_TIMEOUT	= -187	Истекло время выполнения управляющей команды процессором BlackFin
L502_ERR_BF_CMD_RETURN_INSUF_DATA	= -188	Возвращено недостаточно данных в ответ на команду к BlackFin

L502_ERR_BF_LOAD_RDY_TOUT	= -189	Истек таймаут ожидания готовности процессора BlackFin к записи прошивки
L502_ERR_BF_NOT_PRESENT	= -190	Попытка выполнить операцию для которой нужен сигнальный процессор при отсутствии сигнального процессора в модуле
L502_ERR_BF_INVALID_ADDR	= -191	Неверный адрес памяти BlackFin при записи или чтении по HDMA
L502_ERR_BF_INVALID_CMD_DATA_SIZE	= -192	Неверный размер данных, передаваемых с управляющей командой в BlackFin

4.1.3 Флаги, управляющие поиском присутствующих модулей

Тип: t_l502_getdevs_flags		
Описание: Флаги, управляющие поиском присутствующих модулей		
Константа	**Значение**	**Описание**
L502_GETDEVS_FLAGS_ONLY_NOT_OPENED	= 0x0001	Признак, что нужно вернуть серийные номера только тех устройств, которые еще не открыты

4.1.4 Флаги для управления цифровыми выходами.

Тип: t_l502_digout_word_flags		
Описание: Флаги управления цифровыми выходами. Могут быть объединены через логическое “ИЛИ” со значениями цифровых выходов при асинхронном выводе с помощью L502_AsyncOutDig() или переданы в L502_PrepareData() при синхронном выводе.		
Константа	**Значение**	**Описание**
L502_DIGOUT_WORD_DIS_H	= 0x00020000	Запрещение (перевод в третье состояние) старшей половины цифровых выходов
L502_DIGOUT_WORD_DIS_L	= 0x00010000	Запрещение младшей половины цифровых выходов

4.1.5 Константы для выбора опорной частоты

Тип: t_l502_ref_freq		
Описание: Константы для выбора опорной частоты		
Константа	**Значение**	**Описание**
L502_REF_FREQ_2000KHZ	= 2000000	Частота 2МГц
L502_REF_FREQ_1500KHZ	= 1500000	Частота 1.5МГц

4.1.6 Диапазоны измерения для канала АЦП

Тип: t_l502_adc_range		
Описание: Диапазоны измерения для канала АЦП		
Константа	**Значение**	**Описание**
L502_ADC_RANGE_10	= 0	Диапазон +/-10V
L502_ADC_RANGE_5	= 1	Диапазон +/-5V
L502_ADC_RANGE_2	= 2	Диапазон +/-2V
L502_ADC_RANGE_1	= 3	Диапазон +/-1V
L502_ADC_RANGE_05	= 4	Диапазон +/-0.5V
L502_ADC_RANGE_02	= 5	Диапазон +/-0.2V

4.1.7 Режим измерения для логического канала

Тип: t_l502_lch_mode		
Описание: Режим измерения для логического канала		
Константа	**Значение**	**Описание**
L502_LCH_MODE_COMM	= 0	Измерение напряжения относительно общей земли
L502_LCH_MODE_DIFF	= 1	Дифференциальное измерение напряжения
L502_LCH_MODE_ZERO	= 2	Измерение собственного нуля

4.1.8 Режимы синхронизации.

Тип: t_l502_sync_mode		
Описание: Режимы задания источника частоты синхронизации и признака начала синхронного ввода-вывода		
Константа	**Значение**	**Описание**
L502_SYNC_INTERNAL	= 0	Внутренний сигнал
L502_SYNC_EXTERNAL_MASTER	= 1	От внешнего мастера по разъему синхронизации
L502_SYNC_DI_SYN1_RISE	= 2	По фронту сигнала DI_SYN1
L502_SYNC_DI_SYN2_RISE	= 3	По фронту сигнала DI_SYN2
L502_SYNC_DI_SYN1_FALL	= 6	По спаду сигнала DI_SYN1
L502_SYNC_DI_SYN2_FALL	= 7	По спаду сигнала DI_SYN2

4.1.9 Флаги, управляющие обработкой принятых данных

Тип: t_l502_proc_flags		
Описание: Флаги, управляющие обработкой принятых данных		
Константа	**Значение**	**Описание**
L502_PROC_FLAGS_VOLT	= 1	Признак, что нужно преобразовать значения АЦП в вольты

4.1.10 Флаги для обозначения синхронных потоков данных

Тип: t_l502_streams		
Описание: Флаги для обозначения синхронных потоков данных		
Константа	**Значение**	**Описание**
L502_STREAM_ADC	= 0x01	Поток данных от АЦП
L502_STREAM_DIN	= 0x02	Поток данных с цифровых входов
L502_STREAM_DAC1	= 0x10	Поток данных первого канала ЦАП
L502_STREAM_DAC2	= 0x20	Поток данных второго канала ЦАП
L502_STREAM_DOUT	= 0x40	Поток данных на цифровые выводы
L502_STREAM_ALL_IN	= L502_STREAM_ADC | L502_STREAM_DIN	Объединение всех флагов, обозначающих потоки данных на ввод
L502_STREAM_ALL_OUT	= L502_STREAM_DAC1 | L502_STREAM_DAC2 | L502_STREAM_DOUT	Объединение всех флагов, обозначающих потоки данных на вывод

4.1.11 Константы, определяющие тип передаваемого отсчета из ПК в модуль

Тип: t_stream_out_wrd_type		
Описание: Константы, определяющие тип передаваемого отсчета из ПК в модуль		
Константа	**Значение**	**Описание**
L502_STREAM_OUT_WORD_TYPE_DOUT	= 0x0	Цифровой вывод
L502_STREAM_OUT_WORD_TYPE_DAC1	= 0x40000000	Код для 1-го канала ЦАП
L502_STREAM_OUT_WORD_TYPE_DAC2	= 0x80000000	Код для 2-го канала ЦАП

4.1.12 Режим работы модуля L502

Тип: t_l502_mode		
Описание: Режим работы модуля L502		
Константа	**Значение**	**Описание**
L502_MODE_FPGA	= 0	Все потоки данных передаются через ПЛИС минуя сигнальный процессор BlackFin
L502_MODE_DSP	= 1	Все потоки данных передаются через сигнальный процессор, который должен быть загружен прошивкой для обработки этих потоков
L502_MODE_DEBUG	= 2	Отладочный режим

4.1.13 Номера каналов ЦАП.

Тип: t_l502_dac_ch		
Описание: Номер каналов ЦАП для указания в L502_AsyncOutDac()		
Константа	**Значение**	**Описание**
L502_DAC_CH1	= 0	Первый канал ЦАП
L502_DAC_CH2	= 1	Второй канал ЦАП

4.1.14 Флаги, используемые при выводе данных на ЦАП.

Тип: t_l502_dacout_flags		
Описание: Флаги, комбинацию которых можно передать в L502_AsyncOutDac() или L502_PrepareData(), чтобы определить действия, которые должны выполнить эти функции с переданным значением перед выводом их на ЦАП		
Константа	**Значение**	**Описание**
L502_DAC_FLAGS_VOLT	= 0x0001	Указывает, что значение задано в Вольтах и при выводе его нужно перевести его в коды ЦАП. Если флаг не указан, то считается, что значение изначально в кодах
L502_DAC_FLAGS_CALIBR	= 0x0002	Указывает, что нужно применить калибровочные коэффициенты перед выводом значения на ЦАП.

4.1.15 Номера каналов DMA

Тип: t_l502_dma_ch		
Описание: Номера каналов DMA		
Константа	**Значение**	**Описание**
L502_DMA_CH_IN	= 0	Общий канал DMA на ввод
L502_DMA_CH_OUT	= 1	Общий канал DMA на вывод

4.1.16 Цифровые линии, на которых можно включить подтягивающие резисторы

Тип: t_l502_pullups		
Описание: Флаги, определяющие на каких цифровых входах должны быть включены подтягивающие резисторы		
Константа	**Значение**	**Описание**
L502_PULLUPS_DI_H	= 0x01	Старшая половина цифровых входов
L502_PULLUPS_DI_L	= 0x02	Младшая половина цифровых входов
L502_PULLUPS_DI_SYN1	= 0x04	Линия SYN1
L502_PULLUPS_DI_SYN2	= 0x08	Линия SYN2

4.1.17 Флаги, определяющие наличие опций в модуле

Тип: t_l502_dev_flags		
Описание: Флаги, определяющие наличие опций в модуле		
Константа	**Значение**	**Описание**
L502_DEVFLAGS_DAC_PRESENT	= 0x00000002	Признак наличия двухканального канального ЦАП
L502_DEVFLAGS_GAL_PRESENT	= 0x00000004	Признак наличия гальваноразвязки
L502_DEVFLAGS_BF_PRESENT	= 0x00000001	Признак наличия сигнального процессора BlackFin
L502_DEVFLAGS_FLASH_DATA_VALID	= 0x00010000	Признак, что во Flash-памяти присутствует информация о модуле
L502_DEVFLAGS_FLASH_ADC_CALIBR_VALID	= 0x00020000	Признак, что во Flash-памяти присутствуют действительные калибровочные коэффициенты АЦП
L502_DEVFLAGS_FLASH_DAC_CALIBR_VALID	= 0x00040000	Признак, что во Flash-памяти присутствуют действительные калибровочные коэффициенты ЦАП

4.1.18 Флаги для режима циклического вывода

Тип: t_l502_out_cycle_flags		
Описание: Данные флаги могут быть переданы в L502_OutCycleSetup() и L502_OutCycleStop()		
Константа	**Значение**	**Описание**
L502_OUT_CYCLE_FLAGS_FORCE	= 0x01	Флаг указывает, что останов или смена сигнала могут произойти без ожидания конца цикла предыдущего сигнала. Это позволяет выполнить переключение быстрее (однако все равно может быть поставлено на передачу до 256 КСемплов, которые должны будут быть переданы), но точка смены или останова может быть в любом месте периода

4.2 Типы данных.

4.2.1 Калибровочные коэффициенты диапазона.

Тип: t_l502_cbr_coef		
Описание: Структура содержит калибровочные значения смещения нуля и коэффициента шкалы для одного диапазона АЦП или ЦАП.Результирующее значение АЦП вычисляется как (val-offs)*k, где val - некалиброванное значение		
Поле	**Тип**	**Описание поля**
offs	double	смещение нуля
k	double	коэффициент шкалы

4.2.2 Калибровочные коэффициенты модуля.

Тип: t_l502_cbr		
Описание: Структура, содержащая все калибровочные коэффициенты, которые используются модулем L502		
Поле	**Тип**	**Описание поля**
adc	t_l502_cbr_coef	Калибровочные коэффициенты АЦП
res1	uint32_t	Резерв
dac	t_l502_cbr_coef	Калибровочные коэффициенты ЦАП
res2	uint32_t	Резерв

4.2.3 Информация о модуле L502.

Тип: t_l502_info		
Описание: Структура, содержащая постоянную информация о модуле L502, которая как правило не изменяется после открытия		
Поле	**Тип**	**Описание поля**
name	char	Название устройства ("L502")
serial	char	Серийный номер
devflags	uint32_t	в модуле определенных опций
fpga_ver	uint16_t	Версия ПЛИС (старший байт - мажорная, младший - минорная)
plda_ver	uint8_t	Версия ПЛИС, управляющего аналоговой частью
board_rev	uint8_t	Ревизия платы
res	uint8_t	Резерв
cbr	t_l502_cbr	Заводские калибровочные коэффициенты (из Flash-памяти)

4.2.4 Описатель модуля.

Тип: `t_l502_hnd`
Описание: Непрозрачный указатель на структуру, содержащую информацию о настройках модуля и текущем соединении с ним. Пользовательской программе не доступны поля структуры напрямую, а только через функции библиотеки. Функции управления модулем принимают описатель модуля своим первым параметром. Описатель модуля создается с помощью `L502_Create()` и в конце работы освобождается с помощью `L502_Free()`.

4.2.5 Списко серийный номеров

Тип: `t_l502_serial_list`
Описание: Тип определяет массив серийных номеров для количестава модулей, определяемого на этапе работы программы.

4.3 Функции

4.3.1 Функции для создания и освобождения описателя модуля.

4.3.1.1 Создание описателя модуля.

Формат: `t_l502_hnd L502_Create(void)`
Описание: Создание описателя модуля, для последующей работы с модулем L502. В случае успешного выделения памяти инициализирует поля описателя значениями по-умолчанию.
Возвращаемое значение: NULL в случае ошибки, иначе - описатель модуля

4.3.1.2 Освобождение описателя модуля.

Формат: `int32_t L502_Free(t_l502_hnd hnd)`
Описание: Освобождение памяти, выделенной под описатель модуля с помощью `L502_Create()`. После этого описатель уже использовать нельзя, независимо от возвращенного значения!
Параметры: **hnd** — Описатель устройства
Возвращаемое значение: Код ошибки

4.3.2 Функции для открытия и получения информации о модуле.

4.3.2.1 Получение списка серийных номеров модулей L502.

Формат: `int32_t L502_GetSerialList(char serials[][L502_SERIAL_SIZE], uint32_t size, uint32_t flags, uint32_t *devcnt)`

Описание: Функция возвращает список номеров всех найденных модулей L502, независимо от того, открыты они сейчас или нет.
Если нужен список только тех модулей, которые не открыты (то есть только тех, с которыми можно установить соединение), то для этого можно передать в функцию флаг `L502_GETDEVS_FLAGS_ONLY_NOT_OPENED`.

Параметры:

serials — Массив размером size*`L502_SERIAL_SIZE` байт, в который будут сохранены серийные номера найденных модулей. Может быть NULL, если size=0, а devcnt!=NULL, в случае, если нужно только получить количество модулей в системе.

size — Определяет, сколько максимально серийных номеров может быть сохранено в массив serial. Будут сохранены только первые size серийных номеров. Может быть 0, если serials=NULL

flags — Флаги из `t_l502_getdevs_flags`, определяющие поведение функции.

devcnt — Если devcnt!=NULL, то в данную переменную сохраняется общее число найденных модулей L502 (может быть больше size).

Возвращаемое значение:
Если <0 - код ошибки, иначе количество сохраненных серийных номеров в массиве serials (всегда <= size)

4.3.2.2 Открытие модуля по его серийному номеру.

Формат: `int32_t L502_Open(t_l502_hnd hnd, const char *serial)`

Описание: Функция устанавливает связь с модулем L502 по его серийному номеру. После успешного выполнения этой функции, пользователь получает эксклюзивный доступ к модулю через описатель модуля. До закрытия связи с помощью `L502_Close()` никто другой установить связь с модулем не сможет (будет возвращена ошибка `L502_ERR_DEVICE_ACCESS_DENIED`).
Если в качестве серийного номера передан NULL или пустая строка, то будет установлена связь с первым найденным модулем, с которым получится успешно ее установить. Если в системе нет ни одного модуля, то будет возвращена ошибка `L502_ERR_DEVICE_NOT_FOUND`. Если в системе присутствуют модули L502, но соединение ни с одним из них установить не удалось, то будет возвращена ошибка, полученная при попытке установить соединение с последним найденным модулем.
После завершения работы с устройством соединение должно быть закрыто с помощью `L502_Close()`.

Параметры:

hnd — Описатель устройства.

serial — Указатель на строку с серийным номером открываемого модуля или NULL.

Возвращаемое значение:
Код ошибки.

4.3.2.3 Закрытие соединения с модулем.

Формат: `int32_t L502_Close(t_l502_hnd hnd)`

Описание: Функция разрывает соединение с модулем L502, если оно было ранее установлено (в противном случае ничего не делает). Описатель модуля не освобождается. Память под описатель модуля должна быть освобождена вызовом `L502_Free()`.

Параметры:

hnd — Описатель модуля.

Возвращаемое значение:
Код ошибки.

4.3.2.4 Получение информации о модуле.

Формат: `int32_t L502_GetDevInfo(t_l502_hnd hnd, t_l502_info *info)`
Описание: Получение информации о модуле L502, с которым установлена связь.
Параметры: **hnd** — Описатель модуля. **info** — Информация о модуле (смотри описание типа t_l502_info).
Возвращаемое значение: Код ошибки.

4.3.3 Функции для изменения настроек модуля

4.3.3.1 Передача установленных настроек в модуль.

Формат: `int32_t L502_Configure(t_l502_hnd hnd, uint32_t flags)`
Описание: Функция выполняет запись текущих настроек (которые были установлены с помощью функций L502_SetXXX) в модуль. Должна вызываться перед запуском потока данных.
Параметры: **hnd** — Описатель модуля. **flags** — Флаги (резерв - должно быть равно 0).
Возвращаемое значение: Код ошибки.

4.3.3.2 Установка параметров логического канала.

Формат: `int32_t L502_SetLChannel(t_l502_hnd hnd, uint32_t lch, uint32_t phy_ch, uint32_t mode, uint32_t range, uint32_t avg)`

Описание: Функция устанавливает параметры заданного логического канала в логической таблице АЦП.

Параметры:

hnd — Описатель модуля.

lch — Номер логического канала. (от 0 до L502_LTABLE_MAX_CH_CNT-1)

phy_ch — Номер физического канала АЦП, начиная с 0 (0-15 для дифференциального режима, 0-31 для режима с общей землей)

mode — Режим измерения канал АЦП (значение типа t_l502_lch_mode)

range — Диапазон измерения канала (значение типа t_l502_adc_range)

avg — Коэффициент усреднения по каналу. Нулевое значение соответствует значению коэффициента, определенного библиотекой. Для явного задания коэффициента усреднения нужно перед значение от 1 (отсутствие усреднения) до L502_LCH_AVG_SIZE_MAX. В случае если значение усреднения превышает делитель частоты, то это значение будет скорректировано

Возвращаемое значение:
Код ошибки.

4.3.3.3 Установка количества логических каналов.

Формат: `int32_t L502_SetLChannelCount(t_l502_hnd hnd, uint32_t lch_cnt)`

Описание: Функция устанавливает количество логических каналов в логической таблице АЦП.

Параметры:

hnd — Описатель модуля

lch_cnt — Количество логических каналов (от 1 до L502_LTABLE_MAX_CH_CNT)

Возвращаемое значение:
Код ошибки

4.3.3.4 Получение количества логических каналов.

Формат: `int32_t L502_GetLChannelCount(t_l502_hnd hnd, uint32_t *lch_cnt)`
Описание: Функция возвращает установленное ранее с помощью `L502_SetLChannelCount()` количество логических каналов в управляющей таблице АЦП.
Параметры: **hnd** — Описатель модуля **lch_cnt** — Количество логических каналов
Возвращаемое значение: Код ошибки

4.3.3.5 Установка делителя частоты сбора для АЦП.

Формат: `int32_t L502_SetAdcFreqDivider(t_l502_hnd hnd, uint32_t adc_freq_div)`
Описание: Частота сбора АЦП получается как результат деления опорной частоты синхронизации (как в случае внешней, так и внутренней) на делитель, устанавливаемый этой функцией. Альтернативой этой функции служит `L502_SetAdcFreq()`, которая рассчитывает этот делитель на основании переданной требуемой частоты сбора АЦП.
Параметры: **hnd** — Описатель модуля. **adc_freq_div** — Делитель частоты АЦП (от 1 до `L502_ADC_FREQ_DIV_MAX`).
Возвращаемое значение: Код ошибки.

4.3.3.6 Установка значения межкадровой задержки для АЦП.

Формат: `int32_t L502_SetAdcInterframeDelay(t_l502_hnd hnd, uint32_t delay)`
Описание: Функция устанавливает межкадровую задержку для АЦП, то есть количество периодов опорной частоты синхронизации, которое будет пропущено после проведения измерения последнего канала логической таблицы до проведения измерения, соответствующего первому логическому каналу следующего кадра. Альтернативой может являться функция `L502_SetAdcFreq()`, которая рассчитывает значение межкадровой задержки по заданным параметрам частоты сбора и частоты следования кадров (частоты сбора на логический канал).
Параметры: **hnd** — Описатель модуля. **delay** — Значение межкадровой задержки (от 0 до `L502_ADC_INTERFRAME_DELAY_MAX`)
Возвращаемое значение: Код ошибки.

4.3.3.7 Установка делителя частоты синхронного ввода с цифровых линий.

Формат: `int32_t L502_SetDinFreqDivider(t_l502_hnd hnd, uint32_t din_freq_div)`
Описание: Частота синхронного ввода данных с цифровых входов получается как результат деления опорной частоты синхронизации на делитель, устанавливаемый этой функцией. Альтернативой этой функции служит `L502_SetDinFreq()`, которая рассчитывает этот делитель на основании переданной требуемой частоты синхронного ввода с цифровых линий.
Параметры: **hnd** — Описатель модуля. **din_freq_div** — Делитель частоты АЦП (от 1 до `L502_DIN_FREQ_DIV_MAX`).
Возвращаемое значение: Код ошибки.

4.3.3.8 Установка частоты сбора АЦП.

Формат: `int32_t L502_SetAdcFreq(t_l502_hnd hnd, double *f_acq, double *f_frame)`

Описание: Функция подбирает делитель частоты АЦП так, чтобы полученная частота сбора была наиболее близка к указанной в параметре f_acq. Функция возвращает в этом же параметре реальную частоту, которая была установлена. Так же функция может подобрать значение межкадровой задержки так, чтобы частота следования кадров (частота сбора на логический канал) была наиболее близка к указанному значению. Для этого следует передать требуемое значение в переменной f_frame (в ней также по завершению будет возвращено значение установленной частоты). Если в качестве f_frame передан нулевой указатель, то будет установлена нулевая межкадровая задержка.
Если необходимо изменить значение опорной частоты, то данная функция должна быть вызвана после `L502_SetRefFreq()`, в противном случае полученные делители будут давать неверное значение частоты.
Если устанавливается частота кадров, то функция должна вызываться после того, как было заданно нужное количество логических каналов в управляющей таблице с помощью `L502_SetLChannelCount()`.
При использовании внешней опорной частоты синхронизации эта функция будет давать верный результат, только если эта внешняя частота соответствует значению, установленному с помощью `L502_SetRefFreq()`.

Параметры:

hnd — Описатель модуля.

f_acq — На входе принимает требуемое значения частоты сбора АЦП в Герцах. На выходе возвращает реально установленное значение частоты.

f_frame — На входе принимает требуемое значение частоты сбора кадров (частоты сбора на логический канал) АЦП в Герцах. На выходе возвращает реально установленное значение. Если передан нулевой указатель, то устанавливает максимальную частоту сбора кадров (нулевую межкадровую задержку).

Возвращаемое значение:
Код ошибки.

4.3.3.9 Установка частоты синхронного ввода с цифровых входов.

Формат: `int32_t L502_SetDinFreq(t_l502_hnd hnd, double *f_din)`

Описание: Функция подбирает делитель частоты ввода значений с цифровых входов так, чтобы полученная частота ввода была наиболее близка к указанной. Функция возвращает в этом же параметре реальную частоту, которая была установлена.
Если необходимо изменить значение опорной частоты синхронизации, то данная функция должна быть вызвана после `L502_SetRefFreq()`, в противном случае полученный делитель будет давать неверное значение частоты.
При использовании внешней опорной частоты синхронизации эта функция будет давать верный результат, только если эта внешняя частота соответствует значению, установленному с помощью `L502_SetRefFreq()`.

Параметры:

hnd — Описатель модуля.

f_din — На входе принимает требуемое значения частоты ввода с цифровых входов в Герцах. На выходе возвращает реально установленное значение частоты.

Возвращаемое значение:
Код ошибки.

4.3.3.10 Получить текущие значения частот сбора АЦП

Формат: `int32_t L502_GetAdcFreq(t_l502_hnd hnd, double *f_acq, double *f_frame)`

Описание: Функция возвращает ткущие установленные для модуля значения частоты сбора и частоты кадров АЦП (частоты на логический канал) в Герцах, которые были установлены до этого с помощью `L502_SetAdcFreq()` или с помощью функций `L502_SetAdcFreqDivider()`/L502_SetAdcInterframeDelay().

Параметры:

hnd — Описатель модуля.

f_acq — Если не NULL, то на выходе возвращается текущее значение частоты сбора АЦП.

f_frame — Если не NULL, то на выходе возвращается текущее значение частоты кадров АЦП.

4.3.3.11 Установка значения опорной частоты синхронизации.

Формат: `int32_t L502_SetRefFreq(t_l502_hnd hnd, uint32_t freq)`

Описание: Функция задает значение опорной частоты синхронизации, от которой получаются все частоты синхронного ввода/вывода посредством деления на определенный делитель.
При использовании внутренней опорной частоты доступны два значения: 2МГц и 1.5Мгц (2МГц является значением по-умолчанию), для задания которых можно использовать константы из `t_l502_ref_freq`.
При использовании внешней опорной частоты для того, чтобы можно было правильно установить нужную частоту сбора функциями `L502_SetAdcFreq()`/L502_SetDinFreq() и правильно были установлены значения по-умолчанию для размера буфера DMA и шага прерываний, нужно с помощью этой функции задать реально подаваемое значение внешней опорной частоты в Герцах.

Параметры:

hnd — Описатель модуля.

freq — Значение из `t_l502_ref_freq`, которое задает выбранную опорную частоту.

Возвращаемое значение:
Код ошибки.

4.3.3.12 Установка режима генерации частоты синхронизации.

Формат: `int32_t L502_SetSyncMode(t_l502_hnd hnd, uint32_t sync_mode)`

Описание: Функция устанавливает кто будет генератором опорной частоты синхронизации - сам модуль или будет использоваться внешний сигнал.
В режиме `L502_SYNC_INTERNAL` модуль сам будет генерировать для себя частоту синхронизации с частотой, заданной `L502_SetRefFreq()`. При этом запуск генерации будет осуществлен по вызову `L502_StreamsStart()` или по условию, заданому в `L502_SetSyncStartMode()`, а останов по `L502_StreamsStop()`.
В остальных режимах сбор будет осуществляться по внешнему сигналу синхронизации.

Параметры:

hnd — Описатель модуля.

sync_mode — Значение из `t_l502_sync_mode`, определяющее кто будет источником частоты синхронизации.

Возвращаемое значение:
Код ошибки.

4.3.3.13 Установка режима запуска частоты синхронизации.

Формат: `int32_t L502_SetSyncStartMode(t_l502_hnd hnd, uint32_t sync_start_mode)`

Описание: Функция устанавливает условие запуска синхронного ввода/вывода данных.
Если с помощью `L502_SetSyncMode()` установлен режим синхронизации `L502_SYNC_INTERNAL`, то по заданному данной функцией условию модуль начнет генерировать частоту синхронизации, в противном случае по заданному условию модуль начнет использовать внешне заданную частоту синхронизации (т.е. до выполнения условия сигнал синхронизации на заданном входе будет игнорироваться).
Режимы задания условия запуска синхронизации имеют те же значения, что и режимы задания самой частоты (см. тип `t_l502_sync_mode`). В случае `L502_SYNC_INTERNAL` запуск осуществляется при выполнении функции `L502_StreamsStart()`, в противном случае - после выполнения `L502_StreamsStart()` модуль начинает ожидать заданного данной функцией условия. Т.е. даже при задании внешних источников синхронизации, все равно необходимо вызывать `L502_StreamsStart()`.

Параметры:

hnd — Описатель модуля.

sync_start_mode — Значение из `t_l502_sync_mode`, определяющее условие запуска частоты синхронизации.

Возвращаемое значение:
Код ошибки.

4.3.3.14 Установить режим работы модуля.

Формат: `int32_t L502_SetMode(t_l502_hnd hnd, uint32_t mode)`

Описание: Функция устанавливает режим работы модуля, который определяет будет ли потоки данных обрабатывать ПЛИС или сигнальный процессор BlackFin. При включении питания модулем всегда управляет ПЛИС. После загрузки прошивки с помощью `L502_BfLoadFirmware()` модуль переходит в режим управления сигнальным процессором.
Данная функция может использоваться для ручной установки режима, например, для возврата в режим управления ПЛИС или для переключения в режим управления сигнальным процессором, если прошивка уже была загружена (например, через JTAG интерфейс при отладке).

Параметры:

hnd — Описатель модуля.

mode — Режим работы модуля из `t_l502_mode`.

Возвращаемое значение:
Код ошибки.

4.3.3.15 Получение текущего режима работы модуля.

Формат: `int32_t L502_GetMode(t_l502_hnd hnd, uint32_t *mode)`

Описание: Функция возвращает текущий режим работы модуля.

Параметры:

hnd — Описатель модуля.

mode — В данном параметре возвращается текущий режим работы модуля (из `t_l502_mode`).

Возвращаемое значение:
Код ошибки.

4.3.3.16 Установить коэффициенты для калибровки значений АЦП.

Формат: `int32_t L502_SetAdcCoef(t_l502_hnd hnd, uint32_t range, double k, double offs)`

Описание: Функция записывает в ПЛИС коэффициенты для калибровки значений АЦП. При открытии модуля, библиотека считывает калибровочные коэффициенты из защищенной области Flash-памяти модуля и записывает их в ПЛИС для выполнения калибровки на лету.
Результирующее значение АЦП вычисляется по формуле (val+offs)*k, где val - некалиброванное значение.
Данная функция позволяет изменить используемые коэффициенты в то время, пока не запущен синхронный сбор данных. При этом изменяются только текущие коэффициенты, а заводские калибровочные коэффициенты из Flash-памяти сохраняют свое значение и при следующем открытии будут восстановлены.

Параметры:

hnd — Описатель модуля.

range — Диапазон АЦП (из `t_l502_adc_range`).

k — Устанавливаемое значение коэффициента шкалы.

offs — Устанавливаемое значение смещения нуля.

Возвращаемое значение:
Код ошибки.

4.3.3.17 Получение текущих калибровочных коэффициентов АЦП.

Формат: `int32_t L502_GetAdcCoef(t_l502_hnd hnd, uint32_t range, double *k, double *offs)`

Описание: Функция возвращает текущие калибровочные коэффициенты для заданного диапазона измерения АЦП. Эти коэффициенты могут отличаться от заводских значений, сохраненных во Flash-памяти модуля, например, если пользователь использовал `L502_SetAdcCoef()` для установки своих коэффициентов.

Параметры:

hnd — Описатель модуля.

range — Диапазон АЦП (из `t_l502_adc_range`).

k — В данной переменной возвращается текущий коэффициент шкалы.

offs — В данной переменной возвращается текущее смещение нуля.

Возвращаемое значение:
Код ошибки.

4.3.3.18 Установить коэффициенты для калибровки значений ЦАП.

Формат: `int32_t L502_SetDacCoef(t_l502_hnd hnd, uint32_t ch, double k, double offs)`

Описание: Функция устанавливает калибровочные коэффициенты для заданного канала АЦП, которые будут использоваться функциями l502api для калибровки выводимых значений ЦАП, если указан фалаг `L502_DAC_FLAGS_CALIBR`.
Откалиброванное значение ЦАП в кодах получается как (val+offs)*k, где val - некалиброванное значение (в кодах).
При открытии модуля, библиотека считывает калибровочные коэффициенты из защищенной области Flash-памяти модуля и использует их.
Данная функция нужна только если пользователь хочет использовать свои коэффициенты. При этом она не изменяет значения во Flash-памяти, т.е. при следующем открытии модуля коэффициенты будут снова восстановлены из Flash-памяти.

Параметры:

hnd — Описатель модуля.

ch — Канал ЦАП (из `t_l502_dac_ch`).

k — Устанавливаемое значение коэффициента шкалы.

offs — Устанавливаемое значение смещения нуля.

Возвращаемое значение:
Код ошибки.

4.3.3.19 Получение текущих калибровочных коэффициентов ЦАП.

Формат: `int32_t L502_GetDacCoef(t_l502_hnd hnd, uint32_t ch, double *k, double *offs)`

Описание: Функция возвращает текущие калибровочные коэффициенты для заданного канала ЦАП. Эти коэффициенты могут отличаться от заводских значений, сохраненных во Flash-памяти модуля, например, если пользователь использовал `L502_SetDacCoef()` для установки своих коэффициентов.

Параметры:

hnd — Описатель модуля.

ch — Канал ЦАП (из `t_l502_dac_ch`).

k — В данной переменной возвращается текущий коэффициент шкалы.

offs — В данной переменной возвращается текущее смещение нуля.

Возвращаемое значение:
Код ошибки.

4.3.4 Функции асинхронного ввода-вывода

4.3.4.1 Асинхронный вывод данных на канал ЦАП.

Формат: `int32_t L502_AsyncOutDac(t_l502_hnd hnd, uint32_t ch, double data, uint32_t flags)`

Описание: Функция выводит указанное значение на указанный канал ЦАП. Значение может быть задано как в кодах, так и в Вольтах, и к нему могут быть применены калибровочные коэффициенты (определяется флагами).
Функция может вызываться либо когда синхронный сбор не запущен, либо при запущенном сборе данных, если синхронный сбор по этому каналу ЦАП не разрешен.

Параметры:

hnd — Описатель модуля.

ch — Номер канала ЦАП (из `t_l502_dac_ch`).

data — Выводимое значение на ЦАП (в кодах или вольтах)

flags — Флаги из `t_l502_dacout_flags`.

Возвращаемое значение:
Код ошибки.

4.3.4.2 Асинхронный вывод данных на цифровые выходы.

Формат: `int32_t L502_AsyncOutDig(t_l502_hnd hnd, uint32_t val, uint32_t msk)`

Описание: Функция выводит указанное значение на цифровые выходы модуля. Формат значения аналогичен `L502_PrepareData()` - в младших 16 битах указывается выводимое значение, а в старшие - флаги (с помощью которых можно перевести одну из половин в третье состояние).
Функция может вызываться либо когда синхронный сбор не запущен, либо при запущенном сборе данных, если синхронный сбор по цифровым линиям не разрешен.
Можно использовать маску, чтобы вывести только на часть выводов, оставив остальные неизменными, однако следует учесть, что после открытия связи с модулем необходимо сперва сделать вывод на все линии, после чего уже можно использовать маску при последующих вызовах.

Параметры:

hnd — Описатель модуля.

val — Младшая половина - выводимое значение, старшая - флаги из `t_l502_digout_word_flags`.

msk — Маска - указанные в маске биты не будут изменяться с предыдущего выведенного состояния (распространяется и на старшую половину val).

Возвращаемое значение:
Код ошибки.

4.3.4.3 Асинхронный ввод значений с цифровых входов.

Формат: `int32_t L502_AsyncInDig(t_l502_hnd hnd, uint32_t *din)`

Описание: Функция считывает текущее значение цифровых входов. При этом синхронный сбор цифровых входов не должен быть запущен (не разрешен поток `L502_STREAM_DIN`).
Так как модуль L502 не поддерживает аппаратно асинхронный ввод, то если на момент вызова этой функции не запущен синхронный ввод/вывод с помощью `L502_StreamsStart()`, то данная функция на время выполнения запускает синхронный сбор и останавливает его как только будут получено одно новое значение цифровых входов.

Параметры:

hnd — Описатель модуля.

din — При успешном выполнении в этой переменной возвращается текущее состояние цифровых входов. Действительны младшие 18 бит, старшие 14 - резерв. Резервные биты могут быть использованы в последующих версиях, не следует считать, что они всегда будут равны нулю!

Возвращаемое значение:
Код ошибки.

4.3.4.4 Асинхронный ввод одного кадра АЦП.

Формат: `int32_t L502_AsyncGetAdcFrame(t_l502_hnd hnd, uint32_t flags, uint32_t tout, double *data)`

Описание: Функция производит однократный ввод кадра в соответствии с заранее установленной логической таблицей. Частота сбора АЦП соответствует частоте установленной с помощью `L502_SetAdcFreq()`. Частота следования кадров значения не имеет. Сам кадр вводится синхронно, но при последовательном вызове `L502_AsyncGetAdcFrame()` для измерения нескольких кадров задержка между этими кадрами не определена.
Функция так же выполняет обработку принятых данных АЦП, аналогично `L502_ProcessAdcData()`, и принимает набор флагов, аналогичный `L502_ProcessAdcData()`.
Для работы этой функции не должен быть разрешен синхронный ввод АЦП и цифровых линий.
Так как аппаратно асинхронный ввод в плате отсутствует, то эта функция в случае не запущенного потока запускает его внутри себя, принимает один кадр данных и после этого останавливает синхронный сбор.

Параметры:

hnd — Описатель модуля.

flags — Флаги из t_l502_proc_flags

tout — Таймаут на выполнение функции в мс

data — Массив, в котором в случае успеха будут возвращены отсчеты кадра АЦП. Должен быть размером, достаточным для хранения отсчетов типа double в количестве, равном количеству установленных логических каналов в управляющей таблице АЦП.

Возвращаемое значение:
Код ошибки.

4.3.5 Функции для работы с синхронным потоковым вводом-выводом

4.3.5.1 Разрешение синхронных потоков на ввод/вывод.

Формат: `int32_t L502_StreamsEnable(t_l502_hnd hnd, uint32_t streams)`
Описание: Функция разрешает прием/передачу для указанных потоков Не указанные потоки сохраняют свое разрешенное или запрещенное состояние. Может вызываться как до `L502_Configure()`, так и после. Разрешенные потоки устанавливаются как правило до вызова `L502_StreamsStart()`. При желании в некоторых ситуациях можно изменять состав разрешенных потоков во время запущенного сбора данных, однако если эти потоки сильно различаются в частоте, то рассчитанные библиотекой значения буфера и шага прерывания могут не подходить для изменившихся значений (см. `Размер буфера и шаг прерываний для синхронного режима`)
Параметры: **hnd** — Описатель модуля. **streams** — Набор флагов `t_l502_streams`, указывающих, какие потоки должны быть разрешены.
Возвращаемое значение: Код ошибки.

4.3.5.2 Запрещение синхронных потоков на ввод/вывод.

Формат: `int32_t L502_StreamsDisable(t_l502_hnd hnd, uint32_t streams)`
Описание: Функция запрещает передачу синхронных данных для указанных потоков. Не указанные потоки сохраняют свое разрешенное или запрещенное состояние. Функция, противоположная по смыслу `L502_StreamsEnable()`.
Параметры: **hnd** — Описатель модуля. **streams** — Набор флагов `t_l502_streams`, указывающих, какие потоки должны быть запрещены.
Возвращаемое значение: Код ошибки.

4.3.5.3 Запуск синхронных потоков ввода/вывода.

Формат: `int32_t L502_StreamsStart(t_l502_hnd hnd)`
Описание: Функция запуска синхронных потоков данных. Все синхронные потоки тактируются от общей опорной частоты. Если был установлен внутренний старт синхронизации, то синхронизация потоков начнется при выполнении данной функции, в противном случае по данной функции модуль перейдет в состояние ожидания внешнего признака начальной синхронизации. Также функция осуществляет инициализацию канала DMA на ввод данных из платы, если был разрешен поток АЦП или синхронного ввода цифровых линий, и инициалзицую канала DMA на вывод, если был разрешен хотя бы один поток на вывод, но не была вызвана функция `L502_PreloadStart()` (однако в этом случае начало вывода не совпадет с началом ввода).
Параметры: **hnd** — Описатель модуля.
Возвращаемое значение: Код ошибки.

4.3.5.4 Останов синхронных потоков ввода/вывода.

Формат: `int32_t L502_StreamsStop(t_l502_hnd hnd)`
Описание: Функция останова синхронных потоков ввода/вывода данных. После выполнению этой функции модуль завершает генерацию опорной частоты синхронизации (или использовать внешную частоту синхронизации) и останавливает синхронную передачу данных.
Параметры: **hnd** — Описатель модуля.
Возвращаемое значение: Код ошибки.

4.3.5.5 Проверка, запущен ли синхронный ввод/вывод.

Формат: `int32_t L502_IsRunning(t_l502_hnd hnd)`
Описание: Функция проверяет запущен ли синхронный ввод вывод с помощью `L502_StreamsStart()` или какой-либо внутренней логикой в прошивки BlackFin. Если сбор данных не запущен, то функция возвращает ошибку `L502_ERR_STREAM_IS_NOT_RUNNING`, если запущен, то нулевой код ошибки
Параметры: **hnd** — Описатель модуля.
Возвращаемое значение: Код ошибки.

4.3.5.6 Чтение данных АЦП и цивровых входов из модуля.

Формат: `int32_t L502_Recv(t_l502_hnd hnd, uint32_t *buf, uint32_t size, uint32_t tout)`

Описание: Функция считывает данные от модуля, которые были переданы по DMA в буфер драйвера. Функция принимает отсчеты в специальном индексном формате, в котором содержится информация, что это за данные (значения цифровых входов или отсчеты АЦП) и дополнительная информация для АЦП (номер канала, режим). Для разбора полученных отсчетов используется функция `L502_ProcessData()`.
Если в буфере драйвера сейчас находится меньше отсчетов, чем было запрошено, то функция будет ожидать пока придет заданноеколичество данных или пока не истечет указанный таймаут. В последнем случае функция возвратит столько отсчетов, сколько было в буфере при истечении таймаута.
Количество готовых для чтения отсчетов в буфере драйвера можно при желании узнать с помощью функции `L502_GetRecvReadyCount()`.
До вызовов `L502_Recv()` синхронный поток сбора данных должен быть уже запущен с помощью `L502_StreamsStart()`.

Параметры:

hnd — Описатель модуля.

buf — Буфер, в которые будут сохранены отсчеты.

size — Количество считываемых отсчетов (32-битных слов).

tout — Таймаут на прием данных в мс.

Возвращаемое значение:
Если < 0 - код ошибки. Если $>= 0$ - количество считанных слов.

4.3.5.7 Передача потоковых данных ЦАП и цифровых выходов в модуль.

Формат: `int32_t L502_Send(t_l502_hnd hnd, const uint32_t *buf, uint32_t size, uint32_t tout)`
Описание: Функция записывает данные на передачу во внутренний буфер драйвера, после чего эти данные будут по DMA переданы в модуль. Данные должны быть в специальном формате, который определяет, что это за данные (цифровые выходы, канал ЦАП1 или канал ЦАП2). Подготовить данные в нужном формате можно с помощью `L502_PrepareData()`. Если буфер драйвера на передачу заполнен, то функция будут ждать пока он не освободится или пока не истечет указанный таймаут. Количество свободного места в буфере можно при желании узнать с помощью функции `L502_GetSendReadyCount()`. Возвращение означает что данные переданы в буфер драйвера, а не то что они уже дошли до модуля и выведены. Перед вызовом этой функции должна быть запущена предзагрузка данных на вывод с помощью `L502_PreloadStart()`.
Параметры: **hnd** — Описатель модуля. **buf** — Буфер со словами, которые необходимо передать модулю **size** — Количество передаваемых отсчетов (32-битных слов). **tout** — Таймаут на передачу (в буфер драйвера) данных в мс.
Возвращаемое значение: Если < 0 - код ошибки. Если $>= 0$ - количество записанных слов.

4.3.5.8 Обработка принятых отсчетов АЦП от модуля.

Формат: `int32_t L502_ProcessAdcData(t_l502_hnd hnd, const uint32_t *src, double *dest, uint32_t *size, uint32_t flags)`

Описание: Функция выполняет обработку отсчетов АЦП, прочитанных с помощью `L502_Recv()`. Функция проверяет служебную информацию из входного массива и переводит отсчеты АЦП либо в коды, либо в вольты (если указан флаг `L502_PROC_FLAGS_VOLT`).
Функция используется, когда не запущен синхронный ввод с цифровых линий и отсчеты АЦП являются единственными приходящими от модуля данными (если в принятом потоке будут другие данные - то они будут отброшены).
Если запущен синхронный ввод с цифровых линий, то следует использовать `L502_ProcessData()`, которая выделяет данные с цифровых линий в отдельный массив.

Параметры:

hnd — Описатель модуля.

src — Входной массив отсчетов, принятых с помощью `L502_Recv()`.

dest — Массив, в который будут сохранены преобразованные данные от АЦП.

size — На входе - количество слов в массиве src, на выходе - количество сохраненных преобразованных значений в массиве dest

flags — Набор флагов из `t_l502_proc_flags`

Возвращаемое значение:
Код ошибки.

4.3.5.9 Обработка принятых от модуля данных.

Формат: `int32_t L502_ProcessData(t_l502_hnd hnd, const uint32_t *src, uint32_t size, uint32_t flags, double *adc_data, uint32_t *adc_data_size, uint32_t *din_data, uint32_t *din_data_size)`

Описание: Функция выполняет обработку данных, прочитанных с помощью `L502_Recv()`. Функция проверяет служебную информацию из входного массива, разбивает данные на два массива - данные от АЦП, которые переводятся в тип double, и данные от синхронного цифрового ввода.
Данные от АЦП так же могут быть переведены в вольты. При этом данные АЦП приходят от модуля уже откалиброванными с помощью калибровочных коэффициентов, так как калибровка выполняется аппаратно. Если данные АЦП не переводятся в Вольты и при этом не были изменены заводские калибровочные коэффициенты, то возвращенное значение равное `L502_ADC_SCALE_CODE_MAX` соответствует напряжению равному максимальному для используемого диапазона.
Кроме того, функция разбирает сообщения, передаваемые в потоке данных (например, сообщение о переполнении буфера).

Параметры:

hnd — Описатель модуля.

src — Входной массив отсчетов, принятый с помощью `L502_Recv()`.

size — Количество отсчетов (32-битных слов) в массиве src.

flags — Набор флагов из `t_l502_proc_flags`, управляющих поведением функции. Может быть указано несколько флагов через логическое "ИЛИ".

adc_data — Массив, в который будут сохранены данные от АЦП, преобразованные в соответствии с указанными флагами. Может быть NULL, если не нужно сохранять данные от АЦП (тогда adc_data_size должен быть тоже NULL, или в переменной передан размер 0).

adc_data_size — На входе в данном параметре передается резмер буфера adc_data. Если данных от АЦП во входном массиве будет больше adc_data_size, то в adc_data будет сохранено только первые adc_data_size отсчетов. На выходе при успешном завершении функции в данную переменную записывается количество сохранных отсчетов АЦП. Указатель может быть равен NULL, если adc_data = NULL

din_data — Массив, в который будут сохранены отчеты с синхронного цифрового ввода. Младшие 18 битов соответствуют состояниям линий, старшие 14 - резерв. Может быть NULL, если не нужно сохранять данные от синхронного ввода.

din_data_size — Аналогично параметру adc_data_size в этом параметре передается размер буфера din_data в отсчетах, а на выходе сохраняется количество реально сохраненных отсчетов цифрового ввода. Может быть NULL, если din_data = NULL.

Возвращаемое значение:
Код ошибки.

4.3.5.10 Обработка принятых от модуля данных с пользовательскими данными.

Формат: `int32_t L502_ProcessDataWithUserExt(t_l502_hnd hnd, const uint32_t *src, uint32_t size, uint32_t flags, double *adc_data, uint32_t *adc_data_size, uint32_t *din_data, uint32_t *din_data_size, uint32_t *usr_data, uint32_t *usr_data_size)`

Описание: Функция аналогична L502_ProcessData(), но позволяет также выделить пользовательские данные из потока. Пользовательскими данными считаются все отсчеты, которые не являются данными АЦП, данными цифрового ввода или сообщениями. Пользовательские данные складываются без изменений в массив usr_data (если он не равен нулю). Данная функция предназначена в первую очередь для программистов, которые будут использовать модифицированную прошивку сигнального процессора BlackFin.

Параметры:

hnd — Описатель модуля.

src — Входной массив отсчетов, принятый с помощью L502_Recv().

size — Количество отсчетов (32-битных слов) в массиве src.

flags — Набор флагов из t_l502_proc_flags.

adc_data — Массив, в который будут сохранены данные от АЦП (см. L502_ProcessData()).

adc_data_size — см. L502_ProcessData()

din_data — Массив, в который будут сохранены отчеты с синхронного цифрового ввода. См. L502_ProcessData().

din_data_size — см. L502_ProcessData().

usr_data — Массив, в который будут сохранены пользовательские данные без изменения их формата.

usr_data_size — В этом параметре передается размер буфера usr_data а на выходе сохраняется количество реально сохраненных отсчетов пользовательских данных. Может быть NULL только если usr_data = NULL.

Возвращаемое значение:
Код ошибки.

4.3.5.11 Подготовка данных для вывода в модуль.

Формат: `int32_t L502_PrepareData(t_l502_hnd hnd, const double *dac1, const double *dac2, const uint32_t *digout, uint32_t size, int32_t flags, uint32_t *out_buf)`

Описание: Функция принимает данные из трех массивов - данные на цифровые выходы, отсчеты первого и второго канала ЦАП. В качестве массива может быть передан нулевой указатель, если данные из этого источника не требуются. Все используемые массивы должны быть одинакового размера и функция их равномерно перемешивает в общий поток, преобразуя в нужный для модуля формат.
Выходной массив должен будет содержать n*size отсчетов, где n - количество используемых входных массивов (от 1 до 3).
Значения цифровых выходов представляют собой 32-битные слова, младшие 16-бит которых определяют значения выводов, а старшие - флаги из `t_l502_digout_word_flags`, которые могут использоваться в частности для перевода одной (или обеих) из половин выводов в третье состояние.
В качестве значений ЦАП могут использоваться как коды, так и Вольты, в зависимости от переданных флагов, и к выводимым значениям могут быть применены калибровочные коэффициенты. Если используются коды ЦАП с включенной калибровкой, то код `L502_DAC_SCALE_CODE_MAX` определяет код, соответствующий +5V.

Параметры:

hnd — Описатель модуля.

dac1 — Входной массив отсчетов первого канала ЦАП или NULL, если не используется.

dac2 — Входной массив отсчетов второго канала ЦАП или NULL, если не используется.

digout — Входной массив со значениями цифровых выводов или NULL, если не используется.

size — Размер каждого из используемых входных массивов.

flags — Флаги, управляющие работой функции, из `t_l502_dacout_flags`.

out_buf — Выходной массив, в который будут сохранены сформированные отсчеты. Должен быть размера n*size (n - количество используемых входных массивов)

Возвращаемое значение:
Код ошибки.

4.3.5.12 Получить количество отсчетов в буфере потока на ввод.

Формат: `int32_t L502_GetRecvReadyCount(t_l502_hnd hnd, uint32_t *rdy_cnt)`
Описание: Функция возвращает количество отсчетов, которые были приняты из модуля во внутренний буфер в драйвере и готовы для считывания с помощью `L502_Recv()`. То есть если в `L502_Recv()` передать значение, которое вернула данная функция, то `L502_Recv()` вернет это количество данных без ожидания (так как они уже в буфере).
Параметры: **hnd** — Описатель модуля. **rdy_cnt** — Количество готовых к приему отсчетов.
Возвращаемое значение: Код ошибки.

4.3.5.13 Получить размер свободного места в буфере потока на вывод.

Формат: `int32_t L502_GetSendReadyCount(t_l502_hnd hnd, uint32_t *rdy_cnt)`
Описание: Функция возвращает количество отсчетов, соответствующее свободному месту в буфере драйвера на передачу в модуль. Это количество отсчетов может быть передано с помощью `L502_Send()` без ожидания.
Параметры: **hnd** — Описатель модуля. **rdy_cnt** — Количество свободных отсчетов в буфере на передачу.
Возвращаемое значение: Код ошибки.

4.3.5.14 Получить номер следующего ожидаемого логического канала АЦП для обработки.

Формат: `int32_t L502_GetNextExpectedLchNum(t_l502_hnd hnd, uint32_t *lch)`

Описание: Функция возвращает номер логического канала АЦП, который должен быть обработан первым при следующем вызове `L502_ProcessData()`/ `L502_ProcessAdcData()` в случае, если поток данных непрерывен.
По сути, это номер логического канала, слудущий за логическим каналом последнего обработанного до этого отсчета АЦП. Может быть использовано при обработке блоков данных не кратных целому количеству кадров. Если перед `L502_ProcessData()` вызывать данную функцию, то она вернет номер логического канала, соответствующий первому отсчету АЦП, обработаному последующим вызовом `L502_ProcessData()`.
Например, если установлено 7 логических каналов, а в `L502_ProcessData()` передано для обработки кратное 7 количество отсчетов, то последующий вызов `L502_GetNextExpectedLchNum()` вернет номер канала равный 0 (так как обработано целое число кадров и ожидается снова начало кадра). Если в `L502_ProcessData()` передан массив с 7*n + 5 отсчетами АЦП, то следующим ожидаемым каналом будет логический канал с номером 5 (обработаны каналы 0,1,2,3,4 из неполного кадра).

Параметры:

hnd — Описатель модуля.

lch — Номер логического канала (начиная с нуля).

Возвращаемое значение:
Код ошибки.

4.3.5.15 Начало подготовки вывода синхронных данных

Формат: `int32_t L502_PreloadStart(t_l502_hnd hnd)`

Описание: Функция должна вызываться перед началом предзагрузки потоковых синхронных данных на вывод. Для начала выдачи синхронных данных одновременно с началом синхронного ввода, к моменту начала сбора часть данных должна быть уже загружена в модуль до вызова `L502_StreamsStart()`.
Данная функция инициализирует DMA канал на передачу данных на вывод. После вызова этой функции можно загрузить часть данных на вывод с помощью `L502_Send()`.

Параметры:

hnd — Описатель модуля.

Возвращаемое значение:
Код ошибки.

4.3.5.16 Начало загрузки циклического сигнала на вывод

Формат: `int32_t L502_OutCycleLoadStart(t_l502_hnd hnd, uint32_t size)`

Описание: По вызову этой функции в драйвере выделяется место под циклический буфер на вывод. Должна вызываться перед загрузкой циклических данных с помощью `L502_Send()`.
Для успешного выполнения в драйвере должен быть свободный буфер (используется двойная буферизация) - т.е. функция не может быть вызвана сразу после предыдущего `L502_OutCycleSetup()`. Кроме того не должен был быть использован потоковый вывод.

Параметры:

hnd — Описатель модуля.

size — Количество отсчетов в выводимом циклическом сигнале суммарно для всех используемых каналов вывода.

Возвращаемое значение:
Код ошибки.

4.3.5.17 Установка ранее загруженного циклического сигнала на вывод

Формат: `int32_t L502_OutCycleSetup(t_l502_hnd hnd, uint32_t flags)`

Описание: По вызову этой функции ранее загруженный циклический буфер становится активным. Если синхронный ввод-вывод запущен (через `L502_StreamsStart()`), то по этой функции сигнал будет выдаваться на выход, иначе выдача начнется при запуске синхронного ввода-вывода.
Если до этого уже выводился циклический сигнал, то смена на новый произойдет в конце цикла предыдущего сигнала, если не указан флаг `L502_OUT_CYCLE_FLAGS_FORCE`.
Данная функция должна быть вызвана только после вызова `L502_OutCycleLoadStart()` и загрузки указанного в ней количества отсчетов в буфер!

Параметры:

hnd — Описатель модуля.

flags — Флаги из `t_l502_out_cycle_flags`.

Возвращаемое значение:
Код ошибки.

4.3.5.18 Останов вывода циклического сигнала

Формат: `int32_t L502_OutCycleStop(t_l502_hnd hnd, uint32_t flags)`
Описание: По вызову этой функции прекращается выдача ранее установленного циклического сигнала с помощью `L502_OutCycleSetup()`. Остановка осуществляется по после выдачи последнего семпла в периоде (но функция возвращает управление сразу, делая только запрос на останов), что позволяет знать какие значения останутся на выходах. При вызове же `L502_StreamsStop()` (или при запрещении всех потоков на вывод через `L502_StreamsDisable()`) останов всех потоков происходит сразу и точную точка не известна.
Параметры: **hnd** — Описатель модуля. **flags** — Флаги из `t_l502_out_cycle_flags`.
Возвращаемое значение: Код ошибки.

4.3.5.19 Установка размера буфера в драйвере для синхронного ввода или вывода.

Формат: `int32_t L502_SetDmaBufSize(t_l502_hnd hnd, uint32_t dma_ch, uint32_t size)`
Описание: Функция устанавливает размер буфера в драйвере, который используется для временного хранения данных на прием или на передачу. Предназначена для случаев, когда пользователя по каким-либо причинам не удовлетворяет рассчитываемое библиотекой значение по-умолчанию
Параметры: **hnd** — Описатель модуля. **dma_ch** — Определяет, устанавливается размер буфера на ввод или на вывод (значение из `t_l502_dma_ch`). **size** — Размер буфера в 32-битных отсчетах
Возвращаемое значение: Код ошибки.

4.3.5.20 Установка шага прерывания при передаче потока по DMA.

Формат: `int32_t L502_SetDmaIrqStep(t_l502_hnd hnd, uint32_t dma_ch, uint32_t step)`

Описание: Функция устанавливает шаг для генерации прерываний при передаче синхронного потока данных на ввод или на вывод. После передачи указанного количества отсчетов по DMA между буфером драйвера и модулем, модуль генерирует прерывание и драйвер обновляет счетчик переданных данных. Даннай функция предназначена для пользователей, которых не устроит автоматически рассчитываемое библиотекой значение.

Параметры:

hnd — Описатель модуля.

dma_ch — Определяет, шаг прерывания устанавливается на ввод или на вывод (значение из `t_l502_dma_ch`).

step — Шаг прерывания в 32-битных отсчетах

Возвращаемое значение:
Код ошибки.

4.3.6 Функции для работы с сигнальным процессором

4.3.6.1 Загрузка прошивки сигнального процессора BlackFin.

Формат: `int32_t L502_BfLoadFirmware(t_l502_hnd hnd, const char *filename)`

Описание: Функция загружает прошивку сигнального процессора из указанного файла в процессор и запускает ее, проверяет правильность загрузки путем получения версии прошивки (через специальную команду). Прошивка должна быть в бинарном формате LDR.

Параметры:

hnd — Описатель модуля.

filename — Имя файла с загружаемой прошивкой.

Возвращаемое значение:
Код ошибки.

4.3.6.2 Проверка, загружена ли прошивка BlackFIn.

Формат: `int32_t L502_BfCheckFirmwareIsLoaded(t_l502_hnd hnd, uint32_t *version)`

Описание: Функция передает команды процессору BlackFin для получения версии прошивки и ее состояния. Успешное выполнение команд свидетельствует о том, что в BlackFin загружена действительная прошивка. Кроме того прошивке передается информация о модуле (наличие опций, версия ПЛИС и т.д.) для внутреннего использования. В случае успеха модуль переводится в режим DSP.
Данная функция может служить для проверки была ли загружена прошивка раньше (чтобы не загружать повторно) или для проверки была ли она загружена через JTAG-интерфейс.

Параметры:

hnd — Описатель модуля.

version — Если указатель не нулевой, то в данной переменной возвращается версия прошивки BlackFin в случае успешной проверки.

Возвращаемое значение:
Код ошибки.

4.3.6.3 Чтение блока данных из памяти сигнального процессора.

Формат: `int32_t L502_BfMemRead(t_l502_hnd hnd, uint32_t addr, uint32_t *regs, uint32_t size)`

Описание: Функция считывает блок данных напрямую из памяти процессора. Может быть прочитаны данные, как из внутренней памяти (L1), так и из внешней SDRAM. Для выполнения этой функции в BlackFin должна быть загружена его прошивка.
Функция предназначена в первую очередь для пользователей, пишущих свою программу для сигнального процессора.

Параметры:

hnd — Описатель модуля.

addr — Адрес памяти, начиная с которого будет считан блок данных.

regs — Массив, в который будут сохранено прочитанное содержимое памяти.

size — Количество считываемых 32-битных слов.

Возвращаемое значение:
Код ошибки.

4.3.6.4 Запись блока данных в память сигнального процессора.

Формат: `int32_t L502_BfMemWrite(t_l502_hnd hnd, uint32_t addr, const uint32_t *regs, uint32_t size)`

Описание: Функция записывает блок данных напрямую в памяти процессора BlackFin. Блок данных должен быть всегда кратен 8 32-битным словам (32 байтам). Запись может осуществляться как во внутреннюю память (L1), так и во внешнюю SDRAM. Для выполнения этой функции в BlackFin должна быть загружена его прошивка.
Функция предназначена в первую очередь для пользователей, пишущих свою программу для сигнального процессора.
Следует быть осторожным, т.к. запись в область данных, используемую программой может привести к ее неработоспособности.

Параметры:

hnd — Описатель модуля.

addr — Адрес памяти, начиная с которого будет записан блок данных.

regs — Массив с данными для записи в сигнальный процессор.

size — Количество записываемых данных в 32-битных словах (должно быть кратно 8).

Возвращаемое значение:
Код ошибки.

4.3.6.5 Передача управляющей команды сигнальному процессору.

Формат: `int32_t L502_BfExecCmd(t_l502_hnd hnd, uint16_t cmd_code, uint32_t par, const uint32_t *snd_data, uint32_t snd_size, uint32_t *rcv_data, uint32_t rcv_size, uint32_t tout, uint32_t *recvd_size)`

Описание: Функция предназначена для передачи пользовательских управляющих команд процессору для пользователей, пишущих свою прошивку BlackFin.
Управление работой сигнального процессора штатным образом осуществляется через управляющие команды, которые записываются в специальную область памяти сигнального процессора. Сигнальный процессор обрабатывает команду и по завершению записывает в эту же область результат.
Команды делаться на стандартные, которые используются библиотекой l502api и пользовательские, которые может пользователь определять по своему усмотрению. Пользовательские команды начинаются с кода L502_BF_CMD_CODE_USER (0x80000000).

Параметры:

hnd — Описатель модуля.

cmd_code — Код команды - определяет, что за команда выполняется.

par — Параметр, передаваемый с командой (значение зависит от кода команды).

snd_data — Опциональные данные, передаваемые вместе с командой. Если данные не передаются, то должен передаваться нулевой указатель и snd_size = 0.

snd_size — Количество 32-битных слов, передаваемых в snd_data

rcv_data — Массив, в который будут переданы данные, возвращенные процессором по завершению команды. Если данные не должны возвращаться, то должен передаваться нулевой указатель, а rcv_size = 0.

rcv_size — Количество 32-битных слов, которое ожидается, что вернет процессор по выполнению команды. Массив rcv_data должен быть рассчитан на данное количество слов.

tout — Таймаут в течении которого будет ожидаться, когда процессор завершит выполнение команды. Функция возвратит управление либо по завершению команды, либо по таймауту.

recvd_size — Если не является нулевым указателем, то в эту переменную будет сохранено количество 32-битных слов, которое реально вернул процессор после выполнения команды (процессор имеет право вернуть меньше данных, чем запрашивалось в rcv_size).

Возвращаемое значение:
Код ошибки. Если процессор выполнил команду с ненулевым кодом завершения, то этот код и будет возвращен функцией.

4.3.7 Функции для работы с Flash-памятью модуля

4.3.7.1 Чтение блока данных из Flash-памяти.

Формат: `int32_t L502_FlashRead(t_l502_hnd hnd, uint32_t addr, uint8_t *data, uint32_t size)`

Описание: Функция считывает массив данных из Flash-памяти модуля в массив, переданный пользователем. Для считывания не нужно специальное разрешение - оно доступно всегда.

Параметры:

hnd — Описатель модуля.

addr — Адрес начала блока.

data — Массив, куда будут сохранены считанные данные (должен быть не меньше size байт).

size — Количество байт для чтения.

Возвращаемое значение:
Код ошибки.

4.3.7.2 Запись блока данных во Flash-память модуля.

Формат: `int32_t L502_FlashWrite(t_l502_hnd hnd, uint32_t addr, const uint8_t *data, uint32_t size)`

Описание: Функция записывает переданный массив данных во Flash-память модуля. Эта область должна быть предварительно стерта с помощью `L502_FlashErase()` и до начала изменения должна быть вызвана функция `L502_FlashWriteEnable()`, чтобы разрешить любое изменение содержимого Flash-памяти. Пользователю для записи доступны только первые `L502_FLASH_USER_SIZE` байт Flash-памяти.

Параметры:

hnd — Описатель модуля.

addr — Адрес начала блока.

data — Массив с записываемыми данными (должен быть не меньше size байт).

size — Количество байт для записи.

Возвращаемое значение:
Код ошибки.

4.3.7.3 Стирание блока во Flash-памяти.

Формат: `int32_t L502_FlashErase(t_l502_hnd hnd, uint32_t addr, uint32_t size)`
Описание: Функция стирает блок во Flash-памяти модуля (все ячейки будут читаться как 0xFF). Адрес и размер должны быть кратны 4096 байт! Перед вызовом этой функции запись должна быть разрешена запись в пользовательскую область с помощью `L502_FlashWriteEnable()`.
Параметры: **hnd** — Описатель модуля. **addr** — Адрес начала блока (должен быть кратен 4K). **size** — Количество байт для стирания (кратно 4K).
Возвращаемое значение: Код ошибки.

4.3.7.4 Разрешение записи в пользовательскую область Flash-памяти.

Формат: `int32_t L502_FlashWriteEnable(t_l502_hnd hnd)`
Описание: Функция разрешает запись в пользовательскую область Flash-памяти (первые `L502_FLASH_USER_SIZE` байт). Должна быть вызвана до того, как можно будет использовать `L502_FlashErase()` и `L502_FlashWrite()` для изменения содержимого пользовательской области памяти. После завершения изменения следует вызвать `L502_FlashWriteDisable()`.
Параметры: **hnd** — Описатель модуля.
Возвращаемое значение: Код ошибки.

4.3.7.5 Запрет записи в пользовательскую область Flash-памяти.

Формат: `int32_t L502_FlashWriteDisable(t_l502_hnd hnd)`
Описание: Функция запрещает запись в пользовательскую область Flash-памяти модуля (первые `L502_FLASH_USER_SIZE` байт). Должна быть вызвана после того, как нужные данные в пользовательской области были изменены с помощью `L502_FlashErase()` и `L502_FlashWrite()`, чтобы защитить пользовательскую область от случайной изменения в дальнейшем.
Параметры: **hnd** — Описатель модуля.
Возвращаемое значение: Код ошибки.

4.3.8 Дополнительные вспомогательные функции.

4.3.8.1 Получить версию библиотеки.

Формат: `uint32_t L502_GetDllVersion(void)`
Описание: Функция возвращает версию библиотеки l502api.dll. Версия возвращается в виде 32-битного числа. Строковое представление возвращенной версии - четыре числа, старшее соответствует старшему байту, младшее - младшему. Старший байт - можорная версия, второй по старшинству байт - минорная, третий - ревизия, четвертый - номер сборки (не используется - всегда 0)
Возвращаемое значение: 32-битное число, представляющее собой версию библиотеки

4.3.8.2 Получить версию драйвера устройства.

Формат: `int32_t L502_GetDriverVersion(t_l502_hnd hnd, uint32_t *ver)`
Описание: Функция возвращает версию драйвера, установленного для указанного открытого устройства. Версия возвращается в виде 32-битного числа. Строковое представление возвращенной версии - четыре числа, старшее соответствует старшему байту, младшее - младшему. Старший байт - можорная версия, второй по старшинству байт - минорная, третий - ревизия, четвертый - номер сборки (не используется - всегда 0). Это та версия, которая отображается в диспетчере устройств в Windows или с помощью modinfo в Linux.
Параметры: **hnd** — Описатель модуля. **ver** — 32-битное число, представляющее собой версию драйвера
Возвращаемое значение: Код ошибки.

4.3.8.3 Получение строки об ошибке.

Формат: `const char* L502_GetErrorString(int32_t err)`
Описание: Функция возвращает строку, соответствующую переданному коду ошибки. В настроящее время возвращается всегда русская версия строки (возможно в будущем будет возможность сменить язык глобальной функцией). Следует учесть, что в ОС Windows строка возвращается в стандартной для Windows кодировке CP1251, в то время как в Linux используется кодировка UTF-8.
Параметры: **err** — Код ошибки, для которого нужно вернуть строку.
Возвращаемое значение: Указатель на строку, соответствующую коду ошибки

4.3.8.4 Моргание светодиодом на передней панели.

Формат: `int32_t L502_LedBlink(t_l502_hnd hnd)`
Описание: При вызове этой функции, если не запущен синхронный ввод/вывод, происходит кратковременное затухание красного цвета светодиода на передней панели. Может быть использована для визуальной идентификации модуля после его открытия. При запущенном синхронном вводе/выводе светодиод на передней панели всегда горит зеленым цветом и данная функция не влияет на его состояние.
Параметры: **hnd** — Описатель модуля.
Возвращаемое значение: Код ошибки.

4.3.8.5 Установка подтягивающих резисторов на входных линиях.

Формат: `int32_t L502_SetDigInPullup(t_l502_hnd hnd, uint32_t pullups)`
Описание: Функция может использоваться для включения подтягивающих резисторов на цифровых входах. Отдельно можно задавать включены или отключены подтяжки на младшей половине цифровых линий, старшей половине, на линии SYN1 и на линии SYN2. На не указанных линиях подтягивающие резисторы будут отключены, если они были включены до этого. При включении питания все подтягивающие резисторы отключены.
Параметры: **hnd** — Описатель модуля. **pullups** — Флаги (из `t_l502_pullups`), определяющие, на каких линиях включены подтягивающие резисторы.
Возвращаемое значение: Код ошибки.