

Программное обеспечение

LComp

Руководство программиста

*Комплект ПО для разработки приложений (SDK)
Windows 98/Me/2000/XP*

L-264/L-305/L-1221/L-1250/L-1450/L-761/L-780/L-783/L-791/E14-440/E14-140/E20-10/E154



ЗАО "Л-КАРД"

117105, г. Москва, Варшавское ш., д. 5, корп. 4, стр. 2
тел.: (095) 785-95-25
факс: (095) 785-95-14

Адреса в Интернет:

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

E-Mail:

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

Представители в регионах:

Украина: HOLIT Data Systems, 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
Екатеринбург: Аск, www.ask.ru, 71-4444
Казань: ООО 'Шатл', shuttle@kai.ru, (8432) 38-1600

Предупреждение

Version 6.0r0

Copyright (c) 1998-2008 L-Card Ltd.

Интерфейс и функциональные возможности драйвера и библиотеки могут измениться в последующих версиях.

Учтите это при проектировании своих приложений.

Корректная поддержка плат с помощью это драйвера возможна только при соответствующих прошивках BIOS ADSP. Пока это выполняется для L-305 L-264 L-1250 L-1251 L-1221 L-761 L-780 L-783 L-1450 L-791 E14-440 E14-140 E20-10 E154.

Описание технологии

Данный релиз драйвера и библиотеки поддерживает следующие платы:

| | | | |
|---------|--------------------|---------|--|
| L032 | ISA | 2000/XP | Установите плату как L-1250 и используйте функции ввода/вывода в порты. |
| L305 | ISA | 2000/XP | |
| L264 | ISA | 2000/XP | |
| L1250 | ISA | 2000/XP | |
| L1450 | ISA | 2000/XP | |
| L1620 | ISA | 2000/XP | Установите плату как L-1250, при загрузке используйте БИОС от платы L-1620. Далее вызывайте функции L-1250. Разница только в формате данных. |
| L1251 | ISA with fast read | 2000/XP | |
| L1221 | ISA with fast read | 2000/XP | |
| L761 | PCI | 2000/XP | |
| L780 | PCI | 2000/XP | |
| L783 | PCI | 2000/XP | |
| L791 | PCI | 2000/XP | |
| E14-440 | USB | 2000/XP | |
| E14-140 | USB | 2000/XP | |
| E20-10 | USB | 2000/XP | |
| E154 | USB | 2000/XP | |

Принцип действия:

Устройство АЦП собирает данные в кольцевой буфер или FIFO, реализованный в ОЗУ сигнального процессора или микросхемы ПЛИС. При заполнении части буфера

генерируется прерывание. Драйвер устройства по этим прерываниям вычитывает данные и помещает их в большой кольцевой буфер, реализованный в ОЗУ компьютера. Большой кольцевой буфер драйвера доступен пользовательскому приложению - имеется указатель на начало этого буфера. Кроме этого пользователю доступен счетчик заполнения буфера (тоже посредством указателя). Используя этот счетчик, пользователь может забирать данные из правильной части кольцевого буфера (т.е. из той, в которую драйвер уже записал данные).

Приложение может:

- забирать данные из буфера для сохранения непрерывного потока данных;
- обрабатывать данные на месте - тогда старые данные будут замещаться новыми;

Связь драйвера с приложением возможна двумя способами:

- чтение счетчика заполнения буфера (циклическое заполнение буфера);
- ожидание сообщения о готовности буфера (однократное заполнение буфера);

Первый способ работает всегда, но требует ресурсов от компьютера при ожидании в цикле. Второй способ удобно использовать при осциллографическом режиме работы.

Использование такого режима работы - по прерываниям - обусловлено тем, что платы PCI L-761/780/L783 построены на микросхеме без поддержки BusMastering и для них такой способ ввода непрерывного потока данных является единственно возможным. При этом загрузка ЦПУ минимальна благодаря высокой скорости чтения на шине PCI. Для ISA плат при таком режиме работы удастся достичь наивысшей скорости передачи данных, однако для них загрузка процессора может достигать 50-90% на предельных скоростях, что не очень хорошо, но все же позволяет осуществлять непрерывный сбор данных на диск.

Для ISA плат поддерживающих ввод/вывод данных по DMA введен еще один режим. В этом режиме плата передает данные в компьютер по DMA в режиме автоинициализации. Буфер DMA маленький - до 2048 слов. При заполнении половины этого буфера плата генерирует прерывание, по которому драйвер копирует данные из буфера DMA в большой кольцевой буфер. Кроме этого возможен одновременный вывод на ЦАП платы данных из такого же маленького буфера - по DMA с автоинициализацией. В таком режиме работы скорость ввода данных не превышает 200 кГц, но при этом загрузка машины существенно ниже. Возможна работа двух плат на ввод по DMA.

Для платы L791 режим ввода и принцип сбора немного другие. Эта плата поддерживает режим ввода/вывода BusMaster. При этом основной принцип работы с библиотекой остается прежний - приложение забирает данные из кольцевого буфера в памяти компьютера. Только поступают они туда не по прерываниям от платы, а по BusMaster-каналу. Заполнение буфера контролируется также по чтению счетчика, только этот счетчик находится непосредственно в плате. Прерывания от платы тоже могут выступать сигналами готовности данных - пользователь может установить события на них и потом обрабатывать.

Для USB модулей реализован также принцип кольцевого буфера путем циклической перепосылки запросов на ввод данных. Работа с ЦАП в потоковом режиме для этих модулей не поддерживается. Она возможна при использовании библиотек не совместимых с LComp - LUsbApi. При этом используется другой интерфейс драйвера ldevusbu.sys.

Установка и настройка PCI плат

Первое правило при установке плат - необходимо убедиться, что компьютер настроен и все драйвера для него установлены. Особенно драйвера для чипсета. Также надо проверить, что в плате L-Card прошита самая свежая конфигурационная ПЗУ (см. каталог UTILS после установки драйверов). Как правило, для установки PCI платы необходимо просто вставить ее в компьютер и установить драйвера. После этого плата готова к использованию. Но возможны ситуации, когда это не так. Новые драйвера - это полноценные WDM драйвера способные работать в Windows с поддержкой ACPI и соответственно shared IRQ. Но возможны ситуации, когда другие устройства некорректно работают с ACPI и при этом разделяют ресурсы с платой L-Card. Тогда возможны зависания системы и частичная или полная неработоспособность платы L-Card или какой-то другой. Для решения этой проблемы, необходимо какими либо средствами исключить разделение ресурсов платой L-Card с другими устройствами компьютера. Разделение ресурсов также может понизить производительность системы и/или платы L-Card и тогда тоже желательно его ликвидировать. Ниже приведены пути отключения ACPI и исключения ситуации Shared IRQ для Windows 98 и 2000 (под Me и XP аналогично).

Общая часть:

- В БИОСе компьютера надо поискать ключ вида **Plug & Play OS Installed** и установить его в **No**. Это заставит именно БИОС производить первоначальную настройку PCI плат и Windows, потом будет использовать именно ее.
- В БИОСе компьютера найти, если есть ключ **ACPI function** и поставить его в **Disabled**.
- В БИОСе компьютера найти, если есть ключи вида **PCI Slot(0,1..) use IRQ** и поставить там фиксированное свободное прерывание вместо **Auto**.

Если это все проделать на компьютере до установки ОС, то при установке ОС она установится в варианте без поддержки ACPI. Если ОС уже стоит, то придется отключить ACPI в ОС.

Windows 98:

- Загрузиться в **Safe Mode**.
- Зайти в **Панель Управления**.
- Открыть иконку **Система**.
- В ней в **Диспетчере устройств** выбрать **Системные устройства**.
- Если у Вас там значится **Plug & play BIOS** и нет нигде слова ACPI, то и ACPI соответственно нет и ничего делать больше не надо.
- Иначе надо удалить все системные устройства, а **Системная кнопка ACPI** заменить на **Plug & play BIOS** с помощью обновления драйвера(выбрать все устройства).
- Перезагрузиться.
- Посмотреть в **Диспетчер устройств**. Там все должно быть в одном экземпляре и без восклицательных знаков. Если это не так, то надо удалять дублеров вместе с оригиналами и перезагружаться пока все не придет в норму. К сожалению, это процесс довольно трудно формализуется(PCI irq holder может быть много - это нормально).
- Как результат у Вас должна получиться нормальная система с **Plug & play BIOS** в системных устройствах.

Windows 2000:

- Зайти в **Панель Управления**.
- Открыть иконку **Система**.
- Выбрать закладку **Оборудование**.

- В ней в **Диспетчере устройств** выбрать **Компьютер**.
- Если у Вас там значится **Standart PC**, то ACPI соответственно нет и ничего делать больше не надо.
- Иначе надо сменить тип компьютера на **Standart PC** с помощью кнопки обновления драйвера (выбрать все устройства).
- Перезагрузиться.
- Посмотреть в **Диспетчер устройств**. Там все должно быть в одном экземпляре и без восклицательных знаков. Если это не так то надо удалять дублеров вместе с оригиналами и перезагружаться пока все не придет в норму. К сожалению, это процесс довольно трудно формализуется.
- Как результат у Вас должна получиться нормальная система с типом компьютера **Standart PC**.

Если ресурсы по прежнему разделяются, то можно попробовать переставить плату L-Card в другой слот PCI т.к. некоторые слоты PCI всегда разделяют прерывания с AGP слотом или дополнительными PCI слотами (если слотов >4).

Установка и настройка ISA плат

Первое правило при установке плат - необходимо убедиться, что компьютер настроен и все драйвера для него установлены. Особенно драйвера для чипсета. После установки платы в компьютер и инсталляции драйверов надо вызвать мастер установки новых устройств Windows:

- Далее выбрать **Добавить устройство**.
- Подождать пока система поищет в своей базе, и выбрать **Добавить новое устройство**.
- Выбрать **Выбор из списка**.
- Выбрать **L-Card ADC/DAC ISA boards**.
- Выбрать нужную плату - появится диалог настройки ресурсов.
- В диалоге выбрать нужную конфигурацию и настроить ресурсы в соответствии с установленными переключками. Не обязательно выбирать самую сложную конфигурацию. Можно ограничиться просто адресом и прерыванием, если требуется только ввод с АЦП и наоборот вывод на ЦАП будет работать только при конфигурации с ПДП для ЦАП. Некоторые платы требуют для работы АЦП наличие ПДП.
- Завершить работу мастера и перезагрузить компьютер - в списке устройств должна появиться установленная плата с выбранными ресурсами.

Использование реестра Windows

Напрямую с реестром библиотека и пользователь больше не работают. Информация о системных ресурсах назначенных плате извлекается драйверами при помощи PnP менеджера Windows. Она хоть и хранится в реестре, но в служебном формате. Для PCI плат эта информация видна на вкладке ресурсов для соответствующей платы. Для ISA плат она там устанавливается. Только тип платы и процессора DSP задаются посредством INF файла при установке плат. Получить их можно посредством вызова библиотечной функции GetSlotParam. Изменить соответственно - изменив INF файл.

Создание своего дистрибутива

Если Вы написали свое приложение и хотите оформить его в виде дистрибутива, то включите в него следующие файлы:

- Idevisa.sys - WDM драйвер для ISA плат - копировать в WINDOWS\SYSTEM32\DRIVERS;
- Idevpcim.sys - WDM драйвер для PCI плат L791 - копировать в WINDOWS\SYSTEM32\DRIVERS;
- Idevpci.sys - WDM драйвер для PCI плат - копировать в WINDOWS\SYSTEM32\DRIVERS;
- Idevusb.sys - WDM драйвер для USB модулей - копировать в WINDOWS\SYSTEM32\DRIVERS;
- Idevs.sys - поддерживающий драйвер - копировать в WINDOWS\SYSTEM32\DRIVERS;
- Icardisa.inf - INF файл для ISA плат - копировать в WINDOWS\INF;
- Icardpci.inf - INF файл для PCI плат - копировать в WINDOWS\INF;
- Idevpcim.inf - INF файл для PCI плат L791 - копировать в WINDOWS\INF;
- Idevusb.inf - INF файл для USB модулей - копировать в WINDOWS\INF;
- Icomp.dll - DLL библиотека для работы с платами - класть лучше всего в один каталог с приложением;

Все это будет работать под операционными системами Windows 2000/XP. Оригинальный скрипт инсталляции написан с помощью бесплатной программы NSIS(www.nullsoft.com) и прилагается.

Низкоуровневое API драйвера

Введение

Драйвер поддерживает некоторый низкоуровневый интерфейс, с помощью которого можно управлять платой без использования промежуточной DLL библиотеки. Все обращения к драйверу выполняются посредством вызова стандартной функции DeviceIoControl с передачей ей соответствующих параметров. Предварительно драйвер должен быть открыт с помощью CreateFile. При завершении работы с драйвером необходимо вызвать CloseHandle.

Еще не написано. См. исходники библиотеки, если есть необходимость.

Описание API DLL библиотеки

Введение

Библиотека функций создана для того, чтобы упростить связь приложений с драйверами. Ниже приведен полный список функций поддерживаемых библиотекой - фактически это файл `ifs_idev.h`. Работа с библиотекой построена на принципах COM-интерфейса, но это не COM в полном смысле этого слова. Для всех плат функции имеют одно и тоже название. Те из них, которые не поддерживаются конкретной платой, возвращают статус `L_NOTSUPPOTRED`. Трактовка параметров в некоторых функциях различается для конкретных типов плат, о чем написано в описании функции.

```
struct LUnknown
{
    IFC(HRESULT)    QueryInterface(const IID& iid, void** ppv) = 0;
    IFC(ULONG)      AddRef() = 0;
    IFC(ULONG)      Release() = 0;
};

struct IDaqLDevice:LUnknown
{
    IFC(ULONG)      inbyte ( ULONG offset, PCHAR data, ULONG len=1, ULONG key=0) = 0;
    IFC(ULONG)      inword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0) = 0;
    IFC(ULONG)      indword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) = 0;
    IFC(ULONG)      outbyte ( ULONG offset, PCHAR data, ULONG len=1, ULONG key=0) = 0;
    IFC(ULONG)      outword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0) = 0;
    IFC(ULONG)      outdword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) = 0;
    IFC(ULONG)      inmbyte ( ULONG offset, PCHAR data, ULONG len=1, ULONG key=0) = 0;
    IFC(ULONG)      inmword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0) = 0;
    IFC(ULONG)      inmdword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) = 0;
    IFC(ULONG)      outmbyte ( ULONG offset, PCHAR data, ULONG len=1, ULONG key=0) = 0;
    IFC(ULONG)      outmword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0) = 0;
    IFC(ULONG)      outmdword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) = 0;
    IFC(ULONG)      GetWord_DM(USHORT Addr, PUSHORT Data) = 0;
    IFC(ULONG)      PutWord_DM(USHORT Addr, USHORT Data) = 0;
    IFC(ULONG)      PutWord_PM(USHORT Addr, ULONG Data) = 0;
    IFC(ULONG)      GetWord_PM(USHORT Addr, PULONG Data) = 0;
    IFC(ULONG)      GetArray_DM(USHORT Addr, ULONG Count, PUSHORT Data) = 0;
    IFC(ULONG)      PutArray_DM(USHORT Addr, ULONG Count, PUSHORT Data) = 0;
    IFC(ULONG)      PutArray_PM(USHORT Addr, ULONG Count, PULONG Data) = 0;
    IFC(ULONG)      GetArray_PM(USHORT Addr, ULONG Count, PULONG Data) = 0;
    IFC(ULONG)      SendCommand(USHORT Cmd) = 0;
    IFC(ULONG)      PlataTest() = 0;
    IFC(ULONG)      GetSlotParam(PSLOT_PAR slPar) = 0;
    IFC(HANDLE)    OpenLDevice() = 0;
    IFC(ULONG)      CloseLDevice() = 0;

    IFC(ULONG)      SetParametersStream(PDAQ_PAR sp, ULONG *UsedSize, void** Data, void**
Sync, ULONG StreamId = L_STREAM_ADC) = 0;
    IFC(ULONG)      RequestBufferStream(ULONG *Size, ULONG StreamId = L_STREAM_ADC) = 0;
    IFC(ULONG)      FillDAQparameters(PDAQ_PAR sp) = 0;

    IFC(ULONG)      InitStartLDevice() = 0;
    IFC(ULONG)      StartLDevice() = 0;
    IFC(ULONG)      StopLDevice() = 0;
    IFC(ULONG)      LoadBios(char *FileName) = 0;

    IFC(ULONG)      IoAsync(PDAQ_PAR sp) =0;

    IFC(ULONG)      ReadPlataDescr(LPVOID pd) = 0;
    IFC(ULONG)      WritePlataDescr(LPVOID pd, USHORT Ena) = 0;
    IFC(ULONG)      ReadFlashWord(USHORT FlashAddress, PUSHORT Data) = 0;
    IFC(ULONG)      WriteFlashWord(USHORT FlashAddress, USHORT FlashWord) = 0;
```

```
IFC (ULONG) EnableFlashWrite (USHORT Flag) = 0;  
IFC (ULONG) EnableCorrection (USHORT Ena=1) = 0;  
IFC (ULONG) GetParameter (ULONG name, PULONG param) = 0;  
IFC (ULONG) SetParameter (ULONG name, PULONG param) = 0;  
};
```

CreateInstance

Функция создает объект для конкретного слота. Тип объекта определяется автоматически внутри этой функции.

Описание:

C: LUnknown* CreateInstance(ULONG Slot);

Pascal: function CreateInstance(Slot:ULONG): LUnknown;

Параметры:

ULONG Slot - номер слота, для которого создается объект (0,1 ...).

Возвращает:

- указатель на объект типа LUnknown или NULL в случае ошибки.

Реализована:

L7XX L1250 L1221 L305 L264 L1450 E14-440 E14-140 E20-10 E154

Примечание:

Дополнительную информацию о типе ошибки можно получить вызвав GetLastError. Если она вернула L_ERROR_NOBOARD значит в запрашиваемом слоте нет платы.

L_ERROR_INUSE - плата в этом слоте уже используется кем-то. L_ERROR -

возвращается когда невозможно создать объект. L_NOTSUPPORTED - если в слоте установлена плата, которая не поддерживается этой библиотекой. Пример использования этой функции при сканировании слотов см. L1221.DSK.

После вызова CreateInstance надо вызвать QueryInteface для получения указателя на интерфейс с которым дальше работать.

Подключение и работа с библиотекой (на CPP)

Общий принцип работы с библиотекой:

- Загрузить библиотеку с помощью LoadLibrary.
- Создать объект, связанный с конкретным виртуальным слотом при помощи вызова CreateInstance.
- Получить указатель на интерфейс вызвав QueryInterface
- Далее вызывать функции этого интерфейса.

Виртуальные слоты это собственно порядковые числа в названиях линков драйверов. Начинаются с 0 и так далее по порядку. Разделения на ISA,PCI или USB платы нет. Причем определить, что за плата соответствует конкретному слоту, можно только открыв его и прочитав информацию GetSlotParam и ReadPlataDescr (+ для L1450, E440, E2010 предварительно надо загрузить плату). GetSlotParam даст информацию о типе платы и назначенных ей ресурсах. Далее для PCI плат более подробную информацию даст ReadPlataDescr. Для L-1450, E440, E2010 также можно вызвать ReadPlataDescr, но предварительно в нее надо загрузить БИОС. Вызов ReadPlataDescr обязателен перед началом конфигурирования сбора данных поскольку там содержится информация о частоте кварца необходимая при расчетах временных параметров сбора данных. Также там хранятся калибровочный коэффициенты.

Для одной платы начало работы выглядит примерно так:

Файл create.h

```
#ifndef __TEST__
#define __TEST__

typedef IDaqLDevice* (*CREATEFUNCPTR)(ULONG Slot);

ULONG CallCreateInstance(char* name);

extern CREATEFUNCPTR CreateInstance;

#endif
```

Файл create.cpp

```
#include <windows.h>
#include <objbase.h>
#include "..\include\ioctl.h"
#include "..\include\ifc_ldev.h"
#include "..\include\create.h"

CREATEFUNCPTR CreateInstance;

ULONG CallCreateInstance(char* name)
{
    HINSTANCE hComponent = ::LoadLibrary(name);
    if(hComponent==NULL)
    {
        return 0;
    }

    CreateInstance = (CREATEFUNCPTR)::GetProcAddress(hComponent,"CreateInstance");
    if(CreateInstance==NULL)
    {
        return 0;
    }
    return 1;
}
```

Где-то в Вашем проекте (в компьютере одна плата L-783):

```
ULONG slot = 0;

trace("Get IUnknown pointer");
CallCreateInstance("lcomp.dll");
IUnknown* pIUnknown = CreateInstance(slot);
if(pIUnknown == NULL) { trace("CallCreateInstance failed"); return 1; }

trace("Get IDaqLDevice interface");
IDaqLDevice* pI;
HRESULT hr = pIUnknown->QueryInterface(IID_ILDEV, (void**)&pI);
if(!SUCCEEDED(hr)) { trace("Get IDaqLDevice failed"); return 1; }
trace("IDaqLDevice get success");
trace("Free IUnknown");
pIUnknown->Release();

pI->OpenLDevice(); // начало работы с платой
pI->LoadBios("l783");
...

pI->CloseLDevice(); // завершение работы
pI->Release();
```

Подробнее - смотрите примеры.

Подключение и работа с библиотекой (на Pascal/Delphi)

Общий принцип работы с библиотекой:

- Загрузить библиотеку с помощью LoadLibrary.
- Создать объект, связанный с конкретным виртуальным слотом при помощи вызова CreateInstance.
- Получить указатель на интерфейс, вызвав QueryInterface
- Далее вызывать функции этого интерфейса.

Виртуальные слоты это собственно порядковые числа в названиях линков драйверов. Начинаются с 0 и так далее по порядку. Разделения на ISA, PCI или USB платы нет. Причем определить, что за плата соответствует конкретному слоту, можно только открыв его и прочитав информацию GetSlotParam и ReadPlataDescr (+ для L1450, E440, E2010 предварительно надо загрузить плату). GetSlotParam даст информацию о типе платы и назначенных ей ресурсах. Далее для PCI плат более подробную информацию даст ReadPlataDescr. Для L-1450, E440, E2010 также можно вызвать ReadPlataDescr, но предварительно в нее надо загрузить БИОС. Вызов ReadPlataDescr обязателен перед началом конфигурирования сбора данных поскольку там содержится информация о частоте кварца необходимая при расчетах временных параметров сбора данных. Также там хранятся калибровочный коэффициенты..

Для одной платы начало работы выглядит примерно так:

Файл create.pas

```
unit Create;

interface

uses Windows, ioctl, ifc_ldev;

type
  TCreateInstance = function(Slot:ULONG): LUnknown; cdecl;

var
  hModule: THandle;
  CreateInstance: TCreateInstance;

  function CallCreateInstance(name:PChar):ULONG;

implementation

function CallCreateInstance(name:PChar):ULONG;
begin
  hModule:=0;
  hModule:=LoadLibrary(name);
  if(hModule=0) then
  begin
    Result:=0;
    Exit;
  end;
  @CreateInstance:=GetProcAddress(hModule, 'CreateInstance');
  if(@CreateInstance=nil) then
  begin
    Result:=0;
    Exit;
  end;
  Result:=1;
end;

end.
```

Где-то в Вашем проекте (в компьютере одна плата L-1450):

```
var
  pLDev: IDaqLDevice;
  pIUnknown: IUnknown;
  hr: Integer;
  dev: THandle;

  ...
  if(CallCreateInstance('lcomp.dll')=1) then
  begin
    {сообщение об успехе загрузки библиотеки}
  end;

  pIUnknown:=CreateInstance(0);
  hr := pIUnknown.QueryInterface(IID_ILDEV,pLDev);
  if(not Succeeded(hr)) then MessageBox(0,'Get interface failed','Error',MB_OK);
  pIUnknown.Release;
  dev:=pLDev.OpenLDevice;
  ...

  pLDev.CloseLDevice;
  pLDev.Release;
```

Подробнее - смотрите примеры.

Основные функции

OpenLDevice

Эту функцию необходимо вызвать перед началом работы с платой. Функция открывает соответствующий линк драйвера для платы.

Описание:

C: HANDLE OpenLDevice();
Pascal: function OpenLDevice:THandle;

Параметры:

Возвращает:

HANDLE - в случае успеха (дескриптор для работы с платой);
INVALID_HANDLE_VALUE - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Для каждой платы установленной в компьютер драйвер формирует линк по следующему принципу: LDev### (где ### - номер 1..). Номер в названии линка - это виртуальный слот. Номер виртуального слота, для которого будет выполнена функция OpenLDevice, передается как параметр в функции CreateInstance.

CloseLDevice

Эта функция вызывается при завершении работы с платой.

Описание:

C: `ULONG CloseLDevice();`

Pascal: `function CloseLDevice:ULONG;`

Параметры:

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

После вызова этой функции значение дескриптора устройства больше недействительно и не может использоваться при вызове функций библиотеки. Для платы L791 еще происходит удаление выделенной в функции RequesetBufferStream памяти для буфера ПДП.

SetParametersStream

Вызов этой функции настраивает плату АЦП/ЦАП на заданные параметры ввода или вывода данных, устанавливает размера кольцевого буфера на плате, задает интервал генерации прерываний (через столько-то точек), передает приложению адреса большого буфера и переменной синхронизации.

Описание:

C: `ULONG SetParametersStream(PDAQ_PAR sp, ULONG *UsedSize, void** Data, void** Sync, ULONG StreamId);`
Pascal: `function SetParametersStream(var sp:DAQ_PAR; var UsedSize:ULONG; out Data; out Sync; StreamId:ULONG):ULONG;`

Параметры:

PDAQ_PAR sp - структура, которая описывает параметры ввода или вывода данных (ADC_PAR,DAC_PAR или другая в зависимости от типа поля s_Type);
ULONG *UsedSize - переменная, в которой будет возвращено количество реально используемой памяти (в отсчетах АЦП);
void Data** - переменная, в которой будет возвращен адрес начала большого буфера;
void Sync** - переменная, в которой будет возвращен адрес переменной синхронизации;
ULONG StreamId - дескриптор потока (L_STREAM_ADC, L_STREAM_DAC или другой);

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
PDAQ_PAR sp - структура, которая описывает параметры ввода или вывода данных. У этой структуры, если она не NULL, обновляются поля с учетом возможностей платы.

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Принцип быстрого и непрерывного ввода или вывода данных с платы в драйверах всегда одинаков. Различается только направление передачи данных. Поэтому было введено понятие потоков данных. Поток создается 3 функциями - RequestBufferStream FullDAQparameters SetParametersStream. Фактически это большой кольцевой буфер и структура, описывающая параметры сбора данных. Поток может быть с АЦП, на ЦАП, на цифровые линии, с цифровых линий или какой-то нестандартный реализованный в драйвере платы. Интерфейс при этом не меняется. Чтобы различать потоки служит переменная StreamId - это некоторая константа, определенная в заголовочных файлах.

RequestBufferStream

Функция служит для выделения памяти под большой кольцевой буфер.

Описание:

C: `ULONG RequestBufferStream(ULONG *Size, ULONG StreamId);`

Pascal: `function RequestBufferStream(var Size:ULONG; StreamId:ULONG):ULONG;`

Параметры:

ULONG *Size - размер большого буфера в USHORT;

ULONG StreamId - дескриптор потока (L_STREAM_ADC, L_STREAM_DAC или другой);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Size - возвращается количество реально выделенной памяти;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Выделяет память в ОЗУ компьютера под большой кольцевой буфер. Память выделяется с выравниванием размера на 4096 байт. Принцип быстрого и непрерывного ввода или вывода данных с платы в драйверах всегда одинаков. Различается только направление передачи данных. Поэтому было введено понятие потоков данных. Поток создается 3 функциями - RequestBufferStream SetParametersStream FullDAQparameters. Фактически это большой кольцевой буфер и структура, описывающая параметры сбора данных. Поток может быть с АЦП, на ЦАП, на цифровые линии, с цифровых линий или какой-то нестандартный реализованный в драйвере платы. Интерфейс при этом не меняется. Чтобы различать потоки служит переменная StreamId - это некоторая константа, определенная в заголовочных файлах.

Для платы L791 выделяется всегда 512*1024 слов. Это буфер отсчетов ЦАП и АЦП - по 128K 32 битных отсчетов соответственно. Удаляется этот буфер при вызове CloseLDevice .

InitStartLDevice

Функция инициализирует внутренние переменные драйвера перед началом сбора.

Описание:

C: `ULONG InitStartLDevice();`
Pascal: `function InitStartLDevice:ULONG;`

Параметры:

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Надо вызывать перед вызовом функции StartLDevice.

StartLDevice

Функция запускает сбор данных с платы в большой кольцевой буфер.

Описание:

C: `ULONG StartLDevice();`

Pascal: `function StartLDevice:ULONG;`

Параметры:

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

После выполнения функции можно переходить к откачиванию данных из буфера. При этом необходимо следить за синхронизацией поступления данных и их откачки.

StopLDevice

Функция останавливает сбор данных с платы в большой кольцевой буфер.

Описание:

C: `ULONG StopLDevice();`

Pascal: `function StopLDevice:ULONG;`

Параметры:

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

После остановки данные в буфере соответствуют последним данным, полученным от платы. Их можно обрабатывать любым способом. Необходимо только учитывать, что остановка могла произойти в любом месте этого буфера и гарантировать целостность можно только той части буфера, на готовность которой указывала переменная синхронизации.

Для L791 не следует пытаться разрешить режим BusMaster после выполнения команды StopLDevice. При старте буфер данных блокируется в ОЗУ компьютера, формируется таблица адресов и передается в плату. При остановке блокировка снимается и разрешение на передачу данных вызовет повреждение операционной системы.

LoadBios

Загрузка BIOS в плату.

Описание:

C: `ULONG LoadBios(char *FileName);`

Pascal: `function LoadBios(FileName:PChar):ULONG;`

Параметры:

char *FileName - имя файла прошивки BIOS без расширения (lbios009);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 E14-440 E20-10

Примечание:

В модуль E20-10 загружается прошивка ПЛИС **e2010.pld**, указывать ее нужно также без расширения. У L791 нет загружаемого BIOSа. E140 также не требует загрузки BIOS.

GetWord_DM

Читает слово из памяти данных DSP.

Описание:

C: `ULONG GetWord_DM(USHORT Addr, PUSHORT Data);`

Pascal: `function GetWord_DM(Addr:USHORT; var Data:USHORT):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

PUSHORT Data - возвращаемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PUSHORT Data - возвращаемые данные;

Реализована:

L7XX L1250 L305 L264 L1450 L1221 E14-440 E14-140 E154

Примечание:

PutWord_DM

Записывает слово в память данных DSP.

Описание:

C: `ULONG PutWord_DM(USHORT Addr, USHORT Data);`

Pascal: `function PutWord_DM(Addr:USHORT; Data:USHORT):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

USHORT Data - записываемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L305 L264 L1450 L1221 E14-440 E14-140 E154

Примечание:

GetWord_PM

Читает слово из памяти программ DSP.

Описание:

C: `ULONG GetWord_PM(USHORT Addr, PULONG Data);`

Pascal: `function GetWord_PM(Addr:USHORT; var Data:ULONG):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

PULONG Data - возвращаемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PULONG Data - возвращаемые данные;

Реализована:

L7XX L1450 L1221 E14-440 E14-140 E154

Примечание:

PutWord_PM

Читает слово из памяти программ DSP.

Описание:

C: `ULONG PutWord_PM(USHORT Addr, ULONG Data);`

Pascal: `function PutWord_PM(Addr:USHORT; Data:ULONG):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

ULONG Data - записываемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L1221 E14-440 E14-140 E154

Примечание:

GetArray_DM

Читает массив слов из памяти данных DSP.

Описание:

C: `ULONG GetArray_DM(USHORT Addr, ULONG Count, PUSHORT Data);`

Pascal: `function GetArray_DM(Addr:USHORT; Count:ULONG; var Data:USHORT):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

ULONG Count - размер массива в словах;

PUSHORT Data - возвращаемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PUSHORT Data - возвращаемые данные;

Реализована:

L7XX L1450 L1221 E14-440 E14-140 E154

Примечание:

PutArray_DM

Записывает массив слов в память данных DSP.

Описание:

C: `ULONG PutWord_DM(USHORT Addr, ULONG Count, USHORT Data);`

Pascal: `function PutWord_DM(Addr:USHORT; Count:ULONG;`

`Data:USHORT):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

ULONG Count - размер массива в словах;

USHORT Data - записываемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L1221 E14-440 E14-140 E154

Примечание:

GetArray_PM

Читает массив слов из памяти программ DSP.

Описание:

C: `ULONG GetArray_PM(USHORT Addr, ULONG Count, PULONG Data);`

Pascal: `function GetArray_PM(Addr:USHORT; Count:ULONG; var Data:ULONG):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

ULONG count - размер массива в двойных словах;

PULONG Data - возвращаемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PULONG Data - возвращаемые данные;

Реализована:

L7XX L1450 L1221 E14-440

Примечание:

PutArray_PM

Записывает массив слов в память программ DSP.

Описание:

C: `ULONG PutArray_PM(USHORT Addr, ULONG Count, ULONG Data);`

Pascal: `function PutArray_PM(Addr:USHORT; Count:ULONG;`

`Data:ULONG):ULONG;`

Параметры:

USHORT Addr - адрес переменной;

ULONG Count - размер массива в двойных словах;

ULONG Data - записываемые данные;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L1221 E14-440

Примечание:

SendCommand

Посылает выбранную команду DSP.

Описание:

C: `ULONG SendCommand(USHORT Cmd) ;`

Pascal: `function SendCommand(Cmd:USHORT):ULONG;`

Параметры:

USHORT Cmd - код команды;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L264 L305 L1450 L1221 E14-440 E14-140 E154

Примечание:

PlataTest

Тест на наличие платы и успешную загрузку.

Описание:

C: `ULONG PlataTest();`

Pascal: `function PlataTest:ULONG;`

Параметры:

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Для L791, E14-140 E154и E20-10 это просто заглушка всегда возвращающая успех.

IoAsync

Функция для асинхронных операций ввода/вывода (чтение данных с АЦП, вывод данных на ЦАП, работа с цифровыми линиями).

Описание:

C: `ULONG IoAsync(PDAQ_PAR sp);`
Pascal: `function IoAsync(var sp:DAQ_PAR):ULONG;`

Параметры:

PDAQ_PAR sp - структура с параметрами запроса;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PDAQ_PAR sp - структура с результатами запроса (если была операция ввода данных);

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Эта функция реализует все асинхронные операции ввода/вывода (типа одиночного ввода данных).

• Плата L-1450

- Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
s_Type -L_ASYNC_ADC_INP
Chn[0] - логический номер канала;
Результат в **Data[0]**.
- Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
- Для ввода одного отсчета с TTL линий:
s_Type -L_ASYNC_TTL_INP
Data[0] - введенные данные;
- Разрешить/запретить цифровые линии:
s_Type -L_ASYNC_TTL_CFG
Mode - (0/1);
- Для вывода одного отсчета на ЦАП:
s_Type -L_ASYNC_DAC_OUT
Mode - (0/1) номер ЦАП;
Data[0] - данные для ЦАП;

• Платы L-1250/L-264/L-305/L-1620

- Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
s_Type -L_ASYNC_ADC_INP
Chn[0] - логический номер канала;
Результат в **Data[0]**.
- Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
Mode - (0 - for multiplexer ttl(SETCHANNEL in fact))/(1- dsp out)
- Для ввода одного отсчета с TTL линий:

- s_Type** -L_ASYNC_TTL_INP
 - Mode** - (0 - прямые линии)/(1 - линии через DSP)
 - Data[0]** - введенные данные;
 - Для вывода одного отсчета на ЦАП:
 - s_Type** -L_ASYNC_DAC_OUT
 - Mode** - (0 - синхронный/1 - асинхронный) вывод на ЦАП;
 - Data[0]** - данные для ЦАП;
 - Для установки номера ЦАП и режима вывода:
 - s_Type** -L_ASYNC_DAC_CFG
 - Mode** - (0 - выкл режим одновременного вывода)(1 - вкл режим одновременного вывода на ЦАП со вводом с АЦП);
 - Chn[0]** - номер ЦАП(0/1);
- **Плата L-1221**
 - Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
 - s_Type** -L_ASYNC_ADC_INP
 - Chn[0]** - логический номер канала;
 - Результат в **Data[0]**.
 - Для вывода одного отсчета на TTL линии:
 - s_Type** -L_ASYNC_TTL_OUT
 - Data[0]** - данные для вывода;
 - Для ввода одного отсчета с TTL линий:
 - s_Type** -L_ASYNC_TTL_INP
 - Data[0]** - введенные данные;
 - Для настройки цифровых линий:
 - s_Type** -L_ASYNC_TTL_CFG
 - Mode** - маска направлений (0XXX 0000)
- **Платы L-761/L-780/L-783**
 - Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
 - s_Type** -L_ASYNC_ADC_INP
 - Chn[0]** - логический номер канала;
 - Результат в **Data[0]**.
 - Для вывода одного отсчета на TTL линии:
 - s_Type** -L_ASYNC_TTL_OUT
 - Data[0]** - данные для вывода;
 - Для ввода одного отсчета с TTL линий:
 - s_Type** -L_ASYNC_TTL_INP
 - Data[0]** - введенные данные;
 - Для вывода одного отсчета на ЦАП:
 - s_Type** -L_ASYNC_DAC_OUT
 - Mode** - номер ЦАП (0/1);
 - Data[0]** - данные для ЦАП;
 - Разрешить/запретить цифровые линии: (только L780C)
 - s_Type** -L_ASYNC_TTL_CFG
 - Mode** - (0/1);
- **Платы L-791**
 - Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
 - s_Type** -L_ASYNC_ADC_INP
 - Chn[0]** - логический номер канала;
 - Результат в **Data[0]**.
 - Для вывода отсчета на ЦАП надо заполнить структуру ASYNC_PAR так:
 - s_Type** -L_ASYNC_DAC_OUT
 - Chn[0]** -(0/1) выводить на 0 канал
 - Chn[1]** -(0/1) выводить на 1 канал
 - Data[0]** — данные для 0 канала
 - Data[1]** — данные для 1 канала

- Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
- Для ввода одного отсчета с TTL линий:
s_Type -L_ASYNC_TTL_INP
Data[0] - введенные данные;
- Разрешить/запретить цифровые линии:
s_Type -L_ASYNC_TTL_CFG
Mode – (0/1) разрешить/запретить цифровые линии;
- **Модуль E440**
 - Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
s_Type -L_ASYNC_ADC_INP
Chn[0] - логический номер канала;
Результат в **Data[0]**.
 - Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
 - Для ввода одного отсчета с TTL линий:
s_Type -L_ASYNC_TTL_INP
Data[0] - введенные данные;
 - Для вывода одного отсчета на ЦАП:
s_Type -L_ASYNC_DAC_OUT
Mode - номер ЦАП (0/1);
Data[0] - данные для ЦАП;
 - Разрешить/запретить цифровые линии:
s_Type -L_ASYNC_TTL_CFG
Mode - (0/1);
- **Модуль E140**
 - Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
s_Type -L_ASYNC_ADC_INP
Chn[0] - логический номер канала;
Результат в **Data[0]**.
 - Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
 - Для ввода одного отсчета с TTL линий:
s_Type -L_ASYNC_TTL_INP
Data[0] - введенные данные;
 - Для вывода одного отсчета на ЦАП:
s_Type -L_ASYNC_DAC_OUT
Mode - номер ЦАП (0/1);
Data[0] - данные для ЦАП;
 - Разрешить/запретить цифровые линии:
s_Type -L_ASYNC_TTL_CFG
Mode - (0/1);
- **Модуль E2010**
 - Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
 - Для ввода одного отсчета с TTL линий:
s_Type -L_ASYNC_TTL_INP
Data[0] - введенные данные;
 - Для вывода одного отсчета на ЦАП:
s_Type -L_ASYNC_DAC_OUT
Mode - номер ЦАП (0/1);
Data[0] - данные для ЦАП;

- Разрешить/запретить цифровые линии:
s_Type -L_ASYNC_TTL_CFG
Mode - (0/1);
- **Модуль E154**
 - Для ввода одного отсчета с АЦП надо заполнить структуру ASYNC_PAR так:
s_Type -L_ASYNC_ADC_INP
Chn[0] - логический номер канала;
Результат в **Data[0]**.
 - Для вывода одного отсчета на TTL линии:
s_Type -L_ASYNC_TTL_OUT
Data[0] - данные для вывода;
 - Для ввода одного отсчета с TTL линий:
s_Type -L_ASYNC_TTL_INP
Data[0] - введенные данные;
 - Для вывода одного отсчета на ЦАП:
s_Type -L_ASYNC_DAC_OUT
Data[0] - данные для ЦАП;
 - Разрешить/запретить цифровые линии:
s_Type -L_ASYNC_TTL_CFG
Mode - (0/1);

EnableCorrection

Включает/выключает режим коррекции. Сама загружает коэффициенты в плату.

Описание:

C: `ULONG EnableCorrection(USHORT Ena=1);`
Pascal: `function EnableCorrection(Ena:USHORT):ULONG;`

Параметры:

USHORT Ena - новое значение переменной разрешения/запрещения коррекции (1/0);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L1221 E14-440

Примечание:

FillDAQparameters

Заполняет внутреннюю структуру параметров сбора данных значениями из структуры ADC_PAR, DAC_PAR или другой в зависимости от типа поля s_Type.

Описание:

C: ULONG FillDAQparameters(PDAQ_PAR sp);

Pascal: function FillDAQparameters(var sp:DAQ_PAR):ULONG;

Параметры:

PDAQ_PAR sp - структура с параметрами сбора данных;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PDAQ_PAR sp - в структуре обновлены поля Rate, Kadr, NCh с учетом возможностей платы.

Реализована:

L7XX L1250 L305 L264 L1450 L1221 L791 E14-440 E14-140 E20-10 E154

Примечание:

ReadPlataDescr

Чтение пользовательского Flash.

Описание:

C: ReadPlataDescr(LPVOID pd);

Pascal: function ReadPlataDescr(var pd):ULONG;

Параметры:

LPVOID pd - указатель на структуру PLATA_DESCR_U;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

LPVOID pd - указатель на заполненную структуру PLATA_DESCR_U;

Реализована:

L7XX L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Для каждой платы определены свои частные структуры флеша и для удобства они сгруппированы в union PLATA_DESCR_U/U2. U2 появился из-за модуля E2010 у которого флеш больше. Внутри библиотеки все хранится в union PLATA_DESCR_U2.

WritePlataDescr

Запись пользовательского Flash.

Описание:

C: `ULONG WritePlataDescr(LPVOID pd, USHORT Ena);`

Pascal: `function WritePlataDescr(var pd; Ena:USHORT):ULONG;`

Параметры:

LPVOID pd - указатель на структуру PLATA_DESCR_U;

USHORT Ena - разрешение(1) / запрещение(0) записи служебной части пользовательского Flash;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

ReadFlashWord

Чтение слова из пользовательского Flash.

Описание:

C: `ULONG ReadFlashWord(USHORT FlashAddress, PUSHORT Data);`

Pascal: `function ReadFlashWord(FlashAddress:USHORT; var Data:USHORT):ULONG;`

Параметры:

USHORT FlashAddress - адрес, с которого читать;

PUSHORT Data - прочитанное слово;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PUSHORT Data - прочитанное слово;

Реализована:

L7XX L1450 L1221 L791 E14-440 E14-140

Примечание:

Для L1221 и L791 читает байт. Старший байт слова не используется.

WriteFlashWord

Запись слова в пользовательский Flash.

Описание:

C: `ULONG WriteFlashWord(USHORT FlashAddress, USHORT Data);`

Pascal: `function WriteFlashWord(FlashAddress:USHORT; Data:USHORT):ULONG;`

Параметры:

USHORT FlashAddress - адрес, по которому писать;

USHORT Data - записываемое слово;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L1221 L791 E14-440 E14-140

Примечание:

Для L1221 записывает младший байт из слова. Для L791 записывает младший байт из слова в буфер микросхемы. Реальная запись происходит по команде EnableFlashWrite в указанную страницу.

EnableFlashWrite

Разрешение записи в пользовательский Flash.

Описание:

C: `ULONG EnableFlashWrite(USHORT Flag);`

Pascal: `function EnableFlashWrite(Flag:USHORT):ULONG;`

Параметры:

USHORT Flag - разрешение (1) / запрещение (0) записи во Flash;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Для платы L791 Flag это номер страницы в которую будут записаны данные из буфера микросхемы. **0** - это страница со структурой PLATA_DESCR.

GetParameter

Функция служит для считывания разнообразных параметров с платы. Замещает некоторые простые функции из старых API.

Описание:

C: `ULONG GetParameter(ULONG name, PULONG param);`

Pascal: `function GetParameter(name:ULONG; var param:ULONG):ULONG;`

Параметры:

ULONG name - условное название параметра (см примечание);

PULONG param - значение параметра;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PULONG param - значение параметра;

Реализована:

L1221 L1250 L264 L305

Примечание:

- Для 1221 есть следующие параметры:
 - L1221_EXT_COUNTER - считать значение внешнего счетчика (ячейки 0x3D5C 0x3D5B);
 - L1221_OVERFLOW - считать значение ячейки переполнения (0x3D4E);
 - L1221_BIT_RES - считать ячейку с разрядностью АЦП (0x3D58);
- Для 1250/264/305:
 - L1251_MEM_STATE - возвращает размер установленной внешней памяти данных: 0 - нет; 1 - 8K; 2 - 32K; 3 - 128K;
 - L1251_MEM_PM_STATE - тестирует наличие внешней памяти программ на плате: 0 - нет; 1 - есть;

SetParameter

Функция служит для записи разнообразных параметров в плату. Замещает некоторые простые функции из старых API.

Описание:

C: `ULONG SetParameter(ULONG name, PULONG param);`

Pascal: `function SetParameter(name:ULONG; var param:ULONG):ULONG;`

Параметры:

ULONG name - условное название параметра (см примечание);

PULONG param - значение параметра;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PULONG param - значение параметра;

Реализована:

L1221

Примечание:

Для 1221 есть следующие параметры:

- L1221_BIT_RES - считать ячейку с разрядностью АЦП (0x3D58);

SetLDeviceEvent

Функция служит для установки события в драйвере. Работа события облегчает ожидание готовности данных от платы при однократном заполнении буфера. Также позволяет более удобно получать информацию о других процессах в плате.

Описание:

C: `ULONG SetLDeviceEvent(HANDLE hEvent, ULONG EventId=L_EVENT_ADC);`
Pascal: `function SetParameter(hEvent:THandle; EventId:ULONG):ULONG;`

Параметры:

ULONG hEvent - условное название параметра (см примечание);
ULONG EventId – идентификатор события на который установлен event;

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Для EventId есть следующие значения:

- L_EVENT_ADC_BUF 1 – событие по заполнении буфера АЦП;
- L_EVENT_DAC_BUF 2 - событие при работе с буфером ЦАП (L1450, L780M);
- L_EVENT_ADC_OVF 3 - L791 -см описание по генерации прерываний от платы;
- L_EVENT_ADC_FIFO 4 - L791 -см описание по генерации прерываний от платы;
- L_EVENT_DAC_USER 5 - L791 -см описание по генерации прерываний от платы;
- L_EVENT_DAC_UNF 6 - L791 -см описание по генерации прерываний от платы;
- L_EVENT_PWR_OVR 7 - L791 -см описание по генерации прерываний от платы;

Вспомогательные функции

GetSlotParam

Функция возвращает информацию для указанного виртуального слота.

Описание:

C: `ULONG GetSlotParam(PSLOT_PAR sIPar);`

Pascal: `function GetSlotParam(var sIPar:SLOT_PAR):ULONG;`

Параметры:

PSLOT_PAR sIPar - переменная, в которой будут возвращены параметры;

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PSLOT_PAR sIPar - параметры установленные для данного слота;

Реализована:

L7XX L1250 L1221 L305 L264 L1450 L791 E14-440 E14-140 E20-10 E154

Примечание:

Функции для работы с портами ввода/вывода

inbyte

Ввод байта из I/O порта.

Описание:

C: `ULONG inbyte (ULONG offset, PCHAR data, ULONG len=1, ULONG key=0);`
Pascal: `function inbyte (offset:ULONG; var data:UCHAR; len:ULONG;
key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;
PCHAR data - массив, в который будут занесены прочитанные данные;
ULONG len - размер массива в байтах;
ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
PCHAR data - прочитанные данные;

Реализована:

L7XX L1221 L1250 L264 L305 L1450

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

inword

Ввод слова из I/O порта.

Описание:

C: **ULONG inword (ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0);**
Pascal: **function inword (offset:ULONG; var data:USHORT; len:ULONG; key:ULONG):ULONG;**

Параметры:

ULONG offset - смещение порта относительно базового адреса;
PUSHORT data - массив, в который будут занесены прочитанные данные;
ULONG len - размер массива в байтах;
ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
PUSHORT data - прочитанные данные;

Реализована:

L7XX L1221 L1250 L264 L305 L1450

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

indword

Ввод двойного слова из I/O порта.

Описание:

C: `ULONG indword (ULONG offset, PULONG data, ULONG len=4, ULONG key=0);`
Pascal: `function indword (offset:ULONG; var data:ULONG; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;
PULONG data - массив, в который будут занесены прочитанные данные;
ULONG len - размер массива в байтах;
ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
PULONG data - прочитанные данные;

Реализована:

L7XX L1221 L1250 L264 L305 L1450

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

outbyte

Вывод байта в I/O порт.

Описание:

C: `ULONG outbyte (ULONG offset, PCHAR data, ULONG len=1, ULONG key=0);`
Pascal: `function outbyte (offset:ULONG; var data:UCHAR; len:ULONG;
key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;
PCHAR data - массив данных;
ULONG len - размер массива в байтах;
ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

outword

Вывод слова в I/O порт.

Описание:

C: `ULONG outword (ULONG offset, PUSHORT data, ULONG len=1, ULONG key=0);`

Pascal: `function outword (offset:ULONG; var data:USHORT; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PUSHORT data - массив данных;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

outdword

Вывод двойного слова в I/O порт.

Описание:

C: `ULONG outdword (ULONG offset, PULONG data, ULONG len=1, ULONG key=0);`

Pascal: `function outdword (offset:ULONG; var data:ULONG; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PULONG data - массив данных;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L1250 L1221 L305 L264 L1450

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

inmbyte

Ввод байта из памяти.

Описание:

C: `ULONG inmbyte (ULONG offset, PCHAR data, ULONG len=1, ULONG key=0);`
Pascal: `function inmbyte (offset:ULONG; var data:UCHAR; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;
PCHAR data - массив, в который будут занесены прочитанные данные;
ULONG len - размер массива в байтах;
ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
PCHAR data - прочитанные данные;

Реализована:

L7XX L791

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

inmword

Ввод слова из памяти.

Описание:

C: `ULONG inmword (ULONG offset, PUSHORT data, ULONG len=1, ULONG key=0);`

Pascal: `function inmword (offset:ULONG; var data:USHORT; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PUSHORT data - массив, в который будут занесены прочитанные данные;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PUSHORT data - прочитанные данные;

Реализована:

L7XX L791

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

inmddword

Ввод двойного слова из памяти.

Описание:

C: `ULONG inmddword (ULONG offset, PULONG data, ULONG len=1, ULONG key=0);`

Pascal: `function inmddword (offset:ULONG; var data:ULONG; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PULONG data - массив, в который будут занесены прочитанные данные;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

PULONG data - прочитанные данные;

Реализована:

L7XX L791

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

outmbyte

Вывод байта в память.

Описание:

C: `ULONG outmbyte (ULONG offset, PCHAR data, ULONG len=1, ULONG key=0);`

Pascal: `function outmbyte (offset:ULONG; var data:UCHAR; len:ULONG;
key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PCHAR data - массив данных;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L791

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

outmword

Вывод слова в память.

Описание:

C: `ULONG outmword (ULONG offset, PUSHORT data, ULONG len=1, ULONG key=0);`

Pascal: `function outmword (offset:ULONG; var data:USHORT; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PUSHORT data - массив данных;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

-

Реализована:

L7XX L791

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

outmdword

Вывод двойного слова в память.

Описание:

C: `ULONG outmdword (ULONG offset, PULONG data, ULONG len=1, ULONG key=0);`

Pascal: `function outmdword (offset:ULONG; var data:ULONG; len:ULONG; key:ULONG):ULONG;`

Параметры:

ULONG offset - смещение порта относительно базового адреса;

PULONG data - массив данных;

ULONG len - размер массива в байтах;

ULONG key - номер региона (фактически выбирает базовый адрес из списка возможных);

Возвращает:

L_SUCCESS - в случае успеха;

L_ERROR - в случае ошибки;

Реализована:

L7XX L791

Примечание:

В данной реализации нет проверки выхода смещения за разрешенную границу. Про key. У некоторых плат есть несколько регионов портов ввода/вывода. Для переключения между ними и служит этот параметр. Например, при key=0 для L7XX будет осуществляться доступ к служебным регистрам PLX, а при key=1 к DSP(те работа через порты, а не память). В стандартных драйверах этот параметр не задействован и должен быть всегда равен 0.

Типы данных

ADC_PAR

Это обобщенная структура для удобства работы со структурами задачи параметров сбора данных разных плат.

Описание:

C:

```
typedef union _ADC_PARAM_U_ :
{
    ADC_PAR_0 t1;
    ADC_PAR_1 t2;
} ADC_PAR, *PADC_PAR;
```

Параметры:

Примечание:

ADC_PAR_0

Структура служит для передачи параметров сбора данных в плату. Заполняется пользователем и передается в драйвер где и хранится.

Описание:

C:

```
typedef struct _ADC_PARAM_U_0 : public DAQ_PAR
{
// ULONG s_Type;
// ULONG FIFO;
// ULONG IrqStep;
// ULONG Pages;

    ULONG AutoInit;
    double dRate;
    double dKadr;
    double dScale;
    * ULONG Rate;
    * ULONG Kadr;
    * ULONG Scale;
    * ULONG FPDelay
    ULONG SynchroType;
    ULONG SynchroSensitivity;
    ULONG SynchroMode;
    ULONG AdChannel;
    ULONG AdPorog;
    ULONG NCh;
    ULONG Chn[128];
    ULONG IrqEna;
    ULONG AdcEna;
} ADC_PAR_0, *PADC_PAR_0;
```

Параметры:

ULONG s_Type - тип структуры (должен быть L_ADC_PARAM);
ULONG AutoInit - флаг указывающий на тип сбора данных 0 - однократный 1 - циклический;
double dRate - частота опроса каналов в кадре (кГц);
double dKadr - интервал между кадрами (мс);
double dScale - масштаб работы таймера для 1250 или делителя для 1221;
ULONG Rate - частота опроса каналов в кадре (в кодах для процессора, вычисляется библиотекой);
ULONG Kadr - интервал между кадрами (в кодах для процессора, вычисляется библиотекой);
ULONG Scale - масштаб работы таймера для 1250 или делителя для 1221 (в кодах для процессора, вычисляется библиотекой);
ULONG FPDelay - служебная величина задержки выдачи первого отсчета (вычисляется библиотекой);
ULONG SynchroType - тип синхронизации;
ULONG SynchroSensitivity - вид синхронизации;
ULONG SynchroMode - режим синхронизации;
ULONG AdChannel - канал, по которому выполняется синхронизация;
ULONG AdPorog - уровень синхронизации;
ULONG NCh - количество опрашиваемых каналов ;

ULONG Chn[128] - массив с номерами каналов и усилением на них; описывает порядок опроса каналов;

ULONG FIFO - размер половины аппаратного буфера FIFO на плате;

ULONG IrqStep - шаг генерации прерываний;

ULONG Pages - размер кольцевого буфера в шагах прерываний;

ULONG IrqEna - разрешение генерации прерывания от платы (1/0);

ULONG AdcEna - разрешение работы АЦП (1/0);

Примечание:

Структура ADC_PAR используется совместно с вызовом FillDAQparameters для настройки параметров ввода данных с платы АЦП. Поля отмеченные комментарием (//) наследуются из структуры DAQ_PAR. Пользователь должен заполнять все поля кроме тех, которые помечены (*). Особенности трактовки полей этой структуры для различных плат:

• **L-761/780/783**

- **s_Type** - должен быть L_ADC_PARAM;
- **Autolnit** - (**0** - однократное заполнение большого буфера и если установлено событие в функции SetLDeviceEvent, то произойдет генерация события)/(**1** - циклическое заполнение буфера);
- **dRate** - частота опроса каналов в кадре в килogerцах;
- **dKadr** - интервал между кадрами в миллисекундах, фактически определяет скорость сбора данных;
- **SynchroType**
 - **0** - цифровая синхронизация старта, остальные параметры синхронизации не используются;
 - **1** – по-кадровая синхронизация, остальные параметры синхронизации не используются;
 - **2** – аналоговая синхронизация старта по выбранному каналу АЦП;
 - **3** – нет синхронизации;
- **SynchroSensitivity**
 - **0** – аналоговая синхронизация по уровню;
 - **1** – аналоговая синхронизация по переходу;
- **SynchroMode**
 - **0** – по уровню «выше» или переходу «снизу-вверх»;
 - **1** - по уровню «ниже» или переходу «сверху-вниз»;
- **AdChannel** - канала, выбранный для аналоговой синхронизации;
- **AdPorog** - пороговое значение для аналоговой синхронизации в коде АЦП;
- **Nch** – количество опрашиваемых в кадре каналов;
- **Chn** – массив с логическими номерами каналов, слово вида 00000000 XXXXXXXX;

| № бита | Обозначение | Назначение |
|--------|-------------|--------------|
| 0 | MA0 | 0 бит номера |
| 1 | MA1 | 1 бит номера |
| 2 | MA2 | 2 бит номера |
| 3 | MA3 | 3 бит номера |
| 4 | MA4 | «0»/ 4 бит |

| | | |
|---|-----|------------------|
| | | |
| 5 | MA5 | 16 диф/32 общ |
| 6 | GS0 | 0 бит усил. |
| 7 | GS1 | 1 бит усил. |

MA5=MA4=0, MA3-MA0 – номер диф. канала.

MA5=0 MA4=1 – калибровка нуля.

MA5=1 MA4-MA0 – номер входа с общей землей.

GS0 GS1= {00, 01, 10, 11} - усиление {1, 4, 16, 64} для 780/761 и {1, 2, 4, 8} для 783.

• **L-1450**

- **s_Type** - должен быть L_ADC_PARAM;
- **Autolnit** - (**0** - однократное заполнение большого буфера и если установлено событие в функции SetLDeviceEvent, то произойдет генерация события)/(**1** циклическое заполнение буфера);
- **dRate** - частота опроса каналов в кадре в килogerцах;
- **dKadr** - интервал между кадрами в миллисекундах, фактически определяет скорость сбора данных;
- **SynchroType**
 - **0** - цифровая синхронизация старта, остальные параметры синхронизации не используются;
 - **1** – по-кадровая синхронизация, остальные параметры синхронизации не используются;
 - **2** – аналоговая синхронизация старта по выбранному каналу АЦП;
 - **3** – нет синхронизации;
- **SynchroSensitivity**
 - **0** – аналоговая синхронизация по уровню;
 - **1** – аналоговая синхронизация по переходу;
- **SynchroMode**
 - **0** – по уровню «выше» или переходу «снизу-вверх»;
 - **1** - по уровню «ниже» или переходу «сверху-вниз»;
- **AdChannel** - канала, выбранный для аналоговой синхронизации;
- **AdPorog** - пороговое значение для аналоговой синхронизации в коде АЦП;
- **Nch** – количество опрашиваемых в кадре каналов;
- **Chn** – массив с логическими номерами каналов, слово вида 00000000
XXXXXXXXX;

| № бита | Обозначение | Назначение |
|--------|-------------|--------------|
| 0 | MA0 | 0 бит номера |
| 1 | MA1 | 1 бит номера |
| 2 | MA2 | 2 бит номера |
| 3 | MA3 | 3 бит номера |

| | | |
|---|-----|------------------|
| 4 | MA4 | «0»/ 4 бит |
| 5 | MA5 | 16 диф/32 общ |
| 6 | GS0 | 0 бит усил. |
| 7 | GS1 | 1 бит усил. |

MA5=MA4=0, MA3-MA0 – номер диф. канала.

MA5=0 MA4=1 – калибровка нуля.

MA5=1 MA4-MA0 – номер входа с общей землей.

GS0 GS1= {00, 01, 10, 11} - усиление {1, 4, 16, 64}.

• **E14-440/E14-140/E154**

- **s_Type** - должен быть L_ADC_PARAM;
- **Autolnit** - (0 - однократное заполнение большого буфера и если установлено событие в функции SetLDeviceEvent, то произойдет генерация события)/(1 циклическое заполнение буфера);
- **dRate** - частота опроса каналов в кадре в килогерцах;
- **dKadr** - интервал между кадрами в миллисекундах, фактически определяет скорость сбора данных;
- **SynchroType**
 - 0 – нет синхронизации;
 - 1 - цифровая синхронизация старта, остальные параметры синхронизации не используются;
 - 2 – по-кадровая синхронизация, остальные параметры синхронизации не используются;
 - 3 – аналоговая синхронизация старта по выбранному каналу АЦП; для E140 сюда еще можно добавить биты 6 и 7, 6 бит включает внешний clock, 7 бит разрешает трансляцию clock на внешний разъем
- **SynchroSensitivity**
 - 0 – аналоговая синхронизация по уровню;
 - 1 – аналоговая синхронизация по переходу;
- **SynchroMode**
 - 0 – по уровню «выше» или переходу «снизу-вверх»;
 - 1 - по уровню «ниже» или переходу «сверху-вниз»;
- **AdChannel** - канала, выбранный для аналоговой синхронизации;
- **AdPorog** - пороговое значение для аналоговой синхронизации в коде АЦП;
- **Nch** – количество опрашиваемых в кадре каналов;(для E154 макс. 16)
- **Chn** – массив с логическими номерами каналов, слово вида 00000000 XXXXXXXX;

| № бита | Обозначение | Назначение |
|--------|-------------|--------------|
| 0 | MA0 | 0 бит номера |
| 1 | MA1 | 1 бит номера |
| 2 | MA2 | 2 бит номера |
| 3 | MA3 | 3 бит номера |

| | | |
|---|-----|------------------|
| | | |
| 4 | MA4 | «0»/ 4 бит |
| 5 | MA5 | 16 диф/32 общ |
| 6 | GS0 | 0 бит усил. |
| 7 | GS1 | 1 бит усил. |

MA5=MA4=0, MA3-MA0 – номер диф. канала.

MA5=0 MA4=1 – калибровка нуля.

MA5=1 MA4-MA0 – номер входа с общей землей.

GS0 GS1= {00, 01, 10, 11} - усиление {1, 4, 16, 64} .

- **L-1250/51/L-1450C/L-305/L-264:**
 - **NCh** - количество каналов (максимум 32);
 - **IrqStep** - должен быть равен FIFO т.к. плата может работать только по половинкам фифо буфера;
 - **SynchroType** - тип синхронизации (SMode по описанию DOS);
 - **SynchroMode** - режим синхронизации (TtlMask по описанию DOS);
- **L-1221:**
 - **dRate** - частота опроса на каждом канале.
 - **NCh** - битовая маска активных каналов (10101010 for example);
 - **Chn[32]** - первые 8 членов задают усиление на соответствующих каналах;

ADC_PAR_1

Структура служит для передачи параметров сбора данных в плату. Заполняется пользователем и передается в драйвер где и хранится.

Описание:

C:

```
typedef struct _ADC_PARAM_U_1 : public DAQ_PAR
{
// ULONG s_Type;
// ULONG FIFO;
// ULONG IrqStep;
// ULONG Pages;

    ULONG AutoInit;
    double dRate;
    double dKadr;
    ULONG Reserved1;
    ULONG DM_Ena;
    * ULONG Rate;
    * ULONG Kadr;
    ULONG StartCnt;
    ULONG StopCnt;
    ULONG SynchroType;
    ULONG SynchroMode;
    ULONG AdPorog;
    ULONG SynchroSrc;
    ULONG AdclMask;
    ULONG NCh;
    ULONG Chn[128];
    ULONG IrqEna;
    ULONG AdcEna;
} ADC_PAR_1, *PADC_PAR_1;
```

Параметры:

ULONG s_Type - тип структуры (должен быть L_ADC_PARAM);
ULONG AutoInit - флаг указывающий на тип сбора данных 0 - однократный 1 - циклический;
double dRate - частота опроса каналов в кадре (кГц);
double dKadr - интервал между кадрами (мс);
ULONG Reserved1 – зарезервировано;
ULONG DM_Ena - разрешение/запрещение маркировки данных **ULONG Rate** - частота опроса каналов в кадре (в кодах для цифрового автомата);
ULONG Kadr - интервал между кадрами (в кодах для цифрового автомата);
ULONG StartCnt – задержка старта в кадрах;
ULONG StopCnt — сколько кадров собирать после старта;
ULONG SynchroType - тип синхронизации;
ULONG SynchroMode – режим синхронизации и номер канала;
ULONG AdPorog — порог синхронизации;
ULONG SynchroSrc – источник внешней синхронизации;
ULONG AdclMask – задает режим ввода по каналам у модуля E2010;
ULONG NCh - количество опрашиваемых каналов ;
ULONG Chn[128] - массив с номерами каналов и усилением на них; описывает порядок опроса каналов;

ULONG FIFO - размер половины аппаратного буфера FIFO на плате;

ULONG IrqStep;

ULONG Pages – произведение этих двух параметров IrqStep*Pages задает количество отсчетов которое соберет плата при однократном сборе, но не больше чем 128K отсчетов. При циклическом сборе они игнорируются – буфер всегда 128K.

ULONG IrqEna - разрешение генерации прерывания от платы (mask/0), при этом mask - это младшие 16 бит в слове разрешающем прерывания от платы (блок АЦП);

ULONG AdcEna - разрешение работы АЦП (1/0);

Примечание:

Структура ADC_PAR_1 используется совместно с вызовом FillDAQparameters для настройки параметров ввода данных с платы АЦП. Поля отмеченные комментарием (//) наследуются из структуры DAQ_PAR. Пользователь должен заполнять все поля кроме тех, которые помечены (*). Особенности трактовки полей этой структуры для различных плат:

• **L-791**

- **s_Type** - тип структуры (должен быть L_ADC_PARAM);
- **Autolnit** - флаг указывающий на тип сбора данных 0 - однократный 1 – циклический;
- **dRate** - частота опроса каналов в кадре (кГц);
- **dKadr** - интервал между кадрами (мс);
- **Rate** - частота опроса каналов в кадре (в кодах для цифрового автомата);
- **Kadr** - интервал между кадрами (в кодах для цифрового автомата);
- **SynchroType** - тип синхронизации;
- **SynchroSrc** – источник внешней синхронизации;
- **NCh** - количество опрашиваемых каналов ;
- **Chn[128]** - массив с номерами каналов и усилением на них; описывает порядок опроса каналов;
- **FIFO** - размер половины аппаратного буфера FIFO на плате, возможные значения 1,2,4,8,16,32,64,128 отсчетов. При этом при первых трех возможных значениях передача BusMaster идет одиночными значениями, а при большем пакетная передача. Если установить больше 128, то она сама сбросит до 128.
- **IrqStep**;
- **Pages** – произведение этих двух параметров IrqStep*Pages задает количество отсчетов которое соберет плата при однократном сборе, но не больше чем 128K отсчетов. При циклическом сборе они игнорируются – буфер всегда 128K. При IrqEna=1 сгенерит еще прерывание.
- **IrqEna** - разрешение генерации прерывания от платы (mask/0), при этом mask - это младшие 16 бит в слове разрешающем прерывания от платы (блок АЦП) см. описание платы;
- **AdcEna** - разрешение работы АЦП (1/0);

• **E20-10**

- **s_Type** - тип структуры (должен быть L_ADC_PARAM);
- **Autolnit** - флаг указывающий на тип сбора данных 0 - однократный 1 – циклический;
- **dRate** - частота опроса каналов в кадре (кГц);
- **dKadr** - интервал между кадрами (мс);
- **Rate** - частота опроса каналов в кадре (в кодах для цифрового автомата);
- **Kadr** - интервал между кадрами (в кодах для цифрового автомата);
- **SynchroType** - тип синхронизации;
задается константами, определенными в файле e2010cmd.h
 - **INT_START_TRANS 0x01** – внутренний старт с разрешением трансляции сигнала на разъем;

- **INT_START 0x81** – просто внутренний старт;
- **EXT_START_UP 0x84** – внешний импульс старта по переднему фронту;
- **EXT_START_DOWN 0x94** – внешний импульс старта по заднему фронту;
- **EXT_START_DOWN_REVB 0x8C** – внешний импульс старта по заднему фронту для ревизии B;
- **SynchroSrc** – источник тактовых импульсов для АЦП;
 - **INT_CLK_TRANS 0x00** – внутренний источник с трансляцией;
 - **INT_CLK 0x40** – просто внутренний источник;
 - **EXT_CLK_UP 0x42** – внешний источник, по переднему фронту;
 - **EXT_CLK_DOWN 0x62** – внешний источник, по заднему фронту;
- **AdcIMask** – задает режим ввода по каналам у модуля E2010, сигнал или земля + входной диапазон; задается константами, определенными в файле e2010cmd.h через (+) для 0 канала
 - **V30_0 0x0000** – диапазон 3 В;
 - **V10_0 0x0008** – 1 В;
 - **V03_0 0x0010** – 0.3 В;
 - **GND_0 0x0000** – вход заземлен;
 - **SIG_0 0x0400** – вход подключен к сигналу; для остальных каналов аналогично с префиксами **_1 _2 _3**;
- **NCh** - количество опрашиваемых каналов ;
- **Chn[128]** - массив с номерами каналов, описывает порядок опроса каналов;
- **FIFO** -фактически не используется.
- **IrqStep** – размер запроса циклической посылки данных к USB, не более 1M отсчетов;
- **Расширенные параметры синхронизации для E20-10B**
 - **DM_Ena** — вкл/выкл маркирования начала блоков вводимых данных (удобно, например, при аналоговой синхронизации ввода по уровню);
 - **StartCnt** - задержка старта сбора данных в кадрах отсчетов АЦП;
 - **StopCnt** - останов сбора данных после задаваемого здесь кол-ва собранных кадров отсчетов АЦП;
 - **SynchroMode** — режим аналоговой синхронизации и номер канал;
 - **A_SYNC_OFF 0x0000** — нет аналоговой синхронизации;
 - **A_SYNC_UP_EDGE 0x0080** — синхронизация по переднему фронту;
 - **A_SYNC_DOWN_EDGE 0x0084** — синхронизация по заднему фронту;
 - **A_SYNC_HL_LEVEL 0x0088** — синхронизация по положительному уровню;
 - **A_SYNC_LH_LEVEL 0x008C** — синхронизация по отрицательно уровню;
 - + по | (или) соединить с номером канала по которому синхронизация CH_0 или CH_1 или CH_2 или CH-3.
 - **AdPorog** - порог срабатывания при аналоговой синхронизации;

DAC_PAR

Это обобщенная структура для удобства работы со структурами задачи параметров сбора данных разных плат.

Описание:

C:

```
typedef union _DAC_PARAM_U_ :  
{  
  
    DAC_PAR_0 t1;  
    DAC_PAR_1 t2;  
  
} DAC_PAR, *PDAC_PAR;
```

Параметры:

Примечание:

DAC_PAR_0

Структура служит для передачи параметров вывода данных в плату. Заполняется пользователем и передается в драйвер где хранится в похожей структуре, но в удобном для платы представлении.

Описание:

```
C:
typedef struct _DAC_PARAM_U_0: public DAQ_PAR
{
// ULONG s_Type;
// ULONG FIFO;
// ULONG IrqStep;
// ULONG Pages;

    ULONG AutoInit;
    double dRate;
    * ULONG Rate;
    ULONG IrqEna;
    ULONG DacEna;
    ULONG DacNumber;
} DAC_PAR_0, *PDAC_PAR_0;
```

Параметры:

ULONG s_Type -; тип структуры (должен быть L_DAC_PARAM);
ULONG AutoInit - флаг указывающий на тип сбора данных 0 - однократный 1 - циклический; (пока не используется)
double dRate - частота вывода данных на ЦАП (кГц);
ULONG Rate - частота вывода данных на ЦАП (в кодах для процессора);
ULONG FIFO - размер половины аппаратного буфера FIFO на плате;
ULONG IrqStep - шаг генерации прерываний;
ULONG Pages - размер кольцевого буфера в шагах прерываний;
ULONG IrqEna - разрешение генерации прерывания от платы (1/0);
ULONG DacEna - разрешение работы ЦАП (1/0);
ULONG DacNumber - номер канала ЦАП на который выводить данные;

Примечание:

Структура DAC_PAR_0 используется совместно с вызовом FillDAQparameters для настройки параметров вывода данных с ЦАП платы. Поля отмеченные комментарием (//) наследуются из структуры DAQ_PAR. Пользователь должен заполнять все поля кроме тех, которые помечены (*). Особенности трактовки полей этой структуры для различных плат:

- L-761/780/783/L-1450 - полностью так, как описано кроме:
 - **DacNumber** - не задействован (номер ЦАП задается в самих данных);
 - **IrqEna,Pages**- прерывания и реальный кольцевой буфер работают только в L-1450 и L780С, для остальных плат Pages всегда надо задавать 2;
 - **IrqStep** - должен быть равен FIFO;
- L-1250/51/L-305/L-264:
 - **IrqEna,Pages** - прерывания и реальный кольцевой буфер работают только в L-1450 и L780С, для остальных плат Pages всегда надо задавать 2;
 - **IrqStep** - должен быть равен FIFO т.к. плата может работать только по половинкам фифо буфера;
 - **dRate** - надо ставить такой же как и для АЦП т.к. ЦАП на этих платах как бы синхронен с АЦП - на ввод одного кадра с АЦП происходит один вывод на

ЦАП;

DAC_PAR_1

Структура служит для передачи параметров вывода данных в плату. Заполняется пользователем и передается в драйвер где хранится в похожей структуре, но в удобном для платы представлении.

Описание:

```
C:
typedef struct _DAC_PARAM_U_1: public DAQ_PAR
{
// ULONG s_Type;
// ULONG FIFO;
// ULONG IrqStep;
// ULONG Pages;

    ULONG AutoInit;
    double dRate;
    * ULONG Rate;
    ULONG IrqEna;
    ULONG DacEna;
    ULONG Reserved1;
} DAC_PAR_1, *PDAC_PAR_1;
```

Параметры:

ULONG s_Type -; тип структуры (должен быть L_DAC_PARAM);
ULONG AutoInit - флаг указывающий на тип сбора данных 0 - однократный 1 - циклический; (пока не используется)
double dRate - частота вывода данных на ЦАП (кГц);
ULONG Rate - частота вывода данных на ЦАП (в кодах для процессора);
ULONG FIFO - размер половины аппаратного буфера FIFO на плате;
ULONG IrqStep - шаг генерации прерываний;
ULONG Pages - размер кольцевого буфера в шагах прерываний;
ULONG IrqEna - разрешение генерации прерывания от платы (1/0);
ULONG DacEna - разрешение работы ЦАП (1/0);

Примечание:

Структура DAC_PAR_1 используется совместно с вызовом FillDAQparameters для настройки параметров вывода данных с ЦАП платы. Поля отмеченные комментарием (//) наследуются из структуры DAQ_PAR. Пользователь должен заполнять все поля кроме тех, которые помечены (*). Особенности трактовки полей этой структуры для различных плат:

- L-791 - полностью так, как описано кроме:
 - **AutoInit** – не используется;
 - **IrqStep** – не используется;
 - **Pages**- не используется, буфер всегда 128К отсчетов;
 - **FIFO,IrqStep** – не используется, буфер всегда 128К отсчетов; прерывания генерируются по флагам в данных ЦАП

ASYNC_PAR

Структура служит для передачи в плату параметров асинхронного ввода/вывода данных. Используется совместно IoAsync.

Описание:

C:

```
typedef struct _ASYNC_PARAM_ : public DAQ_PAR
{
    // USHORT s_Type;
    // USHORT FIFO;
    // USHORT IrqStep;
    // USHORT Pages;

    double dRate;
    * USHORT Rate;
    USHORT NCh;
    USHORT Chn[128];
    USHORT Data[128];
    ULONG Mode;

} ADC_PAR, *PADC_PAR;
```

Параметры:

USHORT s_Type - указывает для какой операции ввода/вывода содержатся данные в структуре (L_ASYNC_ADC_CFG, L_ASYNC_TTL_CFG, L_ASYNC_DAC_CFG, L_ASYNC_ADC_INP, L_ASYNC_TTL_INP, L_ASYNC_TTL_OUT, L_ASYNC_DAC_OUT);

USHORT Data[128] - массив для данных;

double dRate - частота опроса каналов в кадре (кГц);

USHORT Rate - частота опроса каналов в кадре (в кодах для процессора);

USHORT NCh - количество опрашиваемых каналов ;

USHORT Chn[128] - массив с номерами каналов и усилением на них; описывает порядок опроса каналов;

USHORT FIFO - размер половины аппаратного буфера FIFO на плате;

USHORT IrqStep - шаг генерации прерываний;

USHORT Pages - размер кольцевого буфера в шагах прерываний;

USHORT Mode - задает различные режимы при конфигурации.

Примечание:

Структура ASYNC_PAR используется совместно с вызовом IoAsync. Поля отмеченные комментарием (//) наследуются из структуры DAQ_PAR. Пользователь должен заполнять все поля кроме тех, которые помечены (*). Как заполнять или что читать из этой структуры см. описание функции IoAsync.

PLATA_DESCR

Структура описывает FLASH на PCI платах L-761/L-780/L-783.

Описание:

```
C:
typedef struct __PLATA_DESCR
{
    char SerNum[9];
    char BrdName[5];
    char Rev;
    char DspType[5];
    long Quartz;
    USHORT IsDacPresent;
    USHORT Reserv1[7];
    USHORT KoefADC[8];
    USHORT KoefDAC[4];
    USHORT Custom[32];
} PLATA_DESCR , *PPLATA_DESCR;
```

Параметры:

char SerNum[9] - серийный номер платы;
char BrdName[5] - название платы;
char Rev - ревизия платы;
char DspType[5] - тип DSP;
long Quartz - частота кварца;
USHORT IsDacPresent - наличие ЦАПa;
USHORT Reserv1[7] - зарезервировано;
USHORT KoefADC[8] - калибровочные коэффициенты АЦП;
USHORT KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom[32] - пользовательское место;

Примечание:

PLATA_DESCR_1450

Структура описывает FLASH на ISA плате L-1450.

Описание:

```
C:
typedef struct __PLATA_DESCR_1450
{
    char SerNum[9];
    char BrdName[7];
    char Rev;
    char DspType[5];
    char IsDacPresent;
    char IsExtMemPresent;
    long Quartz;
    USHORT Reserv1[6];
    USHORT KoefADC[8];
    USHORT KoefDAC[4];
    USHORT Custom[32];
} PLATA_DESCR_1450 , *PPLATA_DESCR_1450;
```

Параметры:

char SerNum[9] - серийный номер платы;
char BrdName[7] - название платы;
char Rev - ревизия платы;
char DspType[5] - тип DSP;
char IsDacPresent - наличие ЦАП;
char IsExtMemPresent - наличие внешней памяти данных;
long Quartz - частота кварца;
USHORT Reserv1[6] - зарезервировано;
USHORT KoefADC[8] - калибровочные коэффициенты АЦП;
USHORT KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom[32] - пользовательское место;

Примечание:

PLATA_DESCR_L791

Структура описывает FLASH на PCI плате L-791.

Описание:

```
C:
typedef struct __PLATA_DESCR_L791
{
    USHORT crc;
    char SerNum[16];
    char BrdName[16];
    char Rev;
    char DspType[5];
    long Quartz;
    USHORT IsDacPresent;
    float KoefADC[16];
    float KoefDAC[4];
    USHORT Custom;
} PLATA_DESCR_L791 , *PPLATA_DESCR_L791;
```

Параметры:

char SerNum[16] - серийный номер платы;
char BrdName[16] - название платы;
char Rev - ревизия платы;
char DspType[5] - тип DSP;
USHORT IsDacPresent - наличие ЦАПа;
long Quartz - частота кварца;
float KoefADC[16] - калибровочные коэффициенты АЦП;
float KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom - пользовательское место;

Примечание:

Данная структура используется в интерфейсных функциях, которые работают со служебной областью пользовательского ППЗУ: **ReadPlataDescr** и **WritePlataDescr**.

| Название поля | Назначение и допустимые значения поля |
|---------------|--|
| crc | Контрольная сумма, рассчитанная по всем полям структуры. |
| SerNum | Серийный номер модуля (строка символов максимальной с длиной 16) |
| BrdName | Название модуля (строка символов максимальной с длиной 16) |
| Rev | Ревизия модуля (ascii символ) |
| DspType | Тип используемого в модуле процессора (строка символов с максимальной длиной 16) |
| Quartz | Частота задающего кварца (32-х разрядное целое) |

| Название поля | Назначение и допустимые значения поля |
|----------------|---|
| IsDacPresented | Флаг наличия ЦАП в модуле (логическая величина) |
| KoefAdc[0] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x1'. (число с плавающей точкой одинарной точности) |
| KoefAdc[1] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x2'.(число с плавающей точкой двойной точности) |
| KoefAdc[2] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x4'.(число с плавающей точкой двойной точности) |
| KoefAdc[3] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x8'.(число с плавающей точкой двойной точности) |
| KoefAdc[4] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x16'.(число с плавающей точкой двойной точности) |
| KoefAdc[5] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x32'.(число с плавающей точкой двойной точности) |
| KoefAdc[6] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x64'.(число с плавающей точкой двойной точности) |
| KoefAdc[7] | Коэффициент коррекции смещения нуля АЦП. Усилении 'x128'.(число с плавающей точкой двойной точности) |
| KoefAdc[8] | Коэффициент коррекции масштаба АЦП. Усилении 'x1'. (число с плавающей точкой двойной точности) |
| KoefAdc[9] | Коэффициент коррекции масштаба АЦП. Усилении 'x2'. (число с плавающей точкой двойной точности) |
| KoefAdc[10] | Коэффициент коррекции масштаба АЦП. Усилении 'x4'. (число с плавающей точкой двойной точности) |
| KoefAdc[11] | Коэффициент коррекции масштаба АЦП. Усилении 'x5'. (число с плавающей точкой двойной точности) |
| KoefAdc[12] | Коэффициент коррекции масштаба АЦП. Усилении 'x16'.(число с плавающей точкой двойной точности) |
| KoefAdc[13] | Коэффициент коррекции масштаба АЦП. Усилении 'x32'. (число с плавающей точкой двойной точности) |
| KoefAdc[14] | Коэффициент коррекции масштаба АЦП. Усилении 'x64'.(число с плавающей точкой двойной точности) |
| KoefAdc[15] | Коэффициент коррекции масштаба АЦП. Усилении 'x128'. (число с плавающей точкой двойной точности) |
| KoefDac[0] | Коэффициент коррекции смещения нуля ЦАП. Канал '0'. (число с |

| Название поля | Назначение и допустимые значения поля |
|---------------|---|
| | плавающей точкой двойной точности) |
| KoefDac[1] | Коэффициент коррекции смещения нуля ЦАП. Канал '1'. (число с плавающей точкой двойной точности) |
| KoefDac[2] | Коэффициент коррекции масштаба ЦАП. Канал '0'. (число с плавающей точкой двойной точности) |
| KoefDac[3] | Коэффициент коррекции масштаба ЦАП. Канал '1'. (число с плавающей точкой двойной точности) |

Корректировка данных АЦП/ЦАП.

Схемотехника и использованные компоненты обеспечивают линейность передаточной характеристики АЦП/ЦАП модуля. Однако, в виду отсутствия автоматической коррекции как внутри модуля так и в штатной dll-библиотеки, показания АЦП/ЦАП могут иметь некоторое смещение нуля и неточность в передаче масштаба. Работа по коррекции показаний возлагается на пользовательское приложение.

Для корректировки показаний АЦП/ЦАП можно воспользоваться собственными калибровочными коэффициентами и формулами или штатными коэффициентами.

Штатные коэффициенты вычисляются при наладке модуля на производстве и хранятся в системном ППЗУ модуля. Для того чтобы ими воспользоваться, необходимо:

- считать системное ППЗУ модуля при помощи функции **ReadPlataDescr()**
- из считанной системной информации выбрать коэффициенты масштаба и смещения нуля соответствующие диапазону измерения АЦП или номеру канала ЦАП (см. описание структуры **PLATA_DESCR_L791**)
- воспользоваться приведенной ниже формулой:

Корректировка данных АЦП:

$Y = (X+B)*A$, где:

X – некорректированные данные АЦП [в отсчетах АЦП]

Y – скорректированные данные АЦП [в отсчетах АЦП]

A – коэффициент масштаба [безразмерный]

B – коэффициент смещение нуля [в отсчетах АЦП]

Примечание: Коэффициенты **A** и **B** одни и те же для всех каналов АЦП, но различные для разных диапазонов измерения.

Корректировка данных ЦАП:

$Y = (X+B)*A$, где:

X – некорректированные данные ЦАП [в отсчетах ЦАП]

Y – корректированные данные ЦАП [в отсчетах ЦАП]

A – коэффициент масштаба [безразмерный]

B – коэффициент смещение нуля [в отсчетах ЦАП]

Пример 1:

С АЦП, настроенного на диапазон $\pm 2.5\text{В}$ (усиление $\times 4$), получены следующие данные:

$X_1=1000$, $X_2=-1000$, $X_3=0$

тогда, если положить что **pd** – структура типа **PLATA_DESCR_L791** предварительно участвовавшая в вызове функции **ReadPlataDescr()**, то коэффициенты коррекции и скорректированные данные можно получить так:

A=pd. KoefAdc[10], **B=pd. KoefADC [2]**

$Y_1=(B+1000)*A$, **$Y_2=(B-1000)*A$** , **$Y_3=B*A$**

Пример 2:

На втором канале ЦАП необходимо выставить напряжение, соответствующее следующим кодам:

$X_1=1000$, $X_2=-1000$, $X_3=0$

тогда, если положить что **pd** – структура типа **PLATA_DESCR_L791** предварительно участвовавшая в вызове функции **ReadPlataDescr()**, то коэффициенты коррекции и данные, которые необходимо записать во второй канал ЦАП, можно получить так:

A=pd. KoefDac[3], **B=pd. KoefDac[1]**

$Y_1=(B+1000)*A$, **$Y_2=(B-1000)*A$** , **$Y_3=B*A$**

PLATA_DESCR_E440

Структура описывает FLASH на USB модуле E14-440.

Описание:

```
C:
typedef struct __PLATA_DESCR_440
{
    char SerNum[9];
    char BrdName[7];
    char Rev;
    char DspType[5];
    char IsDacPresent;
    long Quartz;
    char Reserv2[13];
    USHORT KoefADC[8];
    USHORT KoefDAC[4];
    USHORT Custom[32];
} PLATA_DESCR_E440 , *PPLATA_DESCR_E440;
```

Параметры:

char SerNum[9] - серийный номер платы;
char BrdName[7] - название платы;
char Rev - ревизия платы;
char DspType[5] - тип DSP;
long Quartz - частота кварца;
char IsDacPresent - наличие ЦАПа;
char Reserv2[13] - зарезервировано;
USHORT KoefADC[8] - калибровочные коэффициенты АЦП;
USHORT KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom[32] - пользовательское место;

Примечание:

PLATA_DESCR_E140

Структура описывает FLASH на USB модуле E14-140.

Описание:

```
C:
typedef struct __PLATA_DESCR_E140
{
    char SerNum[9];
    char BrdName[11];
    char Rev;
    char DspType[11];
    char IsDacPresent;
    long Quartz;
    char Reserv2[3];
    float KoefADC[8];
    float KoefDAC[4];
    USHORT Custom[20];
} PLATA_DESCR_E140 , *PPLATA_DESCR_E140;
```

Параметры:

char SerNum[9] - серийный номер платы;
char BrdName[11] - название платы;
char Rev - ревизия платы;
char DspType[11] - тип DSP;
long Quartz - частота кварца;
char IsDacPresent - наличие ЦАП;
char Reserv2[3] - зарезервировано;
float KoefADC[8] - калибровочные коэффициенты АЦП;
float KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom[20] - пользовательское место;

Примечание:

Внутри модуля это все храниться в ужасном формате PACKED_PLATA_DESCR_E140 котрый определен в файле ioctl.h. Но от пользователя это все скрыто и знать об этом не обязательно...

PLATA_DESCR_E154

Структура описывает FLASH на USB модуле E154.

Описание:

```
C:
typedef struct __PLATA_DESCR_E154
{
    char SerNum[9];
    char BrdName[11];
    char Rev;
    char DspType[11];
    char IsDacPresent;
    long Quartz;
    char Reserv2[3];
    float KoefADC[8];
    float KoefDAC[4];
    USHORT Custom[20];
} PLATA_DESCR_E154 , *PPLATA_DESCR_E154;
```

Параметры:

char SerNum[9] - серийный номер платы;
char BrdName[11] - название платы;
char Rev - ревизия платы;
char DspType[11] - тип DSP;
long Quartz - частота кварца;
char IsDacPresent - наличие ЦАП;
char Reserv2[3] - зарезервировано;
float KoefADC[8] - калибровочные коэффициенты АЦП;
float KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom[20] - пользовательское место;

Примечание:

Внутри модуля это все храниться в ужасном формате PACKED_PLATA_DESCR_E154 котрый определен в файле ioctl.h. Но от пользователя это все скрыто и знать об этом не обязательно...

PLATA_DESCR_E2010

Структура описывает FLASH на USB модуле E20-10.

Описание:

```
C:
typedef struct __PLATA_DESCR_E2010
{
    char BrdName[16];
    char SerNum[16];
    char DspType[16];
    long Quartz;
    char Rev;
    char IsDacPresent;
    float KoefADC[24];
    float KoefDAC[4];
    USHORT Custom[45];
} PLATA_DESCR_E2010 , *PPLATA_DESCR_E2010;
```

Параметры:

char SerNum[16] - серийный номер платы;
char BrdName[16] - название платы;
char Rev - ревизия платы;
char DspType[16] - тип DSP;
long Quartz - частота кварца;
char IsDacPresent - наличие ЦАП;
float KoefADC[24] - калибровочные коэффициенты АЦП;
float KoefDAC[4] - калибровочные коэффициенты ЦАП;
USHORT Custom[45] - пользовательское место;

Примечание:

PLATA_DESCR_U

Это обобщенная структура для удобства работы с флешами разных плат.

Описание:

C:

```
typedef union __PLATA_DESCR_U
{
    PLATA_DESCR t1;
    PLATA_DESCR_1450 t2;
    PLATA_DESCR_L791 t3;
    WORD_IMAGE wi;
    BYTE_IMAGE bi;
} PLATA_DESCR_U, *PPLATA_DESCR_U;

typedef union __PLATA_DESCR_U2
{
    PLATA_DESCR t1;
    PLATA_DESCR_1450 t2;
    PLATA_DESCR_L791 t3;
    PLATA_DESCR_E440 t4;
    PLATA_DESCR_E140 t5;
    PACKED_PLATA_DESCR_E140 pt5;
    PLATA_DESCR_E2010 t6;

    WORD_IMAGE wi;
    BYTE_IMAGE bi;
    WORD_IMAGE_256 wi256;
    BYTE_IMAGE_256 bi256;
} PLATA_DESCR_U2, *PPLATA_DESCR_U2;
```

Примечание:

Структура PLATA_DESCR_U2 дополнена новыми типами данных для новых модулей и расширена до 256 байт против 128. Библиотека внутри хранит все уже в структуре U2.

WORD_IMAGE

Представление структуры флеша в виде массива слов.

Описание:

```
C:  
typedef struct __WORD_IMAGE  
{  
    WORD data[64];  
} WORD_IMAGE, *PWORD_IMAGE;
```

Примечание:

SLOT_PAR

Структура описывает параметры виртуального слота.

Описание:

```
C:
typedef struct __SLOT_PARAM
{
    ULONG Base;
    ULONG BaseL;
    ULONG Base1;
    ULONG BaseL1;
    ULONG Mem;
    ULONG MemL;
    ULONG Mem1;
    ULONG MemL1;
    ULONG Irq;
    ULONG BoardType;
    ULONG DSPTYPE;
    ULONG Dma;
    ULONG DmaDac;
    ULONG DTA_REG;
    ULONG IDMA_REG;
    ULONG CMD_REG;
    ULONG IRQ_RST;
    ULONG DTA_ARRAY;
    ULONG RDY_REG;
    ULONG CFG_REG;
} SLOT_PAR, *PSLOT_PAR;
```

Параметры:

ULONG Base - базовый адрес первого региона портов;
ULONG BaseL - протяженность первого региона портов в байтах;
ULONG Base1 - базовый адрес второго региона портов;
ULONG BaseL1 - протяженность второго региона портов в байтах;
ULONG Mem - адрес первого региона памяти;
ULONG MemL - протяженность первого региона памяти в байтах;
ULONG Mem1 - адрес второго региона памяти;
ULONG MemL1 - протяженность второго региона памяти в байтах;
ULONG Irq - используемое драйвером аппаратное прерывание;
ULONG BoardType - тип платы;
ULONG DSPTYPE - тип установленного на плате DSP;
ULONG Dma - используемый для ввода данных канал ПДП: 0 - не использовать,5,6;

ULONG DmaDac - используемый для вывода данных канал ПДП: 0 - не использовать,6;
ULONG DTA_REG;
ULONG IDMA_REG;
ULONG CMD_REG;
ULONG IRQ_RST;
ULONG DTA_ARRAY;
ULONG RDY_REG;
ULONG CFG_REG; - адреса регистров платы относительно базового адреса;

Примечание:

Структура SLOT_PAR используется совместно с вызовом GetSlotParam для получения параметров виртуальных слотов.

Для платы L791 значение имеют только поля Mem, MemL, Irq, BoardType. Все остальные ничего не значат.

Коды ошибок

Пока определены следующие коды ошибок:

- **L_SUCCESS 0** - функция выполнена успешно;
- **L_NOTSUPPORTED 1** - функция не поддерживается этой платой;
- **L_ERROR 2** - ошибка при выполнении функции;
- **L_ERROR_NOBOARD 3** - нет платы в запрашиваемом слоте;
- **L_ERROR_INUSE 4** – плата в запрашиваемом слоте уже используется;

Типы плат

В заголовочных файлах определены следующие типы плат поддерживаемые библиотекой:

- **NONE 0** - нет платы;
- **L1250 1** - плата L1250;
- **N1250 2** - плата N1250 (работоспособность не проверялась);
- **L1251 3** - плата L1251;
- **L1221 4** - плата L1221;
- **PCIA 5** - плата серии L7XX rev. A;
- **PCIB 6** - плата серии L7XX rev. B;
- **L264 8** - плата L264;
- **L305 9** - плата L305;
- **L1450C 10** - плата L1450 с совместимым биосом L1250;
- **L1450 11** - плата L1450;
- **L032 12** - плата L032;
- **HI8 13** - плата HI8;
- **PCIC 14** - плата серии L7XX rev. C (L780M);
- **L791 19** - плата серии L79X (L791);
- **E440 30** - модуль E14-440;
- **E140 31** - модуль E14-140;
- **E2010 32** – модуль E20-10;
- **E154 38** – модуль E154;
- **E2010B 39** — модуль E20-10 ревизии B

Другие константы

В заголовочных файлах определены следующие константы:

- **L_ADC_PARAM 1** - трактовать DAQ_PAR как ADC_PAR при передаче в FillDAQparameters;
- **L_DAC_PARAM 2** - трактовать DAQ_PAR как DAC_PAR при передаче в FillDAQparameters;
- **L_ASYNC_ADC_CFG 3** - ASYNC_PAR содержит запрос на конфигурирование АЦП;
- **L_ASYNC_TTL_CFG 4** - ASYNC_PAR содержит запрос на конфигурирование цифровых линий;
- **L_ASYNC_DAC_CFG 5** - ASYNC_PAR содержит запрос на конфигурирование ЦАП;
- **L_ASYNC_ADC_INP 6** - ASYNC_PAR содержит запрос на ввод данных с АЦП;
- **L_ASYNC_TTL_INP 7** - ASYNC_PAR содержит запрос на ввод данных с цифровых линий;
- **L_ASYNC_TTL_OUT 8** - ASYNC_PAR содержит запрос на вывод данных на цифровые линии;
- **L_ASYNC_DAC_OUT 9** - ASYNC_PAR содержит запрос на вывод данных на ЦАП;
- **L_STREAM_ADC 1** - тип потока - поток данных с АЦП;
- **L_STREAM_DAC 2** - тип потока - поток данных с ЦАП;
- **L_EVENT_ADC_BUF 1**
- **L_EVENT_DAC_BUF 2**
- **L_EVENT_ADC_OVF 3**
- **L_EVENT_ADC_FIFO 4**
- **L_EVENT_DAC_USER 5**
- **L_EVENT_DAC_UNF 6**
- **L_EVENT_PWR_OVR 7** – макросы для различных событий устанавливаемых в функции SetLDeviceEvent для платы L791.

Справочные данные по платам

Адресное пр-во и команды биос L-1250/L-264/L-305

Адресное пространство L-1250/L-264/L-305

| Адрес | Чтение | Запись |
|----------|---|---|
| Base+0x0 | Данные от платы. До тех пор, пока процессор ADSP ничего не записал в порт данных, соответствующий бит готовности равен нулю. После записи слова в порт данных процессором ADSP автоматически устанавливается в единичное значение бит готовности, который переключится в ноль после того, как компьютер считает переданное слово. | Данные для передачи в плату. После записи слова в порт данных автоматически устанавливается в нулевое значение бит готовности, который переключится в единицу автоматически после того, как процессор ADSP считает переданное слово. |
| Base+0x8 | Сброс прерывания от платы в PC. В том случае если используется прерывание IRQ10/11, то обработчик прерывания обязан произвести одно чтение из этого порта для сброса запроса прерывания. В противном случае линия прерываний окажется заблокированной. | При записи команды в этот порт автоматически генерируется прерывание в процессоре ADSP IRQ2. Драйвер LBIOS использует этот порт для передачи команд, при этом обработчик прерывания в процессоре в ответ на IRQ2 считывает переданный код через порт данных и вызывает функцию, соответствующую переданному коду. |
| Base+0xC | Порт битов готовности и двух TTL цифровых линий (34 и 35 на внешнем разъеме) | Порт конфигурации. При помощи порта конфигурации можно: - загрузить в процессор управляющую программу; - отключить линию прерывания IRQ10/11; - перевести порт данных в режим работы по каналам ПДП; |

Список команд поддерживаемых биос L-1250/L-264/L-305

| Номер | Обозначение | Описание |
|-------|-----------------|--|
| 0 | cmSTOP_1251 | перевод платы в "тихое" состояние |
| 1 | cmADC_CHAN_1251 | ввод с АЦП с переустановкой каналов |
| 2 | cmOUT_DAC_1251 | вывод на ЦАП |
| 3 | cmSTREAM_1251 | одноканальный ввод с синхронизацией от внутреннего таймера |

| | | |
|----|----------------------|---|
| 4 | cmSOFT_1251 | многоканальный ввод с синхронизацией от внутреннего таймера |
| 5 | cmSAMPLE_1251 | ввод с АЦП без переустановки каналов |
| 6 | cmTTL_IN_1251 | ввод цифровых портов |
| 7 | cmTTL_OUT_1251 | вывод в цифровые порты |
| 8 | cmIRQ_SIMPLE_1251 | генерирование прерываний от внутреннего таймера |
| 9 | cmIRQ_ADC_CHAN_1251 | генерирование прерываний от внутреннего таймера с одноканальным вводом с АЦП |
| 10 | cmIRQ_KADR_1251 | генерирование прерываний от внутреннего таймера с многоканальным вводом с АЦП |
| 11 | cmCALIBR_1251 | зарезервирована |
| 12 | cmTEST_1251 | тестирование наличия платы |
| 13 | cmDAC_STREAM_1251 | вывод на ЦАП массива |
| 14 | cmMEMORY_STATE_1251 | тестирование памяти |
| 15 | cmFILTER_1251 | зарезервирована |
| 16 | cmFLT_NO_LOAD_1251 | зарезервирована |
| 17 | cmSET_MULTI_1251 | зарезервирована |
| 18 | cmSET_SCALE_1251 | установка коэффициента масштабирования таймера |
| 19 | cmSET_UVX_1251 | зарезервирована |
| 20 | cmSET_DSP_SPEED_1251 | установка типа скорости процессора |
| 21 | cmSET_WAITSTATE_1251 | установка задержки на память |
| 22 | cmSET_DELAY_1251 | установка межканальной задержки |
| 23 | cmSYNCHRO_MODE_1251 | установка типа синхронизации |

| | | |
|----|-----------------------|-----------------------------------|
| 24 | cmDAC_CONFIG_1251 | установка номера ЦАПа |
| 25 | cmFIFO_CONFIG_1251 | режим FIFO буфера |
| 26 | cmSOFT_MEMORY_1251 | ввод во внутреннюю память |
| 28 | cmPROGRAM_1251 | программирование модулей в крейте |
| 29 | cmWRITE_TO_MEM_1251 | ввод в память массива |
| 30 | cmREAD_FROM_MEM_1251 | чтение массива из памяти |
| 31 | cmGET_LSM_NAME_1251 | чтение кода модуля |
| 32 | cmSET_2FIFO_TYPE_1251 | установка двойного FIFO |
| 33 | cmSET_LM_DAC_1251 | установка ЦАПа LM модуля |
| 34 | cmGET_LM_TTL_1251 | чтение цифрового модуля |
| 35 | cmCOMPARATPR_1251 | частотомер |
| 36 | cmTEST_PM_1251 | проверка наличия памяти программ |
| 37 | cmLOW_POWER_1251 | cmLOW_POWER_1251 |
| 39 | cmLM404_1251 | работа с модулем LM-404 |

Адресное пр-во и команды биос L-7XX

Существует две ревизии PCI плат - А и В. Они различаются адресацией портов. Ревизия А очень редкая. Наиболее распространенная ревизия – В. Base - один из трех возможных вариантов базового адреса:порты ввода/вывода,память ниже 1Мб,память выше 1Мб.Значения Base можно увидеть под Windows в Панели Управления/Система в ресурсах соответствующих PCI плат. Под Windows всегда используется доступ к плате через память выше 1Мб.

Адресное пространство L-761/L-780/L-783 (Rev A).

| Адрес | Чтение | Запись |
|----------------|---|---|
| Base+0 | Порт для чтения данных с платы по IDMA как при одиночных операция, так и при блочных. | Порт для записи данных в плату по IDMA как при одиночных операция, так и при блочных. |
| Base+40 96 | - | Порт для установки адреса IDMA. |
| Base+81 92 | - | Порт генерации IRQ2 DSP. |
| Base+12 288 | - | Порт сброса прерываний. |

Адресное пространство L-761/L-780/L-783 (Rev B).

| Адрес | Чтение | Запись |
|---------------|--|--|
| Base+0 | Порт для чтения данных с платы по IDMA при одиночных операция. | Порт для записи данных в плату по IDMA при одиночных операция. |
| Base+2 | - | Порт для установки адреса IDMA. |
| Base+4 | - | Порт генерации IRQ2 DSP. |
| Base+6 | - | Порт сброса прерываний. |
| Base+40 96 | Порт для чтения данных с платы по IDMA при блочных операциях. | Порт для записи данных в плату по IDMA при блочных операциях. |

Список команд поддерживаемых биос L-761/L-780/L-783

| Номер | Обозначение | Описание | Использует |
|-------|-------------|---|-----------------|
| 0 | cmTEST_PLX | Проверка загрузки платы и ее работоспособности; | L_TEST_LOAD_PLX |

| | | | |
|----|--------------------------|--|---|
| 1 | cmLOAD_CONTROL_TABLE_PLX | Загрузка управляющей таблицы в память DSP; | L_CONTROL_TABLE_PLX, L_CONTROL_TABLE_LENGTH_PLX |
| 2 | cmADC_ENABLE_PLX | Разрешение/Запрещение работы АЦП; | L_ADC_ENABLE_PLX |
| 3 | cmADC_FIFO_CONFIG_PLX | Конфигурирование параметров кольцевого буфера АЦП; | L_ADC_FIFO_BASE_ADDRESS_PLX, L_ADC_FIFO_BASE_ADDRESS_INDEX_PLX, L_ADC_FIFO_LENGTH_PLX, L_ADC_NEW_FIFO_LENGTH_PLX |
| 4 | cmSET_ADC_RATE_PLX | Установка временных параметров работы АЦП; | L_ADC_RATE_PLX, L_INTER_CADR_DELAY_PLX |
| 5 | cmENABLE_DAC_STREAM_PLX | Разрешение/запрещение выдачи данных из буфера ЦАП. | L_DAC_ENABLE_STREAM_PLX |
| 6 | cmDAC_FIFO_CONFIG_PLX | Конфигурирование параметров буфера ЦАП; | L_DAC_FIFO_BASE_ADDRESS_PLX, L_DAC_FIFO_LENGTH_PLX, L_DAC_NEW_FIFO_LENGTH_PLX |
| 7 | cmSET_DAC_RATE_PLX | Установка частоты вывода данных на ЦАП; | L_DAC_RATE_PLX |
| 8 | cmADC_SAMPLE_PLX | Однократный ввод с АЦП; | L_ADC_SAMPLE_PLX, L_ADC_CHANNEL_PLX |
| 9 | cmTTL_IN_PLX | Чтение данных с цифровых линий; | L_TTL_IN_PLX |
| 10 | cmTTL_OUT_PLX | Вывод данных на цифровые линии; | L_TTL_OUT_PLX |
| 11 | cmSYNCHRO_CONFIG_PLX | Управление синхронизацией; | L_SYNCHRO_TYPE_PLX, L_SYNCHRO_AD_CHANNEL_PLX, L_SYNCHRO_AD_POROG_PLX, L_SYNCHRO_AD_MODE_PLX, L_SYNCHRO_AD_SENSITIVITY_PLX |
| 12 | cmENABLE_IRQ_PLX | Разрешение/запрещение работы с прерываниями; | L_ENABLE_IRQ_PLX, L_ENABLE_IRQ_VALUE_PLX, |

| | | | |
|----|--------------------|---|------------------|
| | | | L_IRQ_STEP_PLX |
| 13 | cmIRQ_TEST_PLX | Тестовая команда генерирует прерывания 10 раз в сек; | L_ENABLE_IRQ_PLX |
| 14 | cmSET_DSP_TYPE_PLX | Передает в драйвер тип установленного на плате DSP и соответствующим образом модифицирует код драйвера; | L_DSP_TYPE_PLX |

Список внутренних переменных биос L-761/L-780/L-783 (8 - признак того, что это DM)

| Адрес | Обозначение | Описание |
|--------|----------------------------|---|
| 0x8A00 | L_CONTROL_TABLE_PLX | Управляющая таблица содержащая логические номера каналов (до 96). В соответствии с ней DSP производит последовательный циклический сбор данных с АЦП. Размер этой таблицы задается переменной L_CONTROL_TABLE_LENGTH_PLX. По умолчанию { 0, 1, 2, 3, 4, 5, 6, 7 } |
| 0x8D00 | L_SCALE_PLX | Массив с 4 калибровочными коэффициентами используемый при корректировке масштаба данных с АЦП. По умолчанию { 7FFF, 0x7FFF, 0x7FFF, 0x7FFF } |
| 0x8D04 | L_ZERO_PLX | Массив с 4 калибровочными коэффициентами используемый при корректировке смещения нуля данных с АЦП. По умолчанию { 0x0, 0x0, 0x0, 0x0 } |
| 0x8D08 | L_CONTROL_TABLE_LENGTH_PLX | Размер управляющей таблицы. По умолчанию 8. |
| 0x8D40 | L_READY_PLX | Флажок готовности платы к дальнейшей работе. После загрузки управляющей программы в DSP необходимо дождаться установления данного флажка в 1. |
| 0x8D41 | L_TMODE1_PLX | Тестовая переменная. После загрузки управляющей программы по этому адресу должно читаться число 0x5555. |
| 0x8D42 | L_TMODE2_PLX | Тестовая переменная. После загрузки управляющей программы по этому адресу должно читаться число 0xAAAA. |
| 0x8D48 | L_DSP_TYPE_PLX | Переменная, передающая драйверу тип установленного на модуле DSP. 0 - ADSP2184; 1 - ADSP2185; 2 - ADSP2186; По умолчанию 0. |
| 0x8D49 | L_COMMAND_PLX | Переменная, при помощи которой драйверу передается номер команды. |

| | | |
|--------|------------------------|--|
| 0x8D4C | L_TTL_OUT_PLX | Переменная, в которой хранятся значения 16-ти выходных цифровых линий. |
| 0x8D4D | L_TTL_IN_PLX | Переменная, в которой хранятся значения 16-ти входных цифровых линий. |
| 0x8D50 | L_FIFO_PTR_PLX | Переменная, в которой хранится текущий адрес заполнения кольцевого буфера. Данная переменная по мере ввода данных меняет свое значение от L_ADC_FIFO_BASE_ADDRESS_PLX до L_ADC_FIFO_ADDRESS_PLX + L_ADC_FIFO_LENGTH_PLX. |
| 0x8D52 | L_TEST_LOAD_PLX | Тестовая переменная. |
| 0x8D53 | L_ADC_RATE_PLX | Переменная, задающая частоту работы АЦП. |
| 0x8D54 | L_INTER_KADR_DELAY_PLX | Переменная, задающая межкадровую задержку при вводе данных с АЦП. |
| 0x8D55 | L_DAC_RATE_PLX | Переменная, задающая частоту вывода данных с ЦАП-ов. |
| 0x8D56 | L_DAC_VALUE_PLX | Величина, которую требуется установить на выходе ЦАП-а. |
| 0x8D57 | L_ENABLE_IRQ_PLX | Запрещение(0)/разрешение(1) генерации прерывания в РС при соответствующем заполнении кольцевого буфера АЦП. По умолчанию - 0. |
| 0x8D58 | L_IRQ_STEP_PLX | Переменная, задающая число отсчетов при заполнении кольцевого буфера АЦП, каждый раз при превышении которого генерируется прерывание в РС |
| 0x8D5A | L_IRQ_FIFO_ADDRESS_PLX | Если произошло прерывание в РС, то начиная с этого адреса можно считать L_IRQ_STEP_PLX отсчетов из кольцевого буфера АЦП. |
| 0x8D5B | L_ENABLE_IRQ_VALUE_PLX | Переменная, значение которой при выполнении соответствующей команды передается в переменную L_ENABLE_IRQ_PLX. |
| 0x8D5C | L_ADC_SAMPLE_PLX | Данная переменная используется при однократном вводе с АЦП, храня считанное значение. |
| 0x8D5D | L_ADC_CHANNEL_PLX | Данная переменная используется при однократном вводе с АЦП, задавая логический номер канала. |
| 0x8D5E | L_DAC_SCLK_DIV_PLX | - |

| | | |
|--------|-----------------------------------|---|
| 0x8D60 | L_CORRECTION_ENABLE_PLX | Разрешение(1)/запрещение(0) корректировки данных аналоговых каналов при помощи калибровочных коэффициентов. По умолчанию - 0. |
| 0x8D62 | L_ADC_ENABLE_PLX | Запрещение(0)/разрешение(1) работы АЦП. |
| 0x8D63 | L_ADC_FIFO_BASE_ADDRESS_PLX | Текущий базовый адрес кольцевого буфера АЦП. По умолчанию - 0x2000. |
| 0x8D64 | L_ADC_FIFO_BASE_ADDRESS_INDEX_PLX | Переменная, задающая базовый адрес кольцевого буфера АЦП. Может принимать три значения: 0 - (0x0000 для ADSP-2185), 1 - (0x2000 для ADSP-2185 -2186), 2 - (0x3000 для ADSP-2185 -2186; 0x2000 для ADSP-2184). |
| 0x8D65 | L_ADC_FIFO_LENGTH_PLX | Текущая длина кольцевого буфера АЦП. По умолчанию 0x800. |
| 0x8D66 | L_ADC_NEW_FIFO_LENGTH_PLX | Переменная, задающая длину кольцевого буфера АЦП. |
| 0x8D67 | L_DAC_ENABLE_STREAM_PLX | Запрещение(0)/разрешение(1) вывода данных из буфера ЦАП на ЦАП. |
| 0x8D68 | L_DAC_FIFO_BASE_ADDRESS_PLX | Текущий базовый адрес буфера ЦАП. Данный буфер расположен в памяти программ DSP. По умолчанию 0xC00. |
| 0x8D69 | L_DAC_FIFO_LENGTH_PLX | Текущая длина буфера ЦАП. По умолчанию 0x400. |
| 0x8D6A | L_DAC_NEW_FIFO_LENGTH_PLX | Переменная, задающая длину буфера ЦАП. |
| 0x8D70 | L_SYNCHRO_TYPE_PLX | Переменная, задающая тип синхронизации. |
| 0x8D73 | L_SYNCHRO_AD_CHANNEL_PLX | При аналоговой синхронизации задает логический номер канала, по которому происходит синхронизация. |
| 0x8D74 | L_SYNCHRO_AD_THRESHOLD_PLX | Порог аналоговой синхронизации. |
| 0x8D75 | L_SYNCHRO_AD_MODE_PLX | Переменная, задающая режим синхронизации по переходу "снизу - вверх"(0) или "сверху - вниз"(1) |
| 0x8D76 | L_SYNCHRO_AD_SENSITIVITY_PLX | Переменная, задающая тип синхронизации по уровню(0) или по переходу(1). |

Адресное пр-во и команды биос L-1221

Адресное пространство L-1221.

| Адрес | Чтение | Запись |
|--------|---|---|
| Base+0 | 16-битные данные передаваемые по каналу IDMA. | 16-битные данные передаваемые по каналу IDMA. |
| Base+2 | Вызывает генерацию прерывания IRQ1 в сигнальном процессоре. Служит для генерации команды. | Устанавливает начальный адрес IDMA . Данные передаваемые/считываемые в/из DSP будут размещаться/считываться начиная с этого адреса. |

Список команд поддерживаемых биос L-1221

| Номер | Обозначение | Описание | Использует |
|-------|------------------------|---|--|
| 0 | cmTEST_1221 | Проверка загрузки платы и ее работоспособности; | L_TEST_LOAD_1221 |
| 1 | cmFLASH_WRIT E_1221 | Запись байта в ППЗУ. {зарезервирована} | - |
| 2 | cmFLASH_READ _1221 | Чтение байта из ППЗУ. | L_FLASH_ADDRESS_1221, L_FLASH_BYTE_1221 |
| 3 | cmSET_TTL_122 1 | Установка цифровых линий. | L_TTL_CONFIG_1221, L_TTL_OUT_1221 |
| 4 | cmGET_TTL_122 1 | Чтение цифровых линий. | L_TTL_CONFIG_1221, L_TTL_IN_1221 |
| 5 | cmENABLE_IRQ _1221 | Разрешение/запрещение прерываний. | L_ENABLE_IRQ_1221 |
| 6 | cmRESET_AD_1 221 | Переинициализация аналогового тракта. | L_CHANNEL_MASK_1221, L_RATE_SCALE_1221, L_READY_1221 |
| 7 | cmSET_GAIN_12 21 | Установка входного диапазона. | L_GAIN_CHANNEL_1221, L_GAIN_1221 |
| 8 | cmSET_RATE_1 221 | Установка частоты ввода. | L_GAIN_1221 |
| 9 | cmCONFIG_FIFO _1221 | Установка параметров буфера. | L_FIFO_START_ADDRESS_1221 , L_FIFO_LENGTH_1221 |

Список внутренних переменных биос L-1221

| Адрес | Обозначение | Описание |
|--------|--------------------------|---|
| 0x2000 | L_BUFFER_1221 | По умолчанию базовый адрес буфера в плате L-1221, размером 4096 слов. Данный адрес используется при чтении данных, поступающих с аналоговых каналов. Если были сделаны переустановки параметров буфера с помощью команды smCONFIG_FIFO_1221, то надо пользоваться переменными L_FIFO_START_ADDRESS_1221 и L_FIFO_LENGTH. |
| 0x3C80 | L_SCALE_1221 | Массив с 64 калибровочными коэффициентами масштаба аналоговых каналов. |
| 0x3CC0 | L_ZERO_1221 | Массив с 64 калибровочными коэффициентами смещения нуля в режиме измерения постоянного напряжения. |
| 0x3D00 | L_ZERO_FLT_1221 | Массив с 64 калибровочными коэффициентами смещения нуля в режиме фильтрации постоянной составляющей. |
| 0x3D40 | L_KADR_1221 | Массив из 8 чисел с последними данными со всех восьми аналоговых каналов платы. Независимо от переменной L_CHANNEL_MASK_1221 в массив L_KADR_1221 постоянно подгружаются данные со всех аналоговых каналов. При этом по адресу L_KADR_1221 будет считываться код с первого канала АЦП, а по адресу L_KADR_1221+7 - с восьмого канала. |
| 0x3D48 | L_TMODE_1221 | Тестовая переменная, после загрузки управляющей программы по этому адресу должно читаться число 5. |
| 0x3D49 | L_COMMAND_1221 | Переменная, при помощи которой задается номер команды. |
| 0x3D4A | L_FLASH_ADDRES S_1221 | Адрес байта в ППЗУ. ППЗУ - энергонезависимая память на плате, в которой хранятся калибровочные коэффициенты. |
| 0x3D4B | L_FLASH_BYTE_12 21 | Байт данных записываемый/считываемый в/из ППЗУ. |
| 0x3D4C | L_TTL_OUT_1221 | Слово, в котором хранятся значения 3 выходных цифровых линий. Формат - 00000000 0XXX000Y.Y - управляет режимом отсечки. |
| 0x3D4D | L_TTL_IN_1221 | Слово, в котором хранятся значения 3 входных цифровых линий. Формат - 00000000 0XXX0000. |
| 0x3D4E | L_OVERFLOW_122 1 | Слово, в котором запоминаются биты, фиксирующие переполнение на аналоговых каналах. 1 - переполнение. Младший бит соответствует 0 каналу, старший - 7. Формат - |

| | | |
|--------|---------------------|--|
| | | 00000000 XXXXXXXX |
| 0x3D4F | L_ENABLE_IRQ_1221 | Слово, разрешающее(1)/запрещающее(0) генерирование прерываний от платы к компьютеру. |
| 0x3D50 | L_FIFO_PTR_1221 | Переменная, в которой хранится текущий адрес заполнения буфера. Данная переменная по мере ввода данных меняет свое значение от L_FIFO_START_ADDRESS_1221 до L_FIFO_START_ADDRESS_1221 + L_FIFO_LENGTH_1221-1. |
| 0x3D51 | L_CHANNEL_MASK_1221 | Битовая маска, при помощи которой задаются активные каналы, т.е. те каналы АЦП, данные с которых надо помещать в буфер. Формат - 00000000 XXXXXXXX. |
| 0x3D52 | L_TEST_LOAD_1221 | Тестовая переменная. 0xAA55. |
| 0x3D53 | L_RATE_1221 | Переменная устанавливающая частоту ввода данных. Целое число от 0 до 14 :0-6.99 kHz,1-7.46 kHz,2-7.99 kHz,3-8.6 kHz,4-9.32 kHz,5-10.17 kHz,6-11.19 kHz,7-12.43 kHz,8-13.98 kHz,9-15.98 kHz,10-18.64 kHz,11-22.37 kHz,12-27.97 kHz,13-37.29 kHz,14-55.93 kHz. |
| 0x3D54 | L_RATE_SCALE_1221 | Масштаб частоты ввода. Уменьшает частоту ввода : 0-1,1-16,2-256,3-4096. |
| 0x3D55 | L_GAIN_CHANNEL_1221 | Номер канала на котором изменяется входной диапазон. |
| 0x3D56 | L_GAIN_1221 | Устанавливаемый входной диапазон : 0-6.3 В,1-3.15 В,2-1.5 В,3-0.75 В,4-0.375 В,5-0.18 В,6-0.08 В,7-0.04 В. |
| 0x3D57 | L_IRQ_STEP_1221 | Шаг, с которым генерируется прерывание по мере заполнения буфера. |
| 0x3D58 | L_N_BIT_1221 | Разрядность АЦП. Фактически сдвиг данных вправо. 16 бит - (0) 15 бит - (-1) ... |
| 0x3D59 | L_FIFO_N_LOW_1221 | Младшее слово числа введенных частей буфера. |
| 0x3D5A | L_FIFO_N_HIGH_1221 | Старшее слово числа введенных частей буфера. |
| 0x3D5B | L_IRQ0_N_LOW_1221 | Младшее слово счетчика внешних импульсов. |
| 0x3D5C | L_IRQ0_N_HIGH_1221 | Старшее слово счетчика внешних импульсов. |

| | | |
|--------|-------------------------------|--|
| | 221 | |
| 0x3D5D | L_TTL_CONFIG_1221 | Конфигурация TTL линий. 1 - выход 0 – вход. Формат - 00000000 0XXX0000 |
| 0x3D63 | L_CORRECTION_ENABLE_1221 | Переменная запрещающая(0)/разрешающая(1) коррекцию вводимых данных. По умолчанию - 0; |
| 0x3D64 | L_FIFO_START_ADDRESS_1221 | Переменная, задающая начальный адрес буфера. Если она равна нулю, то адрес равен 0x0000. Если не равна нулю - адрес равен 0x2000. По умолчанию - 0x2000. |
| 0x3D65 | L_FIFO_LENGTH_1221 | Определяет длину буфера. По умолчанию 0x1000. |
| 0x3D69 | L_READY_1221 | 0 - do calibration 1 – ready. Переменная сигнализирующая о готовности платы. |
| 0x3D70 | L_IRQ_ADDRESS_1221 | Переменная указывающая на каком адресе в буфере произошло прерывание. |
| 0x3F6C | L_TEMPERATURE_1221 | Температура платы. |
| 0x3D5E | L_SYNCHRO_TYPE_1221 | Переменная, задающая тип синхронизации. |
| 0x3D5F | L_SYNCHRO_AD_CHANNEL_1221 | При аналоговой синхронизации задает номер канала, по которому происходит синхронизация. |
| 0x3D60 | L_SYNCHRO_AD_PROG_1221 | Порог аналоговой синхронизации. |
| 0x3D61 | L_SYNCHRO_AD_MODE_1221 | Переменная, задающая режим синхронизации по переходу "снизу - вверх"(0) или "сверху - вниз"(1) |
| 0x3D62 | L_SYNCHRO_AD_SENSITIVITY_1221 | Переменная, задающая тип синхронизации по уровню(0) или по переходу(1). |

Адресное пр-во и команды биос L-1450

Адресное пространство L-1450.

| Адрес | Чтение | Запись |
|--------------|--|---|
| Base+ 0x0 | Порт данных. Чтение 16-битного слова из порта данных. Все 16 считываемых бит являются значимыми. До тех пор, пока процессор DSP ничего не записал в порт данных, соответствующий бит готовности равен нулю. После записи 16-битного слова в порт данных сигнальным процессором автоматически устанавливается в единичное значение бит готовности RDYR, который переключится в ноль после того, как компьютер считает переданное слово. | Порт данных. Запись 16-битного слова в порт данных. Все 16 записываемых бит являются значимыми. После записи слова в порт данных автоматически устанавливается в нулевое значение бит готовности RDYW, который переключится в единицу автоматически после того, как процессор ADSP считает переданное слово. |
| Base+ 0x4 | Порт дополнительных данных. Имеют значение только два младших бита в принимаемых данных, которые используются в штатном LBIOS при работе с платой по прерываниям. | Порт EIORDY. Разрешения использования сигнала I/O CH RDY магистрали ISA. |
| Base+ 0x8 | Порт сброса запроса прерывания. В том случае если используется прерывание IRQ 10/11/12/15, то обработчик данного прерывания в PC обязан произвести одно чтение из описываемого порта для сброса запроса прерывания. В противном случае линия прерываний окажется заблокированной. | Порт команд. При записи номера команды в данный порт автоматически генерируется прерывание IRQ2 в сигнальном процессоре и, одновременно с этим, записывает передаваемое число в порт данных. При этом обработчик прерывания драйвера LBIOS в ответ на запрос IRQ2 считывает переданный номер команды и вызывает процедуру, соответствующую переданному номеру команд. |
| Base+ 0xC | Порт статуса. Порт битов готовности записи RDYW и чтения RDYR. | Порт управления. При помощи порта конфигурации можно: загрузить в процессор управляющую программу, отключить линию прерывания IRQ 10/11/12/15, перевести плату в режим работы по каналам Прямого Доступа к Памяти (ПДП). |

Список команд поддерживаемых биос L-1450

| Номер | Обозначение | Описание | Использует |
|-------|--------------|------------------------------|-------------------|
| 0 | cmTEST_L1450 | Проверка загрузки платы и ее | L_TEST_LOAD_L1450 |

| | | | |
|----|------------------------------------|--|---|
| | | работоспособности. | |
| 1 | cmGET_DM_WO RD_L1450 | Чтение слова из памяти данных DSP | - |
| 2 | cmPUT_DM_WO RD_L1450 | Запись слова в память данных DSP | - |
| 3 | cmGET_PM_WO RD_L1450 | Чтение слова из памяти программ DSP | - |
| 4 | cmPUT_PM_WO RD_L1450 | Запись слова в память программ DSP | - |
| 5 | cmGET_DM_AR RAY_L1450 | Чтение массива слов из памяти данных DSP | - |
| 6 | cmPUT_DM_AR RAY_L1450 | Запись массива слов в память данных DSP | - |
| 7 | cmGET_PM_AR RAY_L1450 | Чтение массива слов из памяти программ DSP | - |
| 8 | cmPUT_PM_ARR AY_L1450 | Запись массива слов в память программ DSP | - |
| 9 | cmENABLE_FLAS H_WRITE_L145 0 | Разрешение процедуры записи в пользовательское ППЗУ | L_FLASH_ENABLE_L1450 |
| 10 | cmREAD_FLASH _WORD_L1450 | Чтение слова из ППЗУ | L_FLASH_ADDRESS_L1450, L_FLASH_DATA_L1450 |
| 11 | cmWRITE_FLAS H_WORD_L1450 | Запись слова в ППЗУ | L_FLASH_ADDRESS_L1450, L_FLASH_DATA_L1450 |
| 12 | cmSTART_L1450 | Разрешение работы АЦП и/или ЦАП | L_ADC_ENABLE_L1450, L_ENA_ADC_IRQ_L1450, L_ADC_DATA_STEP_L1450, L_DAC_ENABLE_L1450, L_ENA_DAC_IRQ_L1450, L_DAC_DATA_STEP_L1450 |
| 13 | cmSET_ADC_PA RS_L1450 | Установка параметров работы АЦП | L_CONTROL_TABLE_L1450, L_CONTROL_TABLE LENGHT_ L1450, L_ADC_RATE_L1450, L_INTER_KADR_SCALE_L1450, L_INTER_KADR_PERIOD_L1450 |
| 14 | cmADC_FIFO_C | Конфигурирование параметров | L_ADC_FIFO_BASE_ADDRESS_ |

| | | | |
|----|-------------------------|---|---|
| | ONFIG_L1450 | FIFO буфера АЦП. | L1450, L_ADC_FIFO_BASE_ADDRESS_INDEX_L1450, L_ADC_FIFO_LENGTH_L1450, L_ADC_NEW_FIFO_LENGTH_L1450 |
| 15 | cmADC_SAMPLE_L1450 | Однократный ввод отсчета АЦП с заданного канала | L_ADC_SAMPLE_L1450, L_ADC_CHANNEL_L1450 |
| 16 | cmSYNCHRO_CONFIG_L1450 | Управление синхронизацией начала ввода данных с АЦП. | L_SYNCHRO_TYPE_L1450, L_SYNCHRO_AD_CHANNEL_L1450, L_SYNCHRO_AD_POROG_L1450, L_SYNCHRO_AD_MODE_L1450, L_SYNCHRO_AD_SENSITIVITY_L1450 |
| 17 | cmDAC_FIFO_CONFIG_L1450 | Конфигурирование параметров FIFO буфера ЦАП. | L_DAC_FIFO_BASE_ADDRESS_L1450, L_DAC_FIFO_LENGTH_L1450, L_DAC_NEW_FIFO_LENGTH_L1450 |
| 18 | cmSET_DAC_RATE_L1450 | Установка частоты вывода данных из FIFO буфера ЦАП | L_DAC_RATE_L1450 |
| 19 | cmENABLE_TTL_OUT_L1450 | Разрешение выходных цифровых линий | L_ENABLE_TTL_OUT_L1450 |
| 20 | cmTTL_IN_L1450 | Считывание состояния 16ти внешних цифровых линий. | L_TTL_IN_L1450 |
| 21 | cmTTL_OUT_L1450 | Управление 16тью внешними цифровыми линиями. | L_TTL_OUT_L1450 |
| 22 | cmIRQ_TEST_L1450 | Тестовая функция для генерирования прерываний в РС с частотой примерно 300 Гц. | L_ENABLE_IRQ_L1450 |
| 23 | cmSET_PAGE_MEMORY_L1450 | Установка номера страницы внешней памяти | L_PAGE_MEMORY_L1450 |
| 24 | cmSET_DSP_TYPE_L1450 | Передает в драйвер LBIOS тип установленного на плате DSP и соответствующим образом модифицирует код драйвера. | L_DSP_TYPE_L1450 |

Список внутренних переменных биос L-1450 (8 - признак того, что это DM)

| Адрес | Обозначение | Описание |
|--------|----------------------------|--|
| 0x8A00 | L_CONTROL_TABLE_L1450 | Управляющая таблица, содержащая последовательность логических номеров каналов (максимум 128 элементов). В соответствии с ней DSP производит последовательный циклический сбор данных с АЦП. Размер этой таблицы задается переменной L_CONTROL_TABLE_LENGTH_L1450 (см. ниже). По умолчанию - { 0, 1, 2, 3, 4, 5, 6, 7} |
| 0x8D00 | L_SCALE_L1450 | Массив с 4 калибровочными коэффициентами, используемый при корректировке масштаба данных с АЦП. По умолчанию - { 0x7FFF, 0x7FFF, 0x7FFF, 0x7FFF} |
| 0x8D04 | L_ZERO_L1450 | Массив с 4 калибровочными коэффициентами, используемый при корректировке смещения нуля данных с АЦП. По умолчанию - { 0x0, 0x0, 0x0, 0x0} |
| 0x8D08 | L_CONTROL_TABLE_LENGTH_PLX | Размер управляющей таблицы (максимум 128 логических каналов). По умолчанию - 8. |
| 0x8D0A | L_IS_EXT_MEMORY_XIST_L1450 | Флажок наличия внешней памяти данных |
| 0x8D0B | L_PAGE_MEMORY_L1450 | Текущий номер используемой страницы внешней памяти (только для плат Rev.'A') |
| 0x8D41 | L_TMODE1_L1450 | Тестовая переменная. После загрузки драйвера (LBIOS) по этому адресу должно читаться число 0x5555. |
| 0x8D42 | L_TMODE2_L1450 | Тестовая переменная. После загрузки драйвера (LBIOS) по этому адресу должно читаться число 0xAAAA. |
| 0x8D43 | L_ENA-DAC_IRQ_L1450 | Данная переменная разрешает (0x1) либо запрещает (0x0) генерирование прерываний в PC по мере необходимости новых данных для FIFO буфера ЦАП. По умолчанию - 0x0. |
| 0x8D48 | L_DSP_TYPE_L1450 | Переменная, передающая драйверу (LBIOS) тип установленного на модуле DSP. Если она равна 0, то на плате установлен ADSP-2184 (4 КСлов памяти программ и 4 КСлов памяти данных). Если она равна 1, то - ADSP-2185 (16 кСлов памяти программ и 16 КСлов памяти данных). Если она равна 2, то - ADSP-2186 (8 КСлов памяти программ и 8 КСлов памяти данных). По умолчанию L_DSP_TYPE_L1450=0. |
| 0x8D4B | L_DAC_DATA_STEP_L1450 | Переменная, задающая шаг (число отсчетов) при генерировании прерываний в PC по мере необходимости в получении новых данных для FIFO буфера ЦАП. При помощи этой переменной можно сделать так, что прерывания в PC будут генерироваться, например, после |

| | | |
|--------|---------------------------|---|
| | | каждых 20 отсчетов выведенных на ЦАП. По умолчанию - 0x200. |
| 0x8D4C | L_TTL_OUT_L1450 | Слово (16 бит) , в котором по-битово хранятся значения 16ти выходных цифровых линий для их выставления по команде C_TTL_OUT_L1450. |
| 0x8D4D | L_TTL_IN_L1450 | Слово (16 бит), в котором после выполнения команды C_TTL_IN_L1450 по-битово хранятся значения 16ти входных цифровых линий. |
| 0x8D4E | L_ENABLE_TTL_OUT_L1450 | Данная переменная разрешает (0x1) либо запрещает (0x0) использование выходных цифровых линий (перевод их в третье состояние) |
| 0x8D53 | L_ADC_RATE_L1450 | Переменная, задающая частоту работы АЦП. |
| 0x8D54 | L_INTER_KADR_SCALE_L1450 | Переменная, совместно с L_INTER_KADR_PERIOD_L1450 задающая межкадровую задержку при вводе данных с АЦП. |
| 0x8D55 | L_DAC_RATE_L1450 | Переменная, задающая частоту вывода данных с ЦАП-ов. |
| 0x8D57 | L_ENA_ADC_IRQ_L1450 | Данная переменная разрешает (0x1) либо запрещает (0x0) генерирование прерываний в РС по мере заполнения FIFO буфера АЦП. По умолчанию - 0x0. |
| 0x8D58 | L_ADC_DATA_STEP_L1450 | Переменная, задающая шаг (число отсчетов) при генерировании прерываний в РС по мере заполнения FIFO буфера АЦП. При помощи этой переменной можно сделать так, что прерывания в РС будут генерироваться, например, через каждые 20 отсчетов. По умолчанию - 0x400. |
| 0x8D5A | L_INTER_KADR_PERIOD_L1450 | Переменная, совместно с L_INTER_KADR_SCALE_L1450 задающая межкадровую задержку при вводе данных с АЦП. |
| 0x8D5C | L_ADC_SAMPLE_L1450 | Данная переменная используется при однократном вводе с АЦП, храня считанное значение. |
| 0x8D5D | L_ADC_CHANNEL_L1450 | Данная переменная используется при однократном вводе с АЦП, задавая логический номер канала. |
| 0x8D60 | L_CORRECTION_ENABLE_L1450 | Переменная запрещающая (0)/ разрешающая (1) корректировку данных аналоговых каналов при помощи калибровочных коэффициентов. По умолчанию L_CORRECTION_ENABLE=0x0. |
| 0x8D62 | L_ADC_ENABLE_L1450 | Переменная запрещающая(0)/разрешающая (1) работу АЦП. |

| | | |
|--------|-------------------------------------|--|
| | 1450 | |
| 0x8D63 | L_ADC_FIFO_BASE_ADDRESS_L1450 | Текущий базовый адрес FIFO буфера АЦП. По умолчанию L_ADC_FIFO_BASE_ADDRESS_L1450 = 0x2000. |
| 0x8D64 | L_ADC_FIFO_BASE_ADDRESS_INDEX_L1450 | Переменная, задающая требуемый базовый адрес FIFO буфера АЦП. Может принимать три значения: 0 - базовый адрес начинается с адреса 0x0 только для ADSP-2185, 1 - базовый адрес начинается с адреса 0x2000 только для ADSP-2185 и ADSP-2186, 2 - базовый адрес начинается с адреса 0x3000 для ADSP-2185 и ADSP-2186 и с адреса 0x2000 для ADSP-2184. |
| 0x8D65 | L_ADC_FIFO_LENGTH_L1450 | Текущая длина FIFO буфера АЦП. По умолчанию L_ADC_FIFO_LENGTH_L1450 = 0x800. |
| 0x8D66 | L_ADC_NEW_FIFO_LENGTH_L1450 | Переменная, задающая требуемую длину FIFO буфера АЦП. |
| 0x8D67 | L_DAC_ENABLE_L1450 | Переменная запрещающая (0)/разрешающая (1) вывод данных из FIFO буфера ЦАП на сам ЦАП. |
| 0x8D68 | L_DAC_FIFO_BASE_ADDRESS_L1450 | Текущий базовый адрес FIFO буфера ЦАП. Данный буфер расположен в памяти программ DSP. По умолчанию L_DAC_FIFO_BASE_ADDRESS_L1450 = 0xC00. |
| 0x8D69 | L_DAC_FIFO_LENGTH_L1450 | Текущая длина FIFO буфера ЦАП. По умолчанию L_DAC_FIFO_LENGTH_L1450 = 0x400. |
| 0x8D6A | L_DAC_NEW_FIFO_LENGTH_L1450 | Переменная, задающая требуемую длину FIFO буфера ЦАП. |
| 0x8D70 | L_SYNCHRO_TYPE_L1450 | Переменная, задающая тип синхронизации. |
| 0x8D73 | L_SYNCHRO_AD_CHANNEL_L1450 | При аналоговой синхронизации задает логический номер канала, по которому происходит синхронизация. |
| 0x8D74 | L_SYNCHRO_AD_POROG_L1450 | Порог аналоговой синхронизации. |
| 0x8D75 | L_SYNCHRO_AD_MODE_L1450 | Переменная, задающая режим синхронизации по переходу "снизу - вверх"(0) или "сверху - вниз"(1) |
| 0x8D76 | L_SYNCHRO_AD_SENSITIVITY_L1450 | Переменная, задающая тип синхронизации по уровню(0) или по переходу(1). |

Замечания для платы L791

Пояснения по работе с платой L791.

Это плата без сигнального процессора на борту - просто цифровой автомат. Передачу данных осуществляет по BusMaster каналу PCI.

Временные параметры сбора задаются таймерами.

Библиотека для работы с платой имеет интерфейс аналогичный интерфейсу других плат. Но есть некоторые особенности и ограничения:

- циклические буфера в компьютере всегда имеют одно и тоже значение 128K отсчетов (32 разрядных) для АЦП и 128K отсчетов для ЦАП;
- переменная -указатель sync служит не только для чтения счетчиков ПДП каналов, но и для прямого доступа к 32-х битным регистрам платы;
- при этом указатель sync от потока ЦАП и от потока АЦП это одно и тоже;
- прерывания от платы доходят к пользователю через события. При этом они разрешаются битами в параметре IrqEna в структуре описывающей сбор данных АЦП (младшие 16 бит) и ЦАП (старшие 16 бит).
- с платой можно работать и напрямую по регистрам управляя сбором данных, но только после команды StartLDevice надо дождаться появления бита ADC_En в регистре управления, после чего запретить сбор. Дальше можно делать что хочется. В плату уже прописаны адреса памяти для BusMaster каналов и установлен обработчик прерывания. Все что хочется можно делать до команды StopLDevice.
- про регистры платы читайте печатную книжку;
- про логические номера каналов читайте печатную книжку;
- размер половины фифо буфера АЦП 1,2,4,8,16,32,64,128 отсчетов;
- при работе следует иметь ввиду что настройки параметров сбора сохраняются внутри драйвера и при старт/стоп режиме если запрограммировали ЦАП и АЦП то они и будут работать, а если ЦАП стал не нужен, то надо сделать новую установку параметров где ЦАП отключить.

Замечания для модуля E14-140/E154

Пояснения по работе с модулем E14-140/E154.

Библиотека для работы с платой имеет интерфейс аналогичный интерфейсу других плат. Но есть некоторые особенности и ограничения:

- у модуля не настраивается FIFO.
- работа в библиотеке имитирует работу по прерываниям путем перепосылки запросов размером IrqStep к модулю и укладывания их в большой буфер.
- максимальный размер IrqStep ограничен 64 кОтчетов.
- Коррекция данных выполняется пользователем.

Замечания для модуля E20-10

Пояснения по работе с модулем E20-10.

Библиотека для работы с платой имеет интерфейс аналогичный интерфейсу других плат. Но есть некоторые особенности и ограничения:

- у модуля не настраивается FIFO.
- работа в библиотеке имитирует работу по прерываниям путем перепосылки запросов размером IrqStep к модулю и укладывания их в большой буфер.
- максимальный размер IrqStep ограничен 1М Отсчетов.
- Коррекция данных выполняется пользователем.

Замечания для плат L780/L783/L761

Пояснения по работе с платой L780M (rev C).

Эта плата поддерживает потоковый вывод данных на ЦАП. Но есть особенности в формате данных для ЦАП.

Если вывод программируется как для обычной 780 платы, то формат данных в буфере USHORT, а если для потокового вывода 780M, то данные ULONG.

Пример кода:

```
USHORT data1;
```

```
for(int i=0;i<1024;i+=2) data1[i]=((USHORT)(1024.0*sin((2.0*(3.1415*i)/1024.0)))&0xFFF)|0x0000;
```

```
for(int i=1;i<1024;i+=2) data1[i]=((USHORT)(1024.0*sin((2.0*(3.1415*i)/1024.0)))&0xFFF)|0x1000;
```

```
// задается два синуса по двум каналам и из памяти DSP
```

или

```
ULONG data1;
```

```
for(int i=0;i<2048;i++) data1[i]=((USHORT)(512*sin((2.0*(3.1415*i)/1024.0)))&0xFFF)|0x0000;
```

```
// по 0 каналу синус и из буфера PC
```

Такая корявость получилась из-за сохранения совместимости для старых драйверов.

Оглавление

| | |
|---|----|
| Предупреждение | 3 |
| Описание технологии | 4 |
| Установка и настройка PCI плат | 6 |
| Установка и настройка ISA плат | 8 |
| Использование реестра Windows | 9 |
| Создание своего дистрибутива | 10 |
| Низкоуровневое API драйвера | 11 |
| Введение | 11 |
| Описание API DLL библиотеки | 12 |
| Введение | 12 |
| CreateInstance | 14 |
| Подключение и работа с библиотекой (на CPP) | 15 |
| Подключение и работа с библиотекой (на Pascal/Delphi) | 17 |
| Основные функции | 19 |
| OpenLDevice | 19 |
| CloseLDevice | 20 |
| SetParametersStream | 21 |
| RequestBufferStream | 22 |
| InitStartLDevice | 23 |
| StartLDevice | 24 |
| StopLDevice | 25 |
| LoadBios | 26 |
| GetWord_DM | 27 |
| PutWord_DM | 28 |
| GetWord_PM | 29 |
| PutWord_PM | 30 |
| GetArray_DM | 31 |
| PutArray_DM | 32 |
| GetArray_PM | 33 |
| PutArray_PM | 34 |
| SendCommand | 35 |
| PlataTest | 36 |
| IoAsync | 37 |
| EnableCorrection | 41 |
| FillDAQparameters | 42 |
| ReadPlataDescr | 43 |
| WritePlataDescr | 44 |
| ReadFlashWord | 45 |
| WriteFlashWord | 46 |
| EnableFlashWrite | 47 |
| GetParameter | 48 |
| SetParameter | 49 |
| SetLDeviceEvent | 50 |
| Вспомогательные функции | 51 |
| GetSlotParam | 51 |
| Функции для работы с портами ввода/вывода | 52 |
| inbyte | 52 |
| inword | 53 |
| indword | 54 |
| outbyte | 55 |

| | |
|--|-----|
| outword | 56 |
| outdword | 57 |
| inmbyte | 58 |
| imword | 59 |
| inmdword | 60 |
| outmbyte | 61 |
| outmword | 62 |
| outmdword | 63 |
| Типы данных | 64 |
| ADC_PAR | 64 |
| ADC_PAR_0 | 65 |
| ADC_PAR_1 | 70 |
| DAC_PAR | 73 |
| DAC_PAR_0 | 74 |
| DAC_PAR_1 | 76 |
| ASYNC_PAR | 77 |
| PLATA_DESCR | 78 |
| PLATA_DESCR_1450 | 79 |
| PLATA_DESCR_L791 | 80 |
| PLATA_DESCR_E440 | 84 |
| PLATA_DESCR_E140 | 85 |
| PLATA_DESCR_E154 | 86 |
| PLATA_DESCR_E2010 | 87 |
| PLATA_DESCR_U | 88 |
| WORD_IMAGE | 89 |
| SLOT_PAR | 90 |
| Коды ошибок | 92 |
| Типы плат | 93 |
| Другие константы | 94 |
| Справочные данные по платам | 95 |
| Адресное пр-во и команды биос L-1250/L-264/L-305 | 95 |
| Адресное пр-во и команды биос L-7XX | 98 |
| Адресное пр-во и команды биос L-1221 | 103 |
| Адресное пр-во и команды биос L-1450 | 107 |
| Замечания для платы L791 | 113 |
| Замечания для модуля E14-140/E154 | 114 |
| Замечания для модуля E20-10 | 115 |
| Замечания для плат L780/L783/L761 | 116 |
| Оглавление | 117 |