Программное обеспечение

LComp

Руководство программиста

Комплект ПО для разработки приложений (SDK)
Windows XP/Vista/W7
L-761/L-780/L-783/L-791/E14-440/E14-140/E20-10/E154

Москва. Апрель. 2013
Ревизия 7.0.0.0

ООО "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

E-Mail:

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

Представители в регионах:

Украина: HOLIT Data Sistems, 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 7.0.0.1

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-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:

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 платы L-791 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. К сожалению эта плата поддерживает только 32-битный режим BusMaster и поэтому в ней реализован комбинированный режим - в небольшой буфер расположенный в 32-битном адресном пространстве данные поступают по BusMaster, а в кольцевой

буфер они перекладываются по прерываниям.

Для USB модулей реализован также принцип кольцевого буфера путем циклической перепосылки запросов на ввод данных.

Большинство плат и модулей поддерживают непрерывный вывод данных на опциональный ЦАП. Механизм вывода потока аналогичен механизму ввода данных с АЦП. У PCI плат без BusmMster это вывод по прерываниям. У USB циклическая перепосылка запросов. У L-791 комбинированный метод с прерываниями и BusMaster (кроме этого в данные для ЦАП необходимо вставить маркер генерайции прерывания с нужным шагом)

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 БИОСЕе 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 именное 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 **Plag&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).

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:

- ldevpcim.sys - WDM драйвер для PCI плат L791 - копировать в WINDOWS\SYSTEM32\DRIVERS;
- ldevpci.sys - WDM драйвер для PCI плат - копировать в WINDOWS\SYSTEM32\DRIVERS;
- ldevusbu.sys - WDM драйвер для USB модулей - копировать в WINDOWS\SYSTEM32\DRIVERS;
- ldevs.sys - поддерживающий драйвер - копировать в WINDOWS\SYSTEM32\DRIVERS;
- lcardpci.inf - INF файл для PCI плат - копировать в WINDOWS\INF;
- ldevpcim.inf - INF файл для PCI плат L791 - копировать в WINDOWS\INF;
- ldevusbu.inf - INF файл для USB модулей - копировать в WINDOWS\INF;
- lcomp.dll - DLL библиотека для работы с платами - класть лучше всего в один каталог с приложением;

Все это будет работать под операционными системами Windows XP/Vista/W7. Оригинальный скрипт инсталляции написан с помощью бесплатной программы NSIS(www.nullsoft.com) и прилагается.

Низкоуровневое API драйвера

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.

См. исходники библиотеки, если есть необходимость.

Базовый класс (LUnknown)

Наследует от :

Описание :

Базовый класс LUnknown.

Определение :

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

Подробнее :

Библиотека функций создана для того, чтобы упростить связь приложений с драйверами. Ниже приведен полный список функций поддерживаемых библиотекой - фактически это файл ifc_ldev.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.

Описание API DLL библиотеки (IDaqLDevice)

Класс IDaqLDevice

Наследует от : LUnknown

Описание :

Основой интерфейс для работы с устройствами.

Определение :

```
struct IDaqLDevice:LUnknown
{

   IFC(ULONG)  inbyte ( ULONG offset, PUCHAR 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, PUCHAR 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, PUCHAR 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, PUCHAR 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 Data) = 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;


    IFC(ULONG)  SetLDeviceEvent(HANDLE hEvent,ULONG EventId = L_STREAM_ADC) = 0;


};
```

Подробнее :

Библиотека функций создана для того, чтобы упростить связь приложений с драйверами. Ниже приведен полный список функций поддерживаемых библиотекой - фактически это файл ifc_ldev.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.

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.

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_ERRROR - возвращается когда невозможно создать объект. L_NOTSUPPORTED - если в слоте установлена плата, которая не поддерживается этой библиотекой.
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 PCI или USB платы нет. Причем определить, что за плата соответствует конкретному слоту, можно только открыв его и прочитав информацию GetSlotParam и ReadPlataDescr (+ для 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 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");
    LUnknown* 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 PCI или USB платы нет. Причем определить, что за плата соответствует конкретному слоту, можно только открыв его и прочитав информацию GetSlotParam и ReadPlataDescr (+ для 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 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:LUnknown;
  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.

Функции для работы с портами ввода/вывода

Введение

В данном разделе собраны функции для работы с портами ввода/вывода плат.

inbyte

Описание :

Ввод байта из I/O порта.

Определение :

```
   IFC(ULONG)  inbyte ( ULONG offset, PUCHAR data, ULONG len=1, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу. Пояснение 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

Описание :

Ввод слова из I/O порта.

Определение :

```
   IFC(ULONG)  inword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу. Пояснение 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

Описание :

Ввод двойного слова из I/O порта.

Определение :

```
   IFC(ULONG)  indword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу. Пояснение 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

Описание :

Вывод байта в I/O порт.

Определение :

```
   IFC(ULONG)  outbyte ( ULONG offset, PUCHAR data, ULONG len=1, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу.
Пояснение 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

Описание :

Вывод слова в I/O порт.

Определение :

```
   IFC(ULONG)  outword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0)
= 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу.
Пояснение 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

Описание :

Вывод двойного слова в I/O порт.

Определение :

```
   IFC(ULONG)  outdword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу.
Пояснение 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

Описание :

Ввод байта из памяти.

Определение :

```
   IFC(ULONG)  inmbyte ( ULONG offset, PUCHAR data, ULONG len=1, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу. Пояснение 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

Описание :

Ввод слова из памяти.

Определение :

```
   IFC(ULONG)  inmword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0)
= 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу. Пояснение 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.

inmdword

Описание :

Ввод двойного слова из памяти.

Определение :

```
   IFC(ULONG)  inmdword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0) =
0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу. Пояснение 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

Описание :

Вывод байта в память.

Определение :

```
   IFC(ULONG)  outmbyte ( ULONG offset, PUCHAR data, ULONG len=1, ULONG key=0)
= 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу.
Пояснение 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

Описание :

Вывод слова в память.

Определение :

```
   IFC(ULONG)  outmword ( ULONG offset, PUSHORT data, ULONG len=2, ULONG key=0)
= 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу.
Пояснение 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

Описание :

Вывод двойного слова в память.

Определение :

```
   IFC(ULONG)  outmdword( ULONG offset, PULONG data, ULONG len=4, ULONG key=0)
= 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

В данной реализации нет проверки выхода смещения за разрешенную границу.
Пояснение 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.

Основные функции

Введение

В данном разделе собраны основные функции для работы с модулями/платами.

GetWord_DM

Описание :

Читает слово из памяти данных DSP/модуля.

Определение :

```
IFC(ULONG)  GetWord_DM(USHORT Addr, PUSHORT Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Data - прочитанные данные;
```

Возвращает :

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

Подробнее :

PutWord_DM

Описание :

Записывает слово в память данных DSP/модуля.

Определение :

```
IFC(ULONG)  PutWord_DM(USHORT Addr, USHORT Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Data - записываемые данные;
```

Возвращает :

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

Подробнее :

PutWord_PM

Описание :

Записывает слово в память программ DSP/модуля.

Определение :

```
IFC(ULONG)  PutWord_PM(USHORT Addr, ULONG Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Data - записываемые данные;
```

Возвращает :

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

Подробнее :

GetWord_PM

Описание :

Читает слово из памяти программ DSP/модуля.

Определение :

```
IFC(ULONG)  GetWord_PM(USHORT Addr, PULONG Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Data - прочитанные данные;
```

Возвращает :

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

Подробнее :

GetArray_DM

Описание :

Читает массив слов из памяти данных DSP.

Определение :

```
IFC(ULONG)  GetArray_DM(USHORT Addr, ULONG Count, PUSHORT Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Count - размер массива в словах;
Data - возвращаемые данные;
```

Возвращает :

```
L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
Data - возвращаемые данные;
```

Подробнее :

PutArray_DM

Описание :

Записывает массив слов в память данных DSP.

Определение :

```
IFC(ULONG)  PutArray_DM(USHORT Addr, ULONG Count, PUSHORT Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Count - размер массива в словах;
Data - записываемые данные;
```

Возвращает :

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

Подробнее :

PutArray_PM

Описание :

Записывает массив слов в память программ DSP.

Определение :

```
IFC(ULONG)  PutArray_PM(USHORT Addr, ULONG Count, PULONG Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Count - размер массива в двойных словах;
Data - записываемые данные;
```

Возвращает :

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

Подробнее :

GetArray_PM

Описание :

Читает массив слов из памяти программ DSP.

Определение :

```
IFC(ULONG)  GetArray_PM(USHORT Addr, ULONG Count, PULONG Data) = 0;
```

Параметры :

```
Addr - адрес переменной;
Count - размер массива в двойных словах;
Data - возвращаемые данные;
```

Возвращает :

```
L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
Data - возвращаемые данные;
```

Подробнее :

SendCommand

Описание :

Посылает выбранную команду в DSP.

Определение :

```
IFC(ULONG)  SendCommand(USHORT Cmd) = 0;
```

Параметры :

```
Cmd - код команды;
```

Возвращает :

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

Подробнее :

PlataTest

Описание :

Тест на наличие платы и успешную загрузку.

Определение :

```
IFC(ULONG)  PlataTest() = 0;
```

Параметры :

Возвращает :

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

Подробнее :

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

GetSlotParam

Описание :

Функция возвращает информацию для указанного виртуального слота.

Определение :

```
IFC(ULONG)  GetSlotParam(PSLOT_PAR slPar) = 0;
```

Параметры :

```
slPar - переменная, в которой будут возвращены параметры;
```

Возвращает :

```
L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
slPar - параметры установленные для данного слота;
```

Подробнее :

OpenLDevice

Описание :

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

Определение :

```
IFC(HANDLE) OpenLDevice() = 0;
```

Параметры :

Возвращает :

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

Подробнее :

Для каждой платы установленной в компьютер драйвер формирует линк по следующему принципу: LDev## (где ## - номер 00,01..).Номер в названии линка - это виртуальный слот. Номер виртуального слота, для которого будет выполнена функция OpenLDevice, передается как параметр в функции CreateInstance.

CloseLDevice

Описание :

Эта функция вызывается при завершении работы с платой.

Определение :

```
IFC(ULONG)  CloseLDevice() = 0;
```

Параметры :

Возвращает :

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

Подробнее :

После вызова этой функции значение дескриптора устройства больше недействительно и не может использоваться при вызове функций библиотеки. Кроме этого еще происходит удаление выделенной в функции RequsetBufferStream памяти для кольцевого буфера.

SetParametersStream

Описание :

Вызов этой функции настраивает плату АЦП/ЦАП на заданные параметры ввода или вывода данных.

Определение :

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

Параметры :

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

Возвращает :

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

Подробнее :

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 переменой синхронизации.

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

Описание :

Функция служит для выделения памяти под большой кольцевой буфер.

Определение :

```
   IFC(ULONG)  RequestBufferStream(ULONG *Size, ULONG StreamId = L_STREAM_ADC)
= 0;
```

Параметры :

```
Size - размер большого буфера в USHORT;
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;
```

Подробнее :

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.

FillDAQparameters

Описание :

Заполняет значениями внутреннюю структуру параметров сбора данных.

Определение :

```
IFC(ULONG)  FillDAQparameters(PDAQ_PAR sp) = 0;
```

Параметры :

```
sp - структура с параметрами сбора данных;
```

Возвращает :

```
L_SUCCESS - в случае успеха;
L_ERROR - в случае ошибки;
sp - в структуре обновлены поля Rate,Kadr,NCh с учетом возможностей платы.
```

Подробнее :

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

InitStartLDevice

Описание :

Функция инициализирует внутренние переменные драйвера перед началом сбора.

Определение :

```
IFC(ULONG)  InitStartLDevice() = 0;
```

Параметры :

Возвращает :

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

Подробнее :

Надо вызывать перед вызовом функции StartLDevice.

StartLDevice

Описание :

Функция запускает сбор данных с платы в большой кольцевой буфер.

Определение :

```
IFC(ULONG)  StartLDevice() = 0;
```

Параметры :

Возвращает :

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

Подробнее :

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

StopLDevice

Описание :

Функция останавливает сбор данных с платы в большой кольцевой буфер.

Определение :

```
IFC(ULONG)  StopLDevice() = 0;
```

Параметры :

Возвращает :

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

Подробнее :

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.

LoadBios

Описание :

Загрузка BIOS в плату.

Определение :

```
IFC(ULONG)  LoadBios(char *FileName) = 0;
```

Параметры :

```
FileName - имя файла прошивки БИОС без расширения (например l783);
```

Возвращает :

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

Подробнее :

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

IoAsync

Описание :

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

Определение :

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

Параметры :

```
sp - структура с параметрами запроса и результатами его выполнения;
```

Возвращает :

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

Подробнее :

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

- **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);
```

- **Плата 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);
```

  - Для вывода сразу двух 16-битных отсчетов на ЦАП (для модуля ревизии B)

```
s_Type -L_ASYNC_DAC_OUT;
Mode - 2;
Data[0] - данные для ЦАП 0;
Data[1] - данные для ЦАП 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);
```

ReadPlataDescr

Описание :

Чтение пользовательского Flash.

Определение :

```
IFC(ULONG)  ReadPlataDescr(LPVOID pd) = 0;
```

Параметры :

```
pd - указатель на структуру PLATA_DESCR_U/PLATA_DESCR_U2 куда будут записаны
считанные данные;
```

Возвращает :

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

Подробнее :

WritePlataDescr

Описание :

Запись пользовательского Flash.

Определение :

```
IFC(ULONG)  WritePlataDescr(LPVOID pd, USHORT Ena) = 0;
```

Параметры :

```
pd - указатель на структуру PLATA_DESCR_U/PLATA_DESCR_U2 из которой будут
записаны данные;
Ena - разрешение(1) / запрещение(0) записи служебной части пользовательского
Flash;
```

Возвращает :

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

Подробнее :

ReadFlashWord

Описание :

Чтение слова из пользовательского Flash.

Определение :

```
IFC(ULONG)  ReadFlashWord(USHORT FlashAddress, PUSHORT Data) = 0;
```

Параметры :

```
FlashAddress - адрес, с которого читать;
Data - прочитанное слово;
```

Возвращает :

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

Подробнее :

WriteFlashWord

Описание :

Запись слова в пользовательский Flash.

Определение :

```
IFC(ULONG)  WriteFlashWord(USHORT FlashAddress, USHORT Data) = 0;
```

Параметры :

```
FlashAddress - адрес, по которому писать;
Data - записываемое слово;
```

Возвращает :

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

Подробнее :

EnableFlashWrite

Описание :

Разрешение записи в пользовательский Flash.

Определение :

```
IFC(ULONG)  EnableFlashWrite(USHORT Flag) = 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

EnableCorrection

Описание :

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

Определение :

```
IFC(ULONG)  EnableCorrection(USHORT Ena=1) = 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

GetParameter

Описание :

Функция возвращает некоторые полезные данные о модуле и позволяет вместе с SetParameter хранить временно данные пользователя.

Определение :

```
IFC(ULONG)  GetParameter(ULONG name, PULONG param) = 0;
```

Параметры :

```
name – идентификатор ULONG переменной;
param – считанное значение ULONG переменной;
```

Возвращает :

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

Подробнее :

С помощью вызова GetParameter с name определенными ниже можно получит ряд полезных значений:

```
#define L_BOARD_TYPE       10000  // собственно поле sl.BoardType
#define L_POINT_SIZE       10001  // размер в байтах отсчета АЦП
#define L_SYNC_ADDR_LO     10002  // адрес переменной sync (счетчик АЦП)
#define L_SYNC_ADDR_HI     10003  // адрес переменной sync (счетчик АЦП)
#define L_DATA_ADDR_LO     10004  // адрес массива data ( АЦП)
#define L_DATA_ADDR_HI     10005  // адрес массива data ( АЦП)
#define L_SYNC1_ADDR_LO    10006  // адрес переменной sync (счетчик ЦАП)
#define L_SYNC1_ADDR_HI    10007  // адрес переменной sync (счетчик ЦАП)
#define L_DATA1_ADDR_LO    10008  // адрес массива data ( ЦАП)
#define L_DATA1_ADDR_HI    10009  // адрес массива data ( ЦАП)

#define L_USER_BASE        10100  // ранее сохраненные любые пользовательские
                                  // 128 ULONG числа
```

SetParameter

Описание :

Функцияпозволяет хранить временно данные пользователя и получать их с помощью GetParameter.

Определение :

```
IFC(ULONG)  SetParameter(ULONG name, PULONG param) = 0;
```

Параметры :

```
name – идентификатор ULONG переменной;
param – сохраняемое значение ULONG переменной;
```

Возвращает :

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

Подробнее :

С помощью вызова SetParameter с name определенными ниже можно сохранить до 128 своих значений ULONG и потом их получить через вызов GetParameter.

```
#define L_USER_BASE        10100
```

SetLDeviceEvent

Описание :

Функция служит для установки события в драйвере. Работа события облегчает ожидание готовности данных от платы при однократном заполнении буфера.

Определение :

```
IFC(ULONG)  SetLDeviceEvent(HANDLE hEvent,ULONG EventId = L_STREAM_ADC) = 0;
```

Параметры :

```
hEvent - handle события от CreateEvent;
EventId – идентификатор события на который установлен event;
```

Возвращает :

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

Подробнее :

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 (L780M):`

Расширенное API DLL библиотеки (IDaqLDevice2)

Класс IDaqLDevice2

Наследует от : LUnknown

Описание :

Расширенный интерфейс для работы с устройствами.

Определение :

```
struct IDaqLDevice2:LUnknown
{
   IFC(ULONG)  InitStartLDeviceEx(ULONG StreamId) = 0;

   IFC(ULONG)  StartLDeviceEx(ULONG StreamId) = 0;

   IFC(ULONG)  StopLDeviceEx(ULONG StreamId) = 0;

};
```

Подробнее :

Для раздельной независимой работы АЦП и ЦАП плат и модулей в библиотеке введен дополнительный интерфейс . Этот интерфейс предоставляет три функции: InitStartLDeviceEx, StartLDeviceEx, StopLDeviceEx. Данные функции полностью аналогичны функциям основного интерфейса, но для раздельной работы с ЦАП и АЦП введен параметр StreamId (L_STREAM_ADC/L_STREAM_DAC). Соответственно чтобы работать с ЦАП и АЦП модулей необходимо сначала получить основной интерфейс, а затем этот дополнительный интерфейс . Запускать и останавливать сбор/выдачу данных необходимо функциями этого дополнительного интерфейса и не следует смешивать их вызов с аналогичными функциями основного интерфейса. Установка параметров работы ЦАП и АЦП при работе с дополнительным интерфейсом аналогично настройке при работе с основным интерфейсом. Пример работы L7XX2.OSC.

InitStartLDeviceEx

Описание :

Функция инициализирует внутренние переменные драйвера перед началом сбора.

Определение :

```
IFC(ULONG)  InitStartLDeviceEx(ULONG StreamId) = 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

Надо вызывать перед вызовом функции StartLDevice.

StartLDeviceEx

Описание :

Функция запускает сбор данных с платы в большой кольцевой буфер.

Определение :

```
IFC(ULONG)  StartLDeviceEx(ULONG StreamId) = 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

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

StopLDeviceEx

Описание :

Функция останавливает сбор данных с платы в большой кольцевой буфер.

Определение :

```
IFC(ULONG)  StopLDeviceEx(ULONG StreamId) = 0;
```

Параметры :

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

Возвращает :

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

Подробнее :

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.

Типы плат/модулей

Описание :

Определения типов плат/модулей. Используется в поле BoardType структуры SLOT_PAR.

Определение :

```
#define NONE  0 // no board in slot
#define L1250 1 // L1250 board
#define N1250 2 // N1250 board (may be not work)
#define L1251 3 // L1251 board
#define L1221 4 // L1221 board
#define PCIA  5 // PCI rev A board
#define PCIB  6 // PCI rev B board
#define L264  8 // L264 ISA board
#define L305  9 // L305 ISA board
#define L1450C 10
#define L1450 11
#define L032 12
#define HI8 13
#define PCIC 14

#define L791     19

#define E440     30
#define E140     31
#define E2010    32
#define E270     33
#define CAN_USB  34
#define AK9      35
#define LTR010   36
#define LTR021   37
#define E154     38
#define E2010B   39
#define LTR031   40
#define LTR030   41
```

Подробнее :

Определения для property

Описание :

Наглядные определения для property. Используются с GetParemeter/SetParameter.

Определение :

```
#define L_BOARD_TYPE       10000 //defines for GetParemeter/SetParameter common
for all boards
#define L_POINT_SIZE       10001
#define L_SYNC_ADDR_LO     10002
#define L_SYNC_ADDR_HI     10003
#define L_DATA_ADDR_LO     10004
#define L_DATA_ADDR_HI     10005
#define L_SYNC1_ADDR_LO     10006
#define L_SYNC1_ADDR_HI     10007
#define L_DATA1_ADDR_LO    10008
#define L_DATA1_ADDR_HI    10009

#define L_USER_BASE        10100 //128 user prpoperty to store data //next prop
10228
```

Подробнее :

Коды ошибок

Описание :

Определение кодов ошибок.

Определение :

```
#define L_SUCCESS 0    // ERROR CODES
#define L_NOTSUPPORTED 1
#define L_ERROR 2
#define L_ERROR_NOBOARD 3
#define L_ERROR_INUSE 4
```

Подробнее :

Определения для типизации структуры DAQ_PAR

Описание :

Определения для типизации структуры DAQ_PAR

Определение :

```
#define L_ADC_PARAM 1 // define s_Type for FillDAQparameters
#define L_DAC_PARAM 2

#define L_ASYNC_ADC_CFG 3
#define L_ASYNC_TTL_CFG 4
#define L_ASYNC_DAC_CFG 5

#define L_ASYNC_ADC_INP 6
#define L_ASYNC_TTL_INP 7

#define L_ASYNC_TTL_OUT 8
#define L_ASYNC_DAC_OUT 9
```

Подробнее :

Типы потоков данных

Описание :

Типы потоков данных

Определение :

```
#define L_STREAM_ADC 1
#define L_STREAM_DAC 2
#define L_STREAM_TTLIN 3
#define L_STREAM_TTLOUT 4
```

Подробнее :

Определения EventId для событий SetLDeviceEvent

Описание :

Определения EventId для событий SetLDeviceEvent

Определение :

```
#define L_EVENT_ADC_BUF 1
#define L_EVENT_DAC_BUF 2
```

Подробнее :

SLOT_PAR

Наследует от :

Описание :

Структура описывает параметры виртуального слота.

Определение :

```
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;
```

Подробнее :

- **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 относительного базового адреса;

Примечание:

Структура SLOT_PAR используется совместно с вызовом GetSlotParam для получения параметров виртуальных слотов. Большинство полей этой структуры утратило свое значение. Интерес представляет **BoardType** при поиске нужной платы в виртуальных слотах. Еще смысл имеет поле **DSPType** для модулей у которых есть сигнальный процессор ADSP. Для PCI плат L-761/L-780/L-783 имеют смысл поля описывающие адресные регионы и поля регистров. Они нужны если требуется работа с функциями для доступа к портам (inbyte итп). Для L-791 в таком случае интересны поля Mem и MemL.

DAQ_PAR

Наследует от :

Описание :

Базовая структура для описания параметров сбора данных.

Определение :

```
typedef struct _DAQ_PARAM_
{
   ULONG s_Type;
   ULONG FIFO;
   ULONG IrqStep;
   ULONG Pages;
} DAQ_PAR, *PDAQ_PAR;
```

Подробнее :

ASYNC_PAR

Наследует от : DAQ_PAR

Описание :

Структура для передачи параметров асинхронного сбора/выдачи данных при вызове IoAsync.

Определение :

```
typedef struct _ASYNC_PARAM_
#ifndef LABVIEW_FW
 : public DAQ_PAR
#endif
{
   double dRate;
   ULONG Rate;
   ULONG NCh;
   ULONG Chn[128];
   ULONG Data[128];
   ULONG Mode;
} ASYNC_PAR, *PASYNC_PAR;
```

Подробнее :

- **ULONG s_Type** (DAQ_PAR) - указывает для какой опреации 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);
- **ULONG Data[128]** - массив для данных;
- **double dRate** - частота опроса каналов в кадре (кГц);
- **ULONG Rate** - частота опроса каналов в кадре (в кодах для процессора);
- **ULONG NCh** - количество опрашиваемых каналов;
- **ULONG Chn[128]** - массив с номерами каналов и усилением на них. Описывает порядок опроса каналов;
- **ULONG FIFO** (DAQ_PAR) - размер половины аппаратного буфера FIFO на плате;
- **ULONG IrqStep** (DAQ_PAR) - шаг генерации прерываний;
- **ULONG Pages** (DAQ_PAR) - размер кольцевого буфера в шагах прерываний;
- **ULONG Mode** - задает различные режимы при конфигурации.

Примечание:

Структура ASYNC_PAR используется совместно с вызовом IoAsync. Часть полей наследуются из структуры DAQ_PAR. Как заполнять или что читать из этой структуры см. описание функции IoAsync.

DAC_PAR_0

Наследует от : DAQ_PAR

Описание :

Структура для передачи параметров работы ЦАП в потоковом режиме.

Определение :

```
typedef struct _DAC_PARAM_U_0
#ifndef LABVIEW_FW
 : public DAQ_PAR
#endif
{
   ULONG AutoInit;
   double dRate;
   ULONG Rate;
   ULONG IrqEna;
   ULONG DacEna;
   ULONG DacNumber;
} DAC_PAR_0, *PDAC_PAR_0;
```

Подробнее :

- **ULONG s_Type** (DAQ_PAR) - тип структуры (должен быть L_DAC_PARAM);
- **ULONG AutoInit** - флаг указывающий на тип сбора/выдачи 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** (DAQ_PAR) - размер половины аппаратного буфера FIFO на плате;
- **ULONG IrqStep** (DAQ_PAR) - шаг генерации прерываний;
- **ULONG Pages** (DAQ_PAR) - 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.

Особенности трактовки полей этой структуры для различных плат:

- L-761/780/783 - полностью так, как описано кроме:
  - **DacNumber** - не задействован (номер ЦАП задается в самих данных);
  - **IrqEna,Pages**- прерывания и реальный кольцевой буфер работают только в L780C, для остальных плат Pages всегда надо задавать 2;
  - **IrqStep** - должен быть равен FIFO;
- E-440:
  - **DacNumber** - не задействован (номер ЦАП задается в самих данных);
  - **IrqEna,Pages** - прерывания и реальный кольцевой буфер (число страниц задавать не меньше 2);
  - **IrqStep** - должен быть равен FIFO (и проверяется на кратность 32 отсчетам);

- E-140 (модификации M):
    - **DacNumber** - не задействован (номер ЦАП задается в самих данных);
    - **IrqEna,Pages**- прерывания и реальный кольцевой буфер (число страниц задавать не меньше 2);
    - **IrqStep** - должен быть равен FIFO (и проверяется на кратность 2048 отсчетам);

DAC_PAR_1

Наследует от : DAQ_PAR

Описание :

Структура для передачи параметров работы ЦАП в потоковом режиме.

Определение :

```
typedef struct _DAC_PARAM_U_1
#ifndef LABVIEW_FW
 : public DAQ_PAR
#endif
{
   ULONG AutoInit;
   double dRate;
   ULONG Rate;
   ULONG IrqEna;
   ULONG DacEna;
   ULONG Reserved1;
} DAC_PAR_1, *PDAC_PAR_1;
```

Подробнее :

- **ULONG s_Type**$_{(DAQ\_PAR)}$ - 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**$_{(DAQ\_PAR)}$ - размер половины аппаратного буфера FIFO на плате;
- **ULONG IrqStep**$_{(DAQ\_PAR)}$ - шаг генерации прерываний;
- **ULONG Pages**$_{(DAQ\_PAR)}$ - 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.

Особенности трактовки полей этой структуры для различных плат:

- L-791 - полностью так, как описано кроме:
  - **AutoInit** - не используется;
  - **IrqStep** - задает шаг прерываний, но признаки генерации в соответсвии с этим шагом надо в данных расставить
  - **Pages**- задает размер кольцевого буфера как Pages*IrqSetp;
  - **FIFO** - не используется;

DAC_PAR

Наследует от :

Описание :

Обобщенная структура для удобства работы со структурами праметров ЦАП.

Определение :

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

Подробнее :

ADC_PAR_0

Наследует от : DAQ_PAR

Описание :

Структура служит для передачи параметров сбора данных в плату.

Определение :

```
typedef struct _ADC_PARAM_U_0
#ifndef LABVIEW_FW
 : public DAQ_PAR
#endif
{
   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;
```

Подробнее :

- **ULONG s_Type**$_{(DAQ\_PAR)}$ - 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**$_{(DAQ\_PAR)}$ - размер половины аппаратного буфера FIFO на плате;
- **ULONG IrqStep**$_{(DAQ\_PAR)}$ - шаг генерации прерываний;
- **ULONG Pages**$_{(DAQ\_PAR)}$ - 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. Особенности трактовки полей этой структуры для различных плат:

- **L-761/780/783**
  - **s_Type** - должен быть L_ADC_PARAM;
  - **AutoInit** - (**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** - 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** - канал, 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.
- **E14-440/E14-140/E154**
    - **s_Type** - должен быть L_ADC_PARAM;
    - **AutoInit** - (**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} .

ADC_PAR_1

Наследует от : DAQ_PAR

Описание :

Структура служит для передачи параметров сбора данных в плату.

Определение :

```
typedef struct _ADC_PARAM_U_1
#ifndef LABVIEW_FW
 : public DAQ_PAR
#endif
{
   ULONG AutoInit;
   double dRate;
   double dKadr;
   ULONG Reserved1;
   ULONG DM_Ena;      // data marker ena/dis
   ULONG Rate;
   ULONG Kadr;
   ULONG StartCnt;    // задержка сбора при старте в количестве кадров
   ULONG StopCnt;     // остановка сбора после количества кадров
   ULONG SynchroType; // in e20-10 start type
   ULONG SynchroMode; // advanced synchro mode + chan number
   ULONG AdPorog;     // порог синхронизации
   ULONG SynchroSrc;  // in e20-10 clock source
   ULONG AdcIMask;    // change from Reserved4 to AdcIMask for e20-10 adc input
config
   ULONG NCh;
   ULONG Chn[128];
   ULONG IrqEna;
   ULONG AdcEna;
} ADC_PAR_1, *PADC_PAR_1;
```

Подробнее :

- **ULONG s_Type**$_{(DAQ\_PAR)}$ - 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 AdcIMask** - 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**$_{(DAQ\_PAR)}$ - размер половины аппаратного буфера FIFO на плате;
- **ULONG IrqStep**$_{(DAQ\_PAR)}$ - шаг генерации прерываний;
- **ULONG Pages**$_{(DAQ\_PAR)}$ - 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 _1 используется совместно с вызовом FillDAQparameters для настройки параметров ввода данных с платы АЦП. Особенности трактовки полей этой структуры для различных плат:

- **L-791**
  - **s_Type** - тип структуры (должен быть L_ADC_PARAM);
  - **AutoInit** - 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** - шаг генерации прерываний (max ограничен 2048 отсчетами);
  - **Pages** - размер кольцевого буфера в шагах прерываний;
  - **IrqEna** - разрешение генерации прерывания от платы (1/0);
  - **AdcEna** - разрешение работы АЦП (1/0);
- **E20-10**
  - **s_Type** - тип структуры (должен быть L_ADC_PARAM);
  - **AutoInit** - 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 тактовыйх 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 1М отсчтетов;
- **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 отсчётов АЦП;
    - **StopCnt** - останов сбора данных после задаваемого здесь кол-ва собранных кадров отсчётов 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;

ADC_PAR

Наследует от :

Описание :

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

Определение :

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

Подробнее :

PLATA_DESCR

Наследует от :

Описание :

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

Определение :

```
typedef struct __PLATA_DESCR
{
   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 ЦАП
   USHORT Reserv1[7];  // зарезервировано
   USHORT KoefADC[8];  // калибровочные коэф. АЦП
   USHORT KoefDAC[4];  // калбровочнык коэф. ЦАП
   USHORT Custom[32];  // пользовательское место
} PLATA_DESCR, *PPLATA_DESCR;
```

Подробнее :

PLATA_DESCR_L791

Наследует от :

Описание :

Структура описывает FLASH на PCI плате L-791.

Определение :

```
typedef struct __PLATA_DESCR_L791
{
   USHORT CRC16;         // контрольная сумма
   char SerNum[16];      // серийный номер платы
   char BrdName[16];     // название платы
   char Rev;             // ревизия платы
   char DspType[5];      // тип DSP
   long Quartz;          // частота кварца
   USHORT IsDacPresent; // наличие ЦАП
   float KoefADC[16];    // калибровочные коэф. АЦП
   float KoefDAC[4];     // калибровочные коэф. ЦАП
   USHORT Custom;        // пользовательское место
} PLATA_DESCR_L791, *PPLATA_DESCR_L791;
```

Подробнее :

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, расчитанная 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)
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)

Название поля	Назначение и допустимые значения поля
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 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

Название поля	Назначение и допустимые значения поля
	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 смешение 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 ±2.5В (усиление x4), получены следующие данные:

$\mathbf{X_1}$=1000, $\mathbf{X_2}$=-1000,$\mathbf{X_3}$=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]

$\mathbf{Y_1}$**=(B+1000)*A,** $\mathbf{Y_2}$**=(B-1000)*A,** $\mathbf{Y_3}$**=B*A**

Пример 2:

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

$\mathbf{X_1}$=1000, $\mathbf{X_2}$=-1000, $\mathbf{X_3}$=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**]**

$\mathbf{Y_1}$=(**B**+1000)***A,** $\mathbf{Y_2}$=(**B**-1000)***A,** $\mathbf{Y_3}$**=B*A**

PLATA_DESCR_E440

Наследует от :

Описание :

Структура описывает FLASH на USB модуле E14-440.

Определение :

```
typedef struct __PLATA_DESCR_E440
{
   char SerNum[9];    // серийный номер платы
   char BrdName[7];   // название платы
   char Rev;          // ревизия платы
   char DspType[5];   // тип DSP
   char IsDacPresent; // наличие ЦАП
   long Quartz;       // частота кварца
   char Reserv2[13];  // зарезервировано
   USHORT KoefADC[8]; // калибровочные коэф. АЦП
   USHORT KoefDAC[4]; // калибровочные коэф. ЦАП
   USHORT Custom[32]; // пользовательское место
} PLATA_DESCR_E440, *PPLATA_DESCR_E440;
```

Подробнее :

PLATA_DESCR_E140

Наследует от :

Описание :

Структура описывает FLASH на USB модуле E14-140.

Определение :

```
typedef struct __PLATA_DESCR_E140
{
   char SerNum[9];    // серийный номер платы
   char BrdName[11];  // название платы
   char Rev;          // ревизия платы
   char DspType[11];  // тип DSP
   char IsDacPresent; // наличие ЦАП
   long Quartz;       // частота кварца
   char Reserv2[3];   // зарезервировано
   float KoefADC[8];  // калибровочные коэф. АЦП
   float KoefDAC[4];  // калибровочные коэф. ЦАП
   USHORT Custom[20]; // пользовательское место
} PLATA_DESCR_E140, *PPLATA_DESCR_E140;
```

Подробнее :

PLATA_DESCR_E154

Наследует от :

Описание :

Структура описывает FLASH на USB модуле E154.

Определение :

```
typedef struct __PLATA_DESCR_E154
{
   char SerNum[9];    // серийный номер платы
   char BrdName[11];  // название платы
   char Rev;          // ревизия платы
   char DspType[11];  // тип DSP
   char IsDacPresent; // наличие ЦАП
   long Quartz;       // частота кварца
   char Reserv2[3];   // зарезервировано
   float KoefADC[8];  // калибровочные коэф. АЦП
   float KoefDAC[4];  // калибровочные коэф. ЦАП
   USHORT Custom[20]; // пользовательское место
} PLATA_DESCR_E154, *PPLATA_DESCR_E154;
```

Подробнее :

PLATA_DESCR_E2010

Наследует от :

Описание :

Структура описывает FLASH на USB модуле E20-10.

Определение :

```
typedef struct __PLATA_DESCR_E2010
{
    char BrdName[16]; // серийный номер платы
    char SerNum[16];  // название платы
    char DspType[16]; // тип DSP
    ULONG Quartz;     // частота кварца
    char Rev;         // ревизия платы
    char IsDacPresent;// наличие ЦАП
    float KoefADC[24];// калибровочные коэф. АЦП
    float KoefDAC[4]; // калибровочные коэф. ЦАП
    USHORT Custom[44];// пользовательское место
    USHORT CRC;       // котрольная сумма
} PLATA_DESCR_E2010, *PPLATA_DESCR_E2010;
```

Подробнее :

PLATA_DESCR_U/U2

Наследует от :

Описание :

Это обобщенная структура для удобства работы с флешами разных плат.

Определение :

```
typedef union __PLATA_DESCR_U
{
   PLATA_DESCR t1;
   PLATA_DESCR_1450 t2;
   PLATA_DESCR_L791 t3;
   PLATA_DESCR_E440 t4;
   PLATA_DESCR_E140 t5;
   PACKED_PLATA_DESCR_E140 pt5;

   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;
   PLATA_DESCR_E154 t7;
   PACKED_PLATA_DESCR_E154 pt7;

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

Подробнее :

Как можно ...

Введение

В данном разделе собраны рецепты как осуществить некоторые операции при работе с платами/модулями.

... прочитать одиночный отсчет с АЦП

Для этого нужно выполнить шаги приведенные ниже.

- Подключить библиотеку LComp так как описано в главе ***"Подключение и работа с библиотекой (на CPP)"*** раздела ***"Описание API DLL библиотеки"***

- После вызова **OpenLDevice,LoadBios** и **ReadPlataDecr** выполнить следующий код:

```
ASYNC_PAR pp;
pp.s_Type = L_ASYNC_ADC_INP;
pp.Chn[0] = 0x00; // 0 канал дифф. подключение (в общем случае лог. номер канала)
pI->IoAsync(&pp);
cout << (short)pp.Data[0] << endl; // в Data[0] код АЦП
```

  При этом на экран будет выведен один отсчет с 0 канала АЦП.

- Завершить работу с библиотекой так как описано в главе ***"Подключение и работа с библиотекой (на CPP)"*** раздела ***"Описание API DLL библиотеки"***

Прочитать таким образом кадр отсчетов с АЦП невозможно. Только один отсчет с одного канала.

Более подробно о возможностях функции **IoAsync** можно прочитать в ее описании

... вывести данные на TTL выходы

Для этого нужно выполнить шаги приведенные ниже.

- Подключить библиотеку LComp так как описано в главе ***«Подключение и работа с библиотекой (на CPP)»*** раздела ***«Описание API DLL библиотеки»***
- После вызова **OpenLDevice, LoadBios** и **ReadPlataDecr** выполнить следующий код:

```
ASYNC_PAR pp;
pp.s_Type = L_ASYNC_TTL_CFG;
pp.Mode = 1;
pI->IoAsync(&pp);

pp.s_Type = L_ASYNC_TTL_OUT;
pp.Data[0] = 0xA525; // в Data[0] данные для цифровых линий
pI->IoAsync(&pp);
```

  При этом на цифровые линии будет выведено 0xA525.

- Завершить работу с библиотекой так как описано в главе ***«Подключение и работа с библиотекой (на CPP)»*** раздела ***«Описание API DLL библиотеки»***

Более подробно о возможностях функции **IoAsync** можно прочитать в ее описании.

... прочитать данные с TTL входов

Для этого нужно выполнить шаги приведенные ниже.

- Подключить библиотеку LComp так как описано в главе ***«Подключение и работа с библиотекой (на CPP)»*** раздела ***«Описание API DLL библиотеки»***
- После вызова **OpenLDevice, LoadBios** и **ReadPlataDecr** выполнить следующий код:

```
ASYNC_PAR pp;
pp.s_Type = L_ASYNC_TTL_INP;
pI->IoAsync(&pp);
cout << (USHORT)pp.Data[0] << endl; // в Data[0] данные с цифровых линий
```

  При этом на экран будет выведено состояние TTL входов.

- Завершить работу с библиотекой так как описано в главе ***«Подключение и работа с библиотекой (на CPP)»*** раздела ***«Описание API DLL библиотеки»***

Более подробно о возможностях функции **IoAsync** можно прочитать в ее описании.

... вывести одиночный отсчет на ЦАП

Для этого нужно выполнить шаги приведенные ниже.

- Подключить библиотеку LComp так как описано в главе ***«Подключение и работа с библиотекой (на CPP)»*** раздела ***«Описание API DLL библиотеки»***
- После вызова **OpenLDevice, LoadBios** и **ReadPlataDecr** выполнить следующий код:

```
// Для плат L780 L761 L783 E440 E140 E154 E2010 L1450
ASYNC_PAR pp;
pp.s_Type = L_ASYNC_DAC_OUT;
pp.Mode = 0; // 0 канал ЦАП; если 1, то 1 канал
pp.Data[0] = 0x00FF; // код для ЦАП
pI->IoAsync(&pp);
```

  При этом на ЦАП будет выведен один отсчет.

- Завершить работу с библиотекой так как описано в главе ***«Подключение и работа с библиотекой (на CPP)»*** раздела ***«Описание API DLL библиотеки»***

Более подробно о возможностях функции **IoAsync** можно прочитать в ее описании. В частности для платы L791 структуру надо заполнить несколько иначе. Как — см. описание **IoAsync**.

... получить поток данных с АЦП

Поток данных с АЦП программируется более сложно чем одиночные асинхронные операции. Начало и завершение работы аналогично тому что нужно для чтения одиночного отсчета. Подробно и с комментариями как программировать потоковый ввод данных с АЦП можно посмотреть в примере L7XX.TST, который находится после установки библиотеки Lcomp в \L-Card\Library\L7XX.TST.

... вывести поток данных на ЦАП

Некоторые платы и модули поддерживают потоковый вывод данных на ЦАП. Программируется это более сложно чем одиночные асинхронные операции. Начало и завершение работы аналогично тому что нужно для чтения одиночного отсчета. Подробно и с комментариями как программировать потоковый вывод данных на ЦАП можно посмотреть в примере L7XX.OSC, который находится после установки библиотеки Lcomp в \L-Card\Library\L7XX.OSC. Для платы L-791 есть отдельный пример L791.GNR - пример простого генератора.

Справочные данные по платам и модулям

Введение

В данном разделе собраны замечания и пояснения для разным плат и модулей.

Адресное пр-во и команды биос L-761/L-780/L-783

Существует две ревизии PCI плат - A и B. Они различаются адресацией портов. Ревизия A очень редкая. Наиболее распространенная ревизия – B. 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+4096	-	Порт для установки адреса IDMA.
Base+8192	-	Порт генерации IRQ2 DSP.
Base+12288	-	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+4096	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_KADR_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_LENGHT_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 PC 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 PC
0x8D5A	L_IRQ_FIFO_ADDRESS_PLX	Если произошло прерывание в PC, 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_POROG_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).

Замечания для платы L-791

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:

- Воод и вывод данных реализован в комбинированном режиме.В небольшой буфер в драйвере данные поступаютпо BusMaster (или забираются), а буфер в PC аналогичен буферу всех остальных плат/модулей и туда данные перекладываются по прерываниям. От старого режима с одним только BusMaster и фиксированными буферами было решено отказаться тк плата поддерживает только 32 бит DMA и не может адресоваться к памяти выше 4G, что сейчас стало повсеместно.Буфера в PC теперь могут быть произвольными (Pages*IrqStep)
- Соответсвенно счетчики sync стали программными и маппинга регистров платы тоже нет.
- про регистры платы читайте печатную книжку;
- про логические номера каналов читайте печатную книжку;
- размер FIFO буфера АЦП можно задавать 1,2,4,8,16,32,64,128 отсчетов;
- Размер IrqStep ограничен 2048 отсчетами (тк буфер Busmaster 4*4096 байт)

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 кОтсчетов.
- 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 1M Отсчетов.
- Коррекция данных выполняется пользователем.

Замечания для плат L-761/L-780/L-783

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.

Замечания для модуля E14-440/E14-140M

Пояснения по работе с модулем E14-440/E14-140M.

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

- у модуля E14-140M 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 кОтсчетов.
- Коррекция данных модуля E14-140M выполняется пользователем.
- Начиная с этой версии драйверов и библиотек модули поддерживают потоковый вывод на ЦАП.
- Данные для ЦАП модуля 440/140 надо задавать аналогично данным для потокового вывода PCI плат, только массив данных 16-битный. Для модуля 140M данные 16 битные и всегда выводятся на 2 ЦАПа по очереди. - 1 2 1 2 1 2. Для 440 модуля номер ЦАП задается в коде данных аналогично PCI платам. IrqStep для модуля E140M следует всегда задавать2048 отсчетов. Для E440 IrqStep должен быть равен FIFO и ограничен соответственно максимальным размером FIFO модуля (те максимальным значением для половины циклического буфера в плате), у E140M это параметр просто не настраивается поэтому всегда 2048.
- пример по работе с ЦАП см L7XX.OSC.

Оглавление