libxidcusb

1.4.1

Создано системой Doxygen 1.7.1

Sun Dec 26 2010 19:58:22

Оглавление

Глава 1

XIDC USB

Глава 2

Введение

2.1 О библиотеке

Спасибо, что вы выбрали мультиплатформенную библиотеку XIDCUSB! Этот документ содержит всю необходимую информацию о библиотеке XIDCUSB. Она использует распространенный и проверенный интерфейс виртуального последовательного порта от компании FTDI, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми под ОС, в том числе Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Windows ME, Windows 98, Linux, Mac OS X, т.е. везде, где может быть установлен свободно распространяемый драйвер виртуального последовательного порта от компании FTDI. Библиотека XIDCUSB может работать с 127 модулями управления на одном компьютере. Она поддерживает подключение и отключение устройств "на лету". Каждый запущенный экземпляр управляющей программы может работать только с одним устройством. Множественный доступ управляющих программ к одному и тому же устройству не допускается.

2.2 Требования к установленному программному обеспечению

2.2.1 Для сборки библиотеки

На ОС Windows:

- Windows 2000 или более новая, 64-х битная система для сборки для 32/64 архитектур, 32-х битная система для только 32-х битных архитектур.
- Microsoft Visual C++ 2005 или более новая

На ОС UNIX:

- 64- или 32-х битная система
- gcc 4 или более новый
- стандартные инструменты: autoconf, autoheader, automake, autoreconf, libtool
- пакет doxygen

На ОС Mac OS X:

- gcc 4 или более новый
- XCode 3

2.2.2 Для использования библиотеки

На ОС Windows:

- Windows 2000 или более новая, 64-х битная система для работы с 32/64 архитектурами, 32-х битная система для только 32-х битных архитектур.
- Microsoft Visual C++ 2005 или более новый (дополнительно)
- mingw (дополнительно)

На ОС UNIX:

- 64- или 32-х битная система
- gcc 4 или более новый
- ОС на базе Debian или Red Hat на ядре linux 2.6 с возможностью устанавливать DEB- или RPM-пакеты
- инструмент make

На ОС Mac OS X:

- gcc 4 или более новый
- XCode 3

2.3 История изменений

2.3.1 version 1.4.1

sun, 26 dec 2010 02:00:00 +0400

- Новая версия
- feature #425: linux version doesn't query ttyACM ttys

2.3.2 version 1.4.0

wed, 24 nov 2010 02:00:00 +0400

- Новая версия
- feature #358: поддержка протокола v12

2.3.3 version 1.3.0

sat, 16 oct 2010 02:00:00 +0400

- Новая версия
- feature #94: поддержка Visual Basic

2.3.4 version 1.2.0

wed, 29 sep 2010 06:00:00 +0400

- Новая версия
- bug #286: зависвание при одновременной работе с двумя устройствами

2.3.5 version 1.1.1r

thu, 22 jul 2010 15:00:00 +0400

- feature #294: Добавлена документация на русском языке

2.3.6 version 1.1.1

fri, 02 apr 2010 15:00:00 +0400

- Новая версия
- bug #267: исправлена ошибка в обновлении прошивки
- feature #264: опубликован Development Kit

2.3.7 version 1.1.0

fri, 19 feb 2010 15:00:00 +0400

- Новая версия
- bug #34: Добавлена возможность работы с реальными последовательными портами
- bug #258: uint8_t, uint32_t
- feature #125: Изменен способ обнаружения устройств
- feature #226: Изменен порядок обнаружения устройств под Linux
- feature #237: Обновлена документация
- feature #261: Изменены форматы команд S005, G005

2.3.8 version 1.0.0

sun, 12 dec 2009 21:00:00 +0400

- Новая версия
- bug #215: Исправлена проверка CRC при ответах
- feature #91: Добавлено описание и примеры по использованию библиотеки с C#
- feature #92: Добавлено описание и примеры по использованию библиотеки с C++ Builder
- feature #93: Добавлено описание и примеры по использованию библиотеки с Delphi
- feature #173: Введена обработка новой команды G1

2.3.9 version 0.9.0

sat, 15 aug 2009 21:00:00 +0400

- Новая версия
- Bug #24: Исправлена ошибка при попытке работы с одним устройством из нескольких программ (возможность заблокирована)
- Bug #56: Исправлена ошибка с обработкой данный в структуре cdc_pwmlevel
- Bug #76: Исправлена орфографическая ошибка в settings_t::flags
- Feature #27: Улучшена производительность
- Feature #98: Обновлена документация
- Feature #99: Обновлены примеры программ

2.3.10 version 0.8.0

thu, 23 jul 2009 18:00:00 +0400

- Новая версия
- bug #11: Добавлена поддержка сборки dll с def-файлами
- bug #41: Исправлено поведение ttl_setup_outset
- bug #42: Исправлено поведение engine_limit_curr, engine_limit_volt
- bug #54: Исправлена проблема в open_device() при повторном вызове
- feature #21: Введена поддержка mingw

2.3.11 version 0.7.0

wed, 08 jul 2009 06:00:00 +0400

- Новая версия
- feature #39: Добавлена функция обновления прошивки
- bug #33: Добавлена работа с флагами TTL_SYNCIN_SINGLE_SHIFT и TTL_SYNCIN_MULTI_SHIFT
- bug #35: Исправлена запись ключа командой write_key
- bug #36: Исправлена работа функции get_analog_data()

2.3.12 version 0.6.0

tue, 30 jun 2009 17:00:00 +0400

- Новая версия
- bug #12: Исправлена ошибка в pid_settings_t: поля kprpm, kiprm, kdprm
- bug #13: Исправлена ошибка в set_pwm_freq()
- bug #14: Исправлена ошибка возникающая при инициализации
- bug #16: Исправлена ошибка с вводом параметров синхронизации
- bug #17: Исправлена ошибка с полем param ttl_setup_front
- bug #25: Исправлена ошибка чтения калибровочных коэффициентов
- feature #32: Добавлены команды 1234, calb, reset
- Добавлена запись ключа

2.3.13 Version 0.5.0

Fri, 21 May 2009 01:12:54 +0400

- Новая версия
- Обновлена документация
- Исправлена проблема при работе с памятью
- Улучшена совместимость с C++ под Windows

2.3.14 Version 0.4.0

Fri, 06 May 2009 11:37:29 +0400

- Новая версия
- Добавлена документация

2.3.15 Version 0.3.0

Fri, 24 Apr 2009 11:17:29 +0400

- Новая версия
- Библиотека переименована
- Исправлена проблема со сборкой под unix
- Улучшена работа с семафорами

2.3.16 Version 0.2.0

Sun, 05 Apr 2009 19:02:43 +0400

- Новая версия
- Добавлена сборка deb и rpm пакетов
- Библиотека под Mac OS X улучшена
- Добавлен номер версии под win32

2.3.17 Version 0.1.0

Thu, 02 Apr 2009 00:02:44 +0400

- Первая версия. Базовая функциональность

Глава 3

Как пересобрать библиотеку

3.1 Сборка в ОС UNIX

Выполните:

```
$ ./autogen.sh
$ ./configure
```

Для 32-битных систем:

```
$ make
```

Для 64-битных систем:

```
$ CFLAGS="-m64" make
```

Установите в /usr/local:

```
$ sudo make install
```

Сборка документации:

```
$ make doxygen-doc
```

Скомпилированная библиотека - /usr/local/lib/libxidcusb.so, заголовочный файл - /usr/local/include/xidcusb.h

3.1.1 Сборка deb пакета

Требования к ПО: 64-битная debian система, gcc, autotools, dpkg-dev.

В первую очередь увеличьте номер версии:

```
$ ./version.sh 0.42.0
```

Запустите скрипт:

```
$ ./builddeb.sh 0.42.0
```

3.1.2 Сборка rpm пакета

Требования к ПО: 64-битная ОС с поддержкой RPM (Fedora, Red Hat, SUSE), gcc, autotools.

В первую очередь увеличьте номер версии:

```
$ ./version.sh 0.42.0
```

Запустите скрипт:

```
$ ./buildrpm.sh 0.42.0
```

3.2 Сборка в ОС Darwin

Выполните:

```
$ ./autogen.sh
$ ./configure
```

Для 32-битных систем:

```
$ make
```

Для 64-битных систем:

```
$ make CFLAGS="-arch x86_64" LDFLAGS="-arch x86_64"
```

Установите в /usr/local:

```
$ sudo make install
```

Скомпилированная библиотека - /usr/local/lib/libxidcusb.dylib, заголовочных файл - /usr/local/include/xidcusb.h

Замечание: только 32- и 64-битные библиотеки могут быть установлены одновременно.

3.3 Сборка фреймворка в ОС Mac OS X

Выполните:

```
$ xcodebuild
```

или откройте libxidusb.xcodeproj и соберите его в XCode 3.x

Собранный фреймворк будет расположен в 'dist/<version>/macosx'

3.4 Сборка в ОС Windows

Убедитесь, что номер версии уже увеличен скриптом version.sh.

Откройте проект libxidcusb.sln в Visual Studio 2005/2008 и соберите его.

Скомпилированная библиотека libxidcusb.dll будет находиться в 'Release/Win32' или 'Release/x64'

Так же можно запустить скрипт build.bat из командной строки Visual Studio. Будут собраны 32- и 64-битные версии библиотеки и расположены в 'dist/<version>/win32' or 'dist/<version>/x64'.

3.5 Доступ к исходным кодам

Исходные коды XIDCUSB могут выданы по отдельному запросу.

Глава 4

Как использовать с...

Для приобретения первых навыков использования библиотеки создано простое тестовое приложение testapp. Языки, отличные от C-подобных, поддерживаются с помощью вызовов с преобразованием аргументов типа stdcall. Простое тестовое приложение на языке C расположено в директории 'testapp', проект на C# - в 'testcs', на VB.NET - в 'testvbnet', для delphi 6 - в 'testdelphi'. Библиотеки, заголовочные файлы и другие необходимые файлы расположены в директориях 'win32'/'win64','macosx' и подобных.

4.1 Visual C++

Тестовое приложение может быть собрано с помощью testapp.sln. В настройках линкера необходимо указать libxidcusb.lib. Для компиляции необходимо использовать также MS Visual C++, mingw-library не поддерживается. Убедитесь, что Microsoft Visual C++ Redistributable Package установлен.

ЗАМЕЧАНИЕ: Пример собран с MS Visual C++ 2008 SP1 и требует пакет 9.0.307291 (поставляется с SDK, файлы vcredist_x86 или vcredist_x64).

Шаги, необходимые для сборки тестового приложения:

- Соберите testapp.sln для вашей архитектуры
- Скопируйте библиотеку libxidcusb.dll в директорию 'Release'
- Запустите testapp.exe в консоле

4.2 MinGW

MinGW это вариант GCC для платформы win32. Требует установки пакета MinGW.

testapp, скомпилированный с помощью MinGW, может быть собран с MS Visual C++ или библиотеками mingw:

```
$ mingw32-make -f Makefile.mingw all
```

Далее скопируйте libxidcusb.dll в текущую директорию и запустите testapp.exe.

4.3 C++ Builder

В первую очередь вы должны создать подходящую для C++ Builder библиотеку. Библиотеки Visual C++ и Builder не совместимы. Выполните:

```
$ implib libxidcusb.lib libxidcusb.lib
```

Затем скомпилируйте тестовое приложение:

```
$ bcc32 -I..\libxidcusb\include -DWIN32 -DNDEBUG -D_WINDOWS testapp.c libxidcusb
    .lib
```

4.4 .NET

Для использования в .NET предлагается "обертка" xidcusbnet.dll. Она распространяется в двух различных архитектурах и зависит от .NET 2.0.

Тестовые приложения на языке C# для Visual Studio 2008 расположены в директоиях testcs (для C#) и testvbnet (для VB.NET). Просто скомпилируйте их, положите libxidcusb.dll рядом со сборкой и запустив программу.

4.5 Delphi

"Обертка" для libxidcusb.dll предлагается как модуль include/xidcusb.pas

Консольное тестовое приложение размещено в директории 'testdelphi'. Проверено с Delphi 6 на 32-битной системе.

Просто скомпилируйте, разместите DLL в директории с исполняемым модулем и запустите его.

4.6 XCode

Testapp должна быть собрана с XCode проект testapp.xcodeproj. Используйте конфигурацию Release. Библиотека поставляется в формате MAC OS X framework, в той же директории находится собранное тестовое приложение Testapp.app.

Запустите приложение Testapp.app проверьте его работу в Console.app.

4.7 GCC

Убедитесь, что libxidcusb (RPM или DEB пакеты) установлены на вашей системе. Пакеты должны устанавливаться с помощью package manager'а вашей ОС.

Testapp может быть собрано следующим образом:

```
$ make all
```

Затем запустите приложение с помощью:

```
$ ./testapp
```

Глава 5

Интерфейсы программирования

Reference: xidcusb.h

Глава 6

Алфавитный указатель структур данных

6.1 Структуры данных

Структуры данных с их кратким описанием.

Глава 7

Список файлов

7.1 Файлы

Полный список документированных файлов.

Глава 8

Структуры данных

8.1 Структура analog_data_t

Аналоговые данные.

#include <xidcusb.h>

Поля данных

- int adc_va1
  "Выходное напряжение на 1 выводе обмотки A" необработанные данные с АЦП
- int adc_va2
  "Выходное напряжение на 2 выводе обмотки A" необработанные данные с АЦП
- int adc_vb1
  "Выходное напряжение на 1 выводе обмотки B" необработанные данные с АЦП
- int adc_vext
  "Напряжение питания" необработанные данные с АЦП
- int adc_vsup
  "Напряжение питания ключей H-моста" необработанные данные с АЦП
- int adc_acur
  "Ток через обмотку A" необработанные данные с АЦП
- int adc_bcur
  "Ток через обмотку B" необработанные данные с АЦП
- int adc_fullcur
  "Полный ток" необработанные данные с АЦП
- int adc_pot
  "Потенциометр" необработанные данные с АЦП

- int adc_tempkey

  Напряжение с датчика температуры, необработанные данные с АЦП.

- int cdc_va1

  "Выходное напряжение на 1 выводе обмотки A" откалиброванные данные

- int cdc_va2

  "Выходное напряжение на 2 выводе обмотки A" откалиброванные данные

- int cdc_vb1

  "Выходное напряжение на 1 выводе обмотки B" откалиброванные данные

- int cdc_vext

  "Напряжение питания" откалиброванные данные

- int cdc_vsup

  "Напряжение питания ключей H-моста" откалиброванные данные

- int cdc_acur

  "Ток через обмотку A" откалиброванные данные

- int cdc_bcur

  "Ток через обмотку B" откалиброванные данные

- int cdc_fullcur

  "Полный ток" откалиброванные данные

- int cdc_tempkey

  Температура, откалиброванные данные.

- int cdc_dutycycle

  Коэффициент заполнения ШИМ.

8.1.1 Подробное описание

Аналоговые данные. Эта структура содержит необработанные данные с АЦП и нормированные значения. Эти данные используются в сервисных целях для тестирования и калибровки устройства.

См. также

get_analog_data

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.2 Структура calibration_settings_t

Калибровочные коэффициенты.

#include <xidcusb.h>

Поля данных

- int amp_vout1
  Коэффициент усиления для "Выходного напряжения 1".
- int amp_vout2
  Коэффициент усиления для "Выходного напряжения 2".
- int amp_vext
  Коэффициент усиления для "Напряжения внешнего питания".
- int amp_vkey
  Коэффициент уисления для "Напряжения питания ключей H-моста".
- int amp_engcur
  Коэффициент усиления для "Тока через мотор".
- int amp_tempkey
  Коэффициент усиления для "Температуры MOSFET ключей".
- int off_vout1
  Коэффициент смещения для "Выходного напряжения 1".
- int off_vout2
  Коэффициент смещения для "Выходного напряжения 2".
- int off_vext
  Коэффициент смещения для "Напряжения внешнего питания".
- int off_vkey
  Коэффициент смещения для "Напряжения питания ключей H-моста".
- int off_engcur
  Коэффициент смещения для "Тока через мотор".
- int off_tempkey
  Коэффициент смещения для "Температуры MOSFET ключей".

8.2.1 Подробное описание

Калибровочные коэффициенты. Эта структура содержит коэффициенты, которые используются для преобразования данных с АЦП в используемые единицы измерения. Диапазон: -32767..32767. В контроллере эти коэффициенты хранятся в энергонезависимой flash памяти.

Все устройства при изготовлении проходят процедуру калибровки и при обычной работе их не требуется изменять. Если вы хотите произвести калибровку устройства, проконсультируйтесь с производителем. Неправильные коэффициенты могут быть причиной перегрева и выхода контроллера из строя.

См. также

get_calibration_coeffs
set_calibration_coeffs

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.3 Структура chart_data_t

Дополнительное состояние устройства.

#include <xidcusb.h>

Поля данных

- int winding_voltage_a

  Напряжение на обмотке A.

- int winding_voltage_b

  Напряжение на обмотке B.

- int winding_voltage_c

  Напряжение на обмотке C.

- int winding_current_a

  Ток через обмотке A.

- int winding_current_b

  Ток через обмотке B.

- int winding_current_c

  Ток через обмотке C.

- int current_temp

  Температура MOSFET ключей.

8.3.1 Подробное описание

Дополнительное состояние устройства. Эта структура содержит основные дополнительные параметры текущего состоянии контроллера, такие напряжения и токи обмоток и температуру.

См. также

get_status

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.4 Структура engine_settings_t

Настройки мотора.

#include <xidcusb.h>

Поля данных

- unsigned int nom_voltage

  Номинальное напряжение мотора.

- unsigned int nom_input

  Номинальный ток через мотор.

- unsigned int nom_rpm

  Номинальная скорость мотора.

- unsigned int flags

  Флаги, управляющие работой мотора.

- int antiplay

  Количество импульсов энкодера, на которое будет проходить позиционер заданную позицию для подхода к ней с одной и той же стороны.

- int microstep_mode

  Настройки микрошагового режима.

- int type

  Тип мотора.

8.4.1 Подробное описание

Настройки мотора. Эта структура содержит настройки мотора. Настройки определяют номинальные значения напряжения, тока, скорости мотора, характер движения и тип мотора. Пожалуйста, загружайте новые настройки когда вы меняете мотор, энкодер или позиционер. Помните, что неправильные настройки мотора могут повредить оборудование.

См. также

set_engine_settings
get_engine_settings

8.4.2 Поля

8.4.2.1 int antiplay

Количество импульсов энкодера, на которое будет проходить позиционер заданную позицию для подхода к ней с одной и той же стороны.

Используется, если установлен флаг ENGINE_ANTIPLAY. Диапазон: 1..65535

8.4.2.2 unsigned int nom_input

Номинальный ток через мотор.

Контроллер будет сохранять ток через мотор не выше номинального, если установлен флаг ENGINE_LIMIT_CURR. Диапазон: 1..65535

8.4.2.3 unsigned int nom_rpm

Номинальная скорость мотора.

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE_LIMIT_RPM.

8.4.2.4 unsigned int nom_voltage

Номинальное напряжение мотора.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT. Диапазон: 1..65535

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.5 Структура home_settings_t

Настройки калибровки позиции.

#include <xidcusb.h>

Поля данных

- int fast_home

  Скорость быстрого движения.

- int slow_home

  Скорость медленного движения.

- unsigned int delta

  Расстояние отхода от точки останова.

- unsigned int flags

Набор флагов, определяющие такие параметры, как направление и условия останова.

8.5.1 Подробное описание

Настройки калибровки позиции. Эта структура содержит настройки, использующиеся при калибровке позиции.

См. также

get_home_settings
set_home_settings
command_home

8.5.2 Поля

8.5.2.1 unsigned int delta

Расстояние отхода от точки останова.

Диапазон: 0..2147483647.

8.5.2.2 int fast_home

Скорость быстрого движения.

Диапазон: 1..32767.

8.5.2.3 int slow_home

Скорость медленного движения.

Диапазон: 1..65535.

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.6 Структура pid_settings_t

Настройки ПИД.

#include <xidcusb.h>

Поля данных

- unsigned int kpi

  Пропорциональный коэффициент ПИД контура по току.

- unsigned int kii

  Интегральный коэффициент ПИД контура по току.

- unsigned int kdi

  Дифференциальный коэффициент ПИД контура по току.

- unsigned int kpu

  Пропорциональный коэффициент ПИД контура по напряжению.

- unsigned int kiu

  Интегральный коэффициент ПИД контура по напряжению.

- unsigned int kdu

  Диференциальный коэффициент ПИД контура по напряжению.

- unsigned int kprpm

  Пропорциональный коэффициент ПИД контура по скорости.

- unsigned int kirpm

  Интегральный коэффициент ПИД контура по скорости.

- unsigned int kdrpm

  Дифференциальный коэффициент ПИД контура по скорости.

- unsigned int kppos

  Пропорциональный коэффициент ПИД контура по позиции.

- unsigned int kipos

  Интегральный коэффициент ПИД контура по позиции.

- unsigned int kdpos

  Дифференциальный коэффициент ПИД контура по позиции.

- unsigned int kpe1

  Пропорциональный коэффициент ПИД доп1.

- unsigned int kie1

  Интегральный коэффициент ПИД доп1.

- unsigned int kde1

  Дифференциальный коэффициент ПИД доп1.

- unsigned int kpe2

  Пропорциональный коэффициент ПИД доп2.

- unsigned int kie2

  Интегральный коэффициент ПИД доп2.

- unsigned int kde2

  Дифференциальный коэффициент ПИД доп2.

- unsigned int kpe3

Пропорциональный коэффициент ПИД доп3.

- unsigned int kie3

  Интегральный коэффициент ПИД доп3.

- unsigned int kde3

  Дифференциальный коэффициент ПИД доп3.

8.6.1 Подробное описание

Настройки ПИД. Эта структура содержит коэффициенты для ПИД регуляторов. Диапазон: 0..65535. Они определяют работу ПИД контуров напряжения, тока, скорости и позиции. Эти коэффициенты хранятся во flash памяти памяти контроллера. Пожалуйста, загружайте новые настройки, когда вы меняете мотор или позиционер. Помните, что неправильные настройки ПИД контуров могут повредить оборудование.

См. также

set_pid_settings
get_pid_settings

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.7 Структура state_t

Состояние устройства.

#include <xidcusb.h>

Поля данных

- DevState device_state

  Состояние устройства.

- PwmState pwm_state

  Состояние ШИМ.

- unsigned int flags

  Флаги состояния.

- unsigned int current_position

  Текущая позиция.

- int current_rpm

  Текущая скорость, RPM.

- int current_input

Ток через мотор.

- int current_engine_voltage

  Текущее усредненное напряжение на моторе.

- int current_power_voltage

  Напряжение питания.

- int current_temp

  Температура MOSFET ключей.

8.7.1 Подробное описание

Состояние устройства. Эта структура содержит основные параметры текущего состоянии контроллера такие как скорость, позиция и флаги состояния.

См. также

get_status

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

8.8 Структура sync_settings_t

Настройки синхронизации.

#include <xidcusb.h>

Поля данных

- unsigned int setup_flags

  Основные флаги синхронизации.

- unsigned int syncin_flags

  Флаги синхронизации входа.

- unsigned int syncout_flags

  Флаги синхронизации выхода.

- SyncPinMode sync_pin_mode

  Тип вывода синхронизации.

- unsigned int impulse_time

  Длительность импульса синхронизации для выхода или задержка защиты от дребезга для входа.

- unsigned int impulse_period

Период генерации импульсов, используется при установленном флаге TTL_SYNCOUT_-ONPERIOD.

8.8.1 Подробное описание

Настройки синхронизации. Эта структура содержит все настройки, определяющие поведение входа\выхода синхронизации.

См. также

get_sync_settings
set_sync_settings

8.8.2 Поля

8.8.2.1 unsigned int impulse_period

Период генерации импульсов, используется при установленном флаге TTL_SYNCOUT_-ONPERIOD.

Диапазон: 0..65535

8.8.2.2 unsigned int impulse_time

Длительность импульса синхронизации для выхода или задержка защиты от дребезга для входа.

Диапазон: 0..65535

Объявления и описания членов структуры находятся в файле:

- include/xidcusb.h

Глава 9

Файлы

9.1 Файл include/xidcusb.h

Header file for XIDCUSB library.

Структуры данных

- struct state_t
  Состояние устройства.
- struct chart_data_t
  Дополнительное состояние устройства.
- struct calibration_settings_t
  Калибровочные коэффициенты.
- struct engine_settings_t
  Настройки мотора.
- struct pid_settings_t
  Настройки ПИД.
- struct sync_settings_t
  Настройки синхронизации.
- struct home_settings_t
  Настройки калибровки позиции.
- struct analog_data_t
  Аналоговые данные.

Макросы

- #define DCCONTROL_API __attribute__((visibility("default")))

Library import macros.

- #define device_undefined -1
  Макрос, означающий неопределенное устройство.

Результаты выполнения команд

- #define result_ok 0
  выполнено успешно.
- #define result_error -1
  общая ошибка.
- #define result_not_implemented -2
  функция не определена.
- #define result_dataerror -3
  ошибка в запросе или в ответе.
- #define result_nodevice -4
  устройство не подключено.

Флаги состояния

Содержат бинарные значения состояния контроллера.

Могут быть объединены с помощью логического ИЛИ.

См. также

state_t::flags
get_status

- #define STATE_RIGHT_EDGE 0x0001
  Достижение правой границы.
- #define STATE_LEFT_EDGE 0x0002
  Достижение левой границы.
- #define STATE_EXT_POWER 0x0004
  Используется внешнее питание.
- #define STATE_ERRC 0x0008
  Недопустимая команда.
- #define STATE_ERRD 0x0010
  Недопустимые данные.
- #define STATE_POWER_DISABLE 0x0020
  Получено сообщение от FTDI об отключении USB-питания.
- #define STATE_TTLIO_PINOUT 0x0040
  Если флаг установлен, вывод синхронизации работает как выход; если флаг сброшен, вывод работает как вход.

- #define STATE_TTLIO_LEVEL 0x0080
  Лог.
- #define STATE_BUTTON_RIGHT 0x0100
  Состояние кнопки "вправо" (1, если нажата).
- #define STATE_BUTTON_LEFT 0x0200
  Состояние кнопки "влево" (1, если нажата).
- #define STATE_OVERHEAT 0x2000
  Температура превысила допустимую вличину.
- #define STATE_OVERLOAD_VOLTAGE 0x4000
  Перегрузка по напряжению.
- #define STATE_OVERLOAD_CURRENT 0x8000
  Перегрузка по току.

Флаги параметров мотора

Определяют настройки движения и работу ограничителей.

Возращаются командой get_engine_settings. Могут быть объединены с помощью логического ИЛИ.

См. также

engine_settings_t::flags
set_engine_settings
get_engine_settings

- #define ENGINE_REVERSE 0x0001
  Флаг реверса.
- #define ENGINE_FINISHING 0x0002
  Флаг точной доводки.
- #define ENGINE_MAX_SPEED 0x0004
  Флаг максимальной скорости.
- #define ENGINE_ANTIPLAY 0x0008
  Компенсация люфта.
- #define ENGINE_ACCEL_ON 0x0010
  Ускорение.
- #define ENGINE_LIMIT_VOLT 0x0020
  Номинальное напряжение мотора.
- #define ENGINE_LIMIT_CURR 0x0040
  Номинальный ток мотора.
- #define ENGINE_LIMIT_RPM 0x0080
  Номинальная частота вращения мотора.

- #define ENGINE_HOLD 0x0100
  Режим удержания.
- #define ENGINE_DYNAMIC_HOLD 0x0200
  Динамическое удержание.
- #define ENGINE_ONLY_FINISHING 0x0400
  Если флаг установлен, движение к точке начинается сразу в состоянии Tune.

Флаги параметров микрошагового режима

Определяют деление шага в микрошаговом режиме.

Используются с шаговыми моторами. Возращаются командой get_engine_settings. Могут быть объединены с помощью логического ИЛИ.

См. также

engine_settings_t::flags
set_engine_settings
get_engine_settings

- #define MICROSTEP_MODE_FULL 0x01
  Полношаговый режим.
- #define MICROSTEP_MODE_FRAC_2 0x02
  Деление шага 1/2.
- #define MICROSTEP_MODE_FRAC_4 0x04
  Деление шага 1/4.
- #define MICROSTEP_MODE_FRAC_8 0x08
  Деление шага 1/8.
- #define MICROSTEP_MODE_FRAC_16 0x10
  Деление шага 1/16.
- #define MICROSTEP_MODE_FRAC_32 0x20
  Деление шага 1/32.
- #define MICROSTEP_MODE_FRAC_64 0x40
  Деление шага 1/64.
- #define MICROSTEP_MODE_FRAC_128 0x80
  Деление шага 1/128.

Флаги, определяющие тип мотора

Определяют тип мотора.

Возращаются командой get_engine_settings.

См. также

engine_settings_t::flags
set_engine_settings
get_engine_settings

- #define ENGINE_TYPE_DC 0x01
  Мотор постоянного тока.
- #define ENGINE_TYPE_STEP 0x02
  Шаговый мотор.
- #define ENGINE_TYPE_BRUSHLESS 0x04
  Безщеточный мотор.

- #define FEEDBACK_POTENTIOMETER 0x00
  Тип обратной связи.
- #define FEEDBACK_ENCODER 0x01
  обратная связь с помощью энкодера.
- #define FEEDBACK_ENCODERDIFF 0x02
  обратная связь с помощью энкодера с дифференциальным входом.
- #define FEEDBACK_ENCODERHALL 0x04
  обратная связь с помощью датчика Холла.
- #define FEEDBACK_EMF 0x08
  обратная связь по ЭДС.
- #define FEEDBACK_NONE 0x10
  обратная связь отсутствует.

Флаги настроек синхронизации входа/выхода

См. также

sync_settings_t::setup_flags
get_sync_settings
set_sync_settings

- #define TTL_SETUP_DIROUT 0x0001
  Если флаг установлен, вывод синхронизации работает как выход; если флаг сброшен, вывод синхронизации работает как вход.
- #define TTL_SETUP_OUTSET 0x0002
  Флаг определяет логический уровень на выводе синхронизации, установленном как выход.
- #define TTL_SETUP_FRONT 0x0010
  Настройка синхронизации в режиме входа.

Флаги настроек синхронизации входа

См. также

sync_settings_t::syncin_flags
get_sync_settings
set_sync_settings

- #define TTL_SYNCIN_ONMOVE 0x0001

  Бесконечное вращение в направлении TTL_SYNCIN_DIRRIGHT, если флаг установлен; иначе - смещение на заданную позицию.

- #define TTL_SYNCIN_MULTI_SHIFT 0x0004

  Если флаг установлен, происходит многократное срабатывание по команде SHIFT; если сброшен - однократное срабатывание по команде MOVE.

- #define TTL_SYNCIN_DIRRIGHT 0x0008

  Направление вращения при синхронизации.

Флаги настроек синхронизации выхода

См. также

sync_settings_t::syncout_flags
get_sync_settings
set_sync_settings

- #define TTL_SYNCOUT_ONSTART 0x0001

  Генерация синхронизирующего импульса при начале движения.

- #define TTL_SYNCOUT_ONSTOP 0x0002

  Генерация синхронизирующего импульса при остановке.

- #define TTL_SYNCOUT_ONPERIOD 0x0004

  Генерация синхронизирующего импульса при каждом импульсе энкодера.

Флаги границ

Типы границ и поведение позиционера на границах.

Могут быть объединены с помощью побитового ИЛИ.

См. также

get_edges_settings
set_edges_settings

- #define BORDER_IS_ENCODER 0x0001

  Если флаг установлен, границы определяются предустановленными точками на шкале позиции.

- #define BORDER_STOP_LEFT 0x0002

  Если флаг установлен, мотор останавливается при достижении левой границы.

- #define BORDER_STOP_RIGHT 0x0004

  Если флаг установлен, мотор останавливается при достижении правой границы.

Флаги концевых выключателей

Определяют направление и состояние границ.

Могут быть объединены с помощью побитового ИЛИ.

См. также

get_edges_settings
set_edges_settings

- #define ENDER_DIRECT 0x0001

  Если флаг установлен, первый концевой выключатель находится справа; иначе - слева.

- #define ENDER_ON1 0x0002

  Если флаг установлен, первый концевой выключатель является нормально замкнутым; иначе - нормально разомкнутым.

- #define ENDER_ON2 0x0004

  Если флаг установлен, второй концевой выключатель является нормально замкнутым; иначе - нормально разомкнутым.

Флаги настроек команды home

Определяют поведение для команды home Могут быть объединены с помощью побитового ИЛИ.

См. также

get_home_settings
set_home_settings
command_home

- #define HOME_DIR_FAST 0x01

  Определяет направление первоначального движения мотора после поступления команды HOME.

- #define HOME_STOP_FAST 0x02

  Если флаг установлен, то первоначальное движение завершается по сигналу со входа синхронизации; иначе - по сигналу с концевого выключателя.

- #define HOME_DIR_SLOW 0x04

  Определяет направление второго движения мотора.

- #define HOME_REV_EN 0x08

  Если флаг установлен, используется Revolution sensor; иначе - этап пропускается.

Определения типов

- typedef int device_t

  Тип идентификатора устройства.

- typedef int result_t

  Тип, определяющий результат выполнения команды.

Перечисления

- enum DevState {
  DevStateOff, DevStateStop, DevStateMoving, DevStateTune,
  DevStateHoming, DevStateCalibr = 0x0C, DevStateAlarm = 0x0A }
  Состояние мотора.

- enum PwmState {
  PwmStateOff, PwmStateBrak, PwmStateRunfwd, PwmStateRunbck,
  PwmStateInvbrak }
  Состояние ШИМ.

- enum SyncPinMode { PinGPIO = 0x00, PinSyncIn = 0x01, PinSyncOut = 0x02 }
  Тип вывода синхронизации.

Функции

Управление устройством

Функции поиска и открытия/закрытия устройств

- device_t DCCONTROL_API open_device (const char *name)
  Открывает устройство по имени name и возвращает идентификатор, который будет использоваться для обращения к устройству.

- result_t DCCONTROL_API close_device (device_t id)
  Закрывает устройство.

- result_t DCCONTROL_API probe_device (const char *name)
  Проверяет, является ли устройство с именем name XIMC-совместимым.

- result_t DCCONTROL_API enumerate_devices (int *name_count, char ***names, int probe_devices)
  Перечисляет все XIMC-совместимые устройства.

- result_t DCCONTROL_API free_enumerate_devices (int name_count, char **names)
  Освобождает память, выделенную enumerate_devices.

- result_t DCCONTROL_API init_device (device_t id)
  Инициализация устройства.

- result_t DCCONTROL_API deinit_device (device_t id)
  Деинициализация устройства.

- result_t DCCONTROL_API get_device_information (device_t id, char *manufacturer, char *manufacturer_id, char *product_description)
  Возвращает информацию об устройстве.

- result_t DCCONTROL_API reset_locks ()
  Снимает блокировку библиотеки в экстренном случае.

Управление мотором

Основные функции для управления движением мотора.

- result_t DCCONTROL_API command_move (device_t id, unsigned int pos)
  Движение в заданную позицию pos.
- result_t DCCONTROL_API command_movr (device_t id, int offset)
  Смещение на заданную позицию относительно текущей, или заданной последней командой command_move.
- result_t DCCONTROL_API command_left (device_t id)
  Движение влево.
- result_t DCCONTROL_API command_right (device_t id)
  Движение вправо.
- result_t DCCONTROL_API command_stop (device_t id)
  Остановка.
- result_t DCCONTROL_API command_zero (device_t id)
  Установка текущей позиции в 0.
- result_t DCCONTROL_API command_home (device_t id)
  Калибровка позиции.
- result_t DCCONTROL_API get_status (device_t id, state_t *state)
  Возвращает информацию о текущем состоянии устройства.

Настройки привода

Функции для чтения/записи большинства настроек контроллера

- result_t DCCONTROL_API get_pwm_freq (device_t id, unsigned int *freq)
  Чтение частоты ШИМ.
- result_t DCCONTROL_API set_pwm_freq (device_t id, unsigned int freq)
  Запись частоты ШИМ.
- result_t DCCONTROL_API get_feedback_settings (device_t id, unsigned int *ips, unsigned int *feedback_flags)
  Чтение настроек обратной связи.
- result_t DCCONTROL_API set_feedback_settings (device_t id, unsigned int ips, unsigned int feedback_flags)
  Запись настроек обратной связи.
- result_t DCCONTROL_API get_move_settings (device_t id, unsigned int *rpm, unsigned int *accel, unsigned int *tuneup_threshold)
  Чтение настроек движения.
- result_t DCCONTROL_API set_move_settings (device_t id, unsigned int rpm, unsigned int accel, unsigned int tuneup_threshold)
  Запись настроек движения.

- result_t DCCONTROL_API get_engine_settings (device_t id, engine_settings_t *engine_settings)
  Чтение настроек мотора.
- result_t DCCONTROL_API set_engine_settings (device_t id, const engine_settings_t *engine_settings)
  Запись настроек мотора.
- result_t DCCONTROL_API get_secure_settings (device_t id, unsigned int *critical_curr, unsigned int *critical_voltage, unsigned int *critical_temp)
  Чтение критических значений напряжения, тока и температуры.
- result_t DCCONTROL_API set_secure_settings (device_t id, unsigned int critical_curr, unsigned int critical_voltage, unsigned int critical_temp)
  Запись критических значений напряжения, тока и температуры.
- result_t DCCONTROL_API get_edges_settings (device_t id, int *border_flags, int *enders_flags, int *left, int *right)
  Чтение настроек границ и концевых выключателей.
- result_t DCCONTROL_API set_edges_settings (device_t id, int border_flags, int enders_flags, int left, int right)
  Запись настроек границ и концевых выключателей.
- result_t DCCONTROL_API get_pid_settings (device_t id, pid_settings_t *pid_settings)
  Чтение настроек ПИД контуров.
- result_t DCCONTROL_API set_pid_settings (device_t id, const pid_settings_t *pid_settings)
  Запись настроек ПИД контуров.
- result_t DCCONTROL_API get_sync_settings (device_t id, sync_settings_t *sync_settings)
  Чтение настроек синхронизации.
- result_t DCCONTROL_API set_sync_settings (device_t id, const sync_settings_t *sync_settings)
  Запись настроек синхронизации.
- result_t DCCONTROL_API get_home_settings (device_t id, home_settings_t *home_settings)
  Чтение настроек калибровки позиции.
- result_t DCCONTROL_API set_home_settings (device_t id, const home_settings_t home_settings)
  Запись настроек калибровки позиции.
- result_t DCCONTROL_API command_read_settings (device_t id)
  Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.
- result_t DCCONTROL_API command_save_settings (device_t id)
  Запись всех текущих настроек во flash память контроллера.

- result_t DCCONTROL_API get_serial_number (device_t id, uint32_t *serial)
  Чтение серийного номера контроллера.
- result_t DCCONTROL_API get_firmware_version (device_t id, unsigned int *major, unsigned int *minor, unsigned int *release)
  Чтение номера версии прошивки контроллера.
- result_t DCCONTROL_API get_analog_data (device_t id, analog_data_t *ad)
  Чтение аналоговых данных, содержащих данные с АЦП и нормированные значения величин.
- result_t DCCONTROL_API get_chart_data (device_t id, chart_data_t *chart_data)
  Возвращает информацию с электрическими параметрами, удобную для построения графиков.

Сервисные функции

Эти функции необходимы для начального конфигурирования, тестирования и обновления устройства.

Не применяйте их при обычной работе с контроллером. При необходимости, проконсультируйтесь пожалуйста с производителем. Неправильное использование этих функций может привести к неработоспособности устройства.

- result_t DCCONTROL_API get_calibration_coeffs (device_t id, calibration_settings_t *cs)
  Чтение калибровочных коэффициентов.
- result_t DCCONTROL_API set_calibration_coeffs (device_t id, const calibration_settings_t *cs)
  Запись калибровочных коэффициентов.
- device_t DCCONTROL_API open_raw_device (const char *name)
  Открывает устройство по имени name и возвращает идентификатор, который будет использоваться для обращения к устройству.
- result_t DCCONTROL_API set_serial_number (device_t id, uint32_t serial, uint32_t key)
  Запись серийного номера во flash память контроллера.
- result_t DCCONTROL_API has_firmware (device_t id, uint8_t *ret)
  Проверка наличия прошивки в контроллере.
- result_t DCCONTROL_API write_key (device_t id, uint32_t key)
  Запись ключа защиты Функция используется только производителем.
- result_t DCCONTROL_API command_calibrate (device_t id)
  Перевод контроллера в состояние калибровки.
- result_t DCCONTROL_API command_reset (device_t id)
  Перезагрузка контроллера.
- result_t DCCONTROL_API command_update_firmware (device_t id, const uint8_t *data, uint32_t data_size)
  Обновление прошивки.

9.1.1 Подробное описание

Header file for XIDCUSB library.

9.1.2 Макросы

9.1.2.1 #define BORDER_IS_ENCODER 0x0001

Если флаг установлен, границы определяются предустановленными точками на шкале позиции.

Если флаг сброшен, границы определяются концевыми выключателями.

9.1.2.2 #define DCCONTROL_API __attribute__((visibility("default")))

Library import macros.

Macros allows to automatically import function from shared library. It automatically expands to dllimport on msvc when including header file

9.1.2.3 #define ENGINE_ACCEL_ON 0x0010

Ускорение.

Если флаг установлен, движение происходит с ускорением.

9.1.2.4 #define ENGINE_ANTIPLAY 0x0008

Компенсация люфта.

Если флаг установлен, позиционер будет подходить к заданной точке всегда с одной стороны. Например, при подходе слева никаких дополнительных действий не совершается, а при подходе справа позиционер проходит целевую позицию на заданное расстояния и возвращается к ней опять же справа.

9.1.2.5 #define ENGINE_DYNAMIC_HOLD 0x0200

Динамическое удержание.

Если флаг установлен, воздействие мотора не прекращается при достижении заданной позиции, контроллер остается в состоянии Tune. Если флаг сброшен, при достижении заданной позиции контроллер переходит в состояние Stop,Brake.

9.1.2.6 #define ENGINE_FINISHING 0x0002

Флаг точной доводки.

Если флаг установлен, мотор выполняет специальные действия для быстрого и точного достижения заданной позиции.

9.1.2.7 #define ENGINE_HOLD 0x0100

Режим удержания.

Ротор мотора удерживается в заданной позиции, если флаг установлен.

9.1.2.8 #define ENGINE_LIMIT_CURR 0x0040

Номинальный ток мотора.

Если флаг установлен, ток через мотор ограничивается заданным номинальным значением.

9.1.2.9 #define ENGINE_LIMIT_RPM 0x0080

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

9.1.2.10 #define ENGINE_LIMIT_VOLT 0x0020

Номинальное напряжение мотора.

Если флаг установлен, напряжение на моторе ограничивается заданным номинальным значением.

9.1.2.11 #define ENGINE_MAX_SPEED 0x0004

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

9.1.2.12 #define ENGINE_ONLY_FINISHING 0x0400

Если флаг установлен, движение к точке начинается сразу в состоянии Tune.

Это специальный режим, который используется при настройке ПИД контура. Не используйте его в обычной работе.

9.1.2.13 #define ENGINE_REVERSE 0x0001

Флаг реверса.

Связывает направление вращения мотора с направлением счета текущей позиции. При сброшенном флаге (по умолчанию) прикладываемое к мотору положительное напряжение увеличивает счетчик позиции. И наоборот, при установленном флаге счетчик позиции увеличивается, когда к мотору приложено отрицательное напряжение. Измените состояние флага, если положительное вращение мотора уменьшает счетчик позиции.

9.1.2.14 #define FEEDBACK_POTENTIOMETER 0x00

Тип обратной связи.

См. также

set_feedback_settings
get_feedback_settings обратная связь с помощью потенциометра.

9.1.2.15 #define HOME_DIR_FAST 0x01

Определяет направление первоначального движения мотора после поступления команды HOME.

Если флаг установлен - вправо; иначе - влево.

9.1.2.16 #define HOME_DIR_SLOW 0x04

Определяет направление второго движения мотора.

Если флаг установлен - вправо; иначе - влево.

9.1.2.17 #define STATE_OVERHEAT 0x2000

Температура превысила допустимую вличину.

Время реакции меньше 1 мс.

9.1.2.18 #define STATE_OVERLOAD_CURRENT 0x8000

Перегрузка по току.

Время реакции меньше 1 мс.

9.1.2.19 #define STATE_OVERLOAD_VOLTAGE 0x4000

Перегрузка по напряжению.

Время реакции меньше 1 мс.

9.1.2.20 #define STATE_TTLIO_LEVEL 0x0080

Лог.

уровень на выводе синхронизации

9.1.2.21 #define TTL_SETUP_FRONT 0x0010

Настройка синхронизации в режиме входа.

Если флаг установлен, срабатывание происходит по фронту импульса. Если флаг сброшен, срабатывание происходит по спаду импульса.

9.1.2.22 #define TTL_SYNCIN_DIRRIGHT 0x0008

Направление вращения при синхронизации.

Если флаг установлен, вращение вправо. Если флаг сброшен - влево. Используется, когда установлен флаг TTL_SYNCIN_ONMOVE.

9.1.3 Перечисления

9.1.3.1 enum DevState

Состояние мотора.

Элементы перечислений:

DevStateOff мотор отключен от контроллера.

DevStateStop мотор остановлен.

DevStateMoving мотор работает.

DevStateTune мотор находится в состоянии точной доводки.

DevStateHoming мотор двигается в начальную позицию.

DevStateCalibr мотор находится в состоянии калибровки.

DevStateAlarm контроллер находится в аварийном состоянии, мотор отключен от контроллера.

9.1.3.2 enum PwmState

Состояние ШИМ.

Модуль ШИМ (широтно-импульсная модуляция) управляет силовыми MOSFET ключами, регулируя ток через мотор.

Элементы перечислений:

PwmStateOff ШИМ выключен, мотор отключен от контроллера.

PwmStateBrak режим блокировки, выводы мотора закорочены.

PwmStateRunfwd мотор вращается вправо.

PwmStateRunbck мотор вращается влево.

PwmStateInvbrak режим обратной блокировки, используется для калибровки в сервисных целях.

9.1.3.3 enum SyncPinMode

Тип вывода синхронизации.

Элементы перечислений:

PinGPIO GPIO.

PinSyncIn вход синхронизации.

PinSyncOut выход синхронизации.

9.1.4 Функции

9.1.4.1 result_t DCCONTROL_API close_device (device_t id)

Закрывает устройство.

Аргументы

id - идентификатор устройства

9.1.4.2 result_t DCCONTROL_API command_calibrate (device_t id)

Перевод контроллера в состояние калибровки.

Функция используется только производителем.

Аргументы

id идентификатор устройства

9.1.4.3 result_t DCCONTROL_API command_home (device_t id)

Калибровка позиции.

Вращает мотор в сторону, заданную HOME_DIR_FAST (если установлен - вправо; иначе - влево) со скоростью home_settings_t::fast_home. Если HOME_STOP_FAST установлен, мотор останавливается при поступлении сигнала со входа синхронизации; иначе - при достижении концевого выключателя. Вращает мотор в обратную сторону на расстояние home_settings_t::home_delta со скоростью home_settings_t::slow_home. Вращает мотор в сторону, заданную HOME_DIR_SLOW (если установлен - вправо; иначе - влево) со скоростью home_settings_t::slow_home. Если HOME_STOP_SLOW установлен, мотор останавливается при достижении концевого выключателя; иначе - при поступлении сигнала со входа синхронизации.

Аргументы

id идентификатор устройства

См. также

home_settings_t
get_home_settings
set_home_settings

9.1.4.4 result_t DCCONTROL_API command_left (device_t id)

Движение влево.

Аргументы

id идентификатор устройства

9.1.4.5 result_t DCCONTROL_API command_move (device_t id, unsigned int pos)

Движение в заданную позицию pos.

Аргументы

pos заданная позиция. Диапазон: 0..4294967295

id идентификатор устройства

9.1.4.6 result_t DCCONTROL_API command_movr (device_t id, int offset)

Смещение на заданную позицию относительно текущей, или заданной последней командой command_move.

Аргументы

offset смещение. Диапазон: -2147483647..2147483647

id идентификатор устройства

9.1.4.7 result_t DCCONTROL_API command_read_settings (device_t id)

Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.

Аргументы

id идентификатор устройства

9.1.4.8 result_t DCCONTROL_API command_reset (device_t id)

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

id идентификатор устройства

9.1.4.9 result_t DCCONTROL_API command_right (device_t id)

Движение вправо.

Аргументы

id идентификатор устройства

9.1.4.10 result_t DCCONTROL_API command_save_settings (device_t id)

Запись всех текущих настроек во flash память контроллера.

Аргументы

id идентификатор устройства

9.1.4.11 result_t DCCONTROL_API command_stop (device_t id)

Остановка.

Аргументы

id идентификатор устройства

9.1.4.12 result_t DCCONTROL_API command_update_firmware (device_t id, const uint8_t * data, uint32_t data_size)

Обновление прошивки.

Аргументы

id идентификатор устройства

data указатель на массив байтов прошивки

data_size размер массива в байтах

9.1.4.13 result_t DCCONTROL_API command_zero (device_t id)

Установка текущей позиции в 0.

Аргументы

id идентификатор устройства

9.1.4.14 result_t DCCONTROL_API deinit_device (device_t id)

Деинициализация устройства.

Чтобы начать работать с устройством необходимо вызвать init_device

Аргументы

id идентификатор устройства

9.1.4.15 result_t DCCONTROL_API enumerate_devices (int * name_count, char *** names, int probe_devices)

Перечисляет все XIMC-совместимые устройства.

Значение name_count определяет количество найденных устройств, их имена хранятся в массиве names. Память, выделенная этой функцией для массива имен names должна быть освобождена с помощью free_enumerate_devices.

Аргументы

[in,out] name_count количество найденных устройств

[out] names указатель на массив имен

[in] probe_devices если не равно 0, проверяет устройства на совместимость. Диапазон: 0..INT_MAX

9.1.4.16 result_t DCCONTROL_API free_enumerate_devices (int name_count, char ** names)

Освобождает память, выделенную enumerate_devices.

Значение name_count определяет размер массива имен names.

Аргументы

[in] name_count количество элементов в names

[in] names динамический массив имен

9.1.4.17 result_t DCCONTROL_API get_analog_data (device_t id, analog_data_t * ad)

Чтение аналоговых данных, содержащих данные с АЦП и нормированные значения величин.

Эта функция используется для тестирования и калибровки устройства.

Аргументы

id идентификатор устройства

[out] ad аналоговые данные

9.1.4.18 result_t DCCONTROL_API get_calibration_coeffs (device_t id, calibration_settings_t * cs)

Чтение калибровочных коэффициентов.

Эта функция заполняет структуру калибровочных коэффициентов, которые хранятся в памяти контроллера. Для обычной работы с устройством эта функция не требуется. Калибровочные коэффициенты применяются для калибровки контроллера.

См. также

set_calibration_coeffs

Аргументы

id идентификатор устройства

[out] cs калибровочные коэффициенты

9.1.4.19 result_t DCCONTROL_API get_chart_data (device_t id, chart_data_t * chart_data)

Возвращает информацию с электрическими параметрами, удобную для построения графиков.

См. также

chart_data_t

Аргументы

id идентификатор устройства

[out] chart_data structure with snapshot of controller parameters.

9.1.4.20 result_t DCCONTROL_API get_device_information (device_t id, char * manufacturer, char * manufacturer_id, char * product_description)

Возвращает информацию об устройстве.

Команда доступна как из инициализированного состояния, так и из исходного.

Аргументы

id идентификатор устройства.

[out] manufacturer имя производителя

[out] manufacturer_id идентификатор производителя

[out] product_description описание устройства

9.1.4.21 result_t DCCONTROL_API get_edges_settings (device_t id, int * border_flags, int * enders_flags, int * left, int * right)

Чтение настроек границ и концевых выключателей.

См. также

set_edges_settings

Аргументы

id идентификатор устройства

[out] border_flags флаги, определяющие тип границ и поведение мотора при их достижении

[out] enders_flags флаги, определяющие настройки концевых выключателей

[out] left позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER

[out] right позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER

9.1.4.22 result_t DCCONTROL_API get_engine_settings (device_t id, engine_settings_t * engine_settings)

Чтение настроек мотора.

См. также

set_engine_settings

Аргументы

id идентификатор устройства

[out] engine_settings структура с настройками мотора

9.1.4.23 result_t DCCONTROL_API get_feedback_settings (device_t id, unsigned int * ips, unsigned int * feedback_flags)

Чтение настроек обратной связи.

Аргументы

id идентификатор устройства

[out] ips разрешение энкодера (импульсов на оборот)

[out] feedback_flags флаги обратной связи

9.1.4.24 result_t DCCONTROL_API get_firmware_version (device_t id, unsigned int * major, unsigned int * minor, unsigned int * release)

Чтение номера версии прошивки контроллера.

Аргументы

id идентификатор устройства

[out] major номер основной версии

[out] minor номер дополнительной версии

[out] release номер релиза

9.1.4.25 result_t DCCONTROL_API get_home_settings (device_t id, home_settings_t * home_settings)

Чтение настроек калибровки позиции.

Эта функция заполняет структуру настроек, использующихся для калибровки позиции.

См. также

home_settings_t

Аргументы

id идентификатор устройства

[out] home_settings настройки калибровки позиции

9.1.4.26 result_t DCCONTROL_API get_move_settings (device_t id, unsigned int * rpm, unsigned int * accel, unsigned int * tuneup_threshold)

Чтение настроек движения.

Аргументы

id идентификатор устройства

[out] rpm скорость, RPM

[out] accel ускорение, RPM/c

[out] tuneup_threshold расстояние от заданной позиции, на котором включается режим tuneup

9.1.4.27 result_t DCCONTROL_API get_pid_settings (device_t id, pid_settings_t * pid_settings)

Чтение настроек ПИД контуров.

Эти настройки определяют поведение контуров позиции, скорости, напряжения и тока. Настройки различны для разных позиционеров.

См. также

set_pid_settings

Аргументы

id идентификатор устройства

[out] pid_settings настройки ПИД

9.1.4.28 result_t DCCONTROL_API get_pwm_freq (device_t id, unsigned int * freq)

Чтение частоты ШИМ.

Аргументы

id идентификатор устройства

[out] freq частота

9.1.4.29 result_t DCCONTROL_API get_secure_settings (device_t id, unsigned int * critical_curr, unsigned int * critical_voltage, unsigned int * critical_temp)

Чтение критических значений напряжения, тока и температуры.

Если одно из этих значений превышено, контроллер отключает все силовые выходы и устанавливает соответствующий флаг в структуре state_t. Время реакции меньше 1 мс.

См. также

state_t::flags

Аргументы

id идентификатор устройства

[out] critical_curr критический ток, при превышении устанавливается флан STATE_-OVERLOAD_CURRENT структуры state_t

[out] critical_voltage критическое напряжение, при превышении устанавливается флаг STATE_OVERLOAD_VOLTAGE структуры state_t

[out] critical_temp критическая температура, при превышении устанавливается флаг STATE_OVERHEAT структуры state_t

9.1.4.30 result_t DCCONTROL_API get_serial_number (device_t id, uint32_t * serial)

Чтение серийного номера контроллера.

Аргументы

id идентификатор устройства

[out] serial serial number

9.1.4.31 result_t DCCONTROL_API get_status (device_t id, state_t * state)

Возвращает информацию о текущем состоянии устройства.

Аргументы

id идентификатор устройства

[out] state структура с информацией о текущем состоянии устройства

9.1.4.32 result_t DCCONTROL_API get_sync_settings (device_t id, sync_settings_t * sync_settings)

Чтение настроек синхронизации.

Эта функция заполняет структуру с настройками синхронизации, определяющими поведение вывода синхронизации.

См. также

set_sync_settings

Аргументы

id идентификатор устройства

[out] sync_settings настройки синхронизации

9.1.4.33 result_t DCCONTROL_API has_firmware (device_t id, uint8_t * ret)

Проверка наличия прошивки в контроллере.

Аргументы

id идентификатор устройства

[out] ret не ноль, если прошивка присутствует

9.1.4.34 result_t DCCONTROL_API init_device (device_t id)

Инициализация устройства.

Аргументы

id идентификатор устройства

9.1.4.35 device_t DCCONTROL_API open_device (const char * name)

Открывает устройство по имени name и возвращает идентификатор, который будет использоваться для обращения к устройству.

Аргументы

[in] name - имя устройства, например COM3 или /dev/tty.s123

9.1.4.36 device_t DCCONTROL_API open_raw_device (const char * name)

Открывает устройство по имени name и возвращает идентификатор, который будет использоваться для обращения к устройству.

Устройство может быть чистым, без прошивки.

Аргументы

[in] name - имя устройства, например COM3 или /dev/tty.s123

9.1.4.37 result_t DCCONTROL_API probe_device (const char * name)

Проверяет, является ли устройство с именем name XIMC-совместимым.

Будте осторожны с вызовом этой функции для неизвестных устройств, т.к. она отправляет данные.

Аргументы

[in] name - имя устройства

9.1.4.38 result_t DCCONTROL_API set_calibration_coeffs (device_t id, const calibration_settings_t * cs)

Запись калибровочных коэффициентов.

Эта функция записывает калибровочные коэффициенты во flash контроллера. Все устройства при изготовлении проходят процедуру калибровки и при обычной работе их не требуется изменять. Если вы хотите произвести калибровку устройства, проконсультируйтесь с производителем. Установка неправильных коэффициентов может стать причиной перегрева и выхода контроллера из строя.

См. также

get_calibration_coeffs

Аргументы

id идентификатор устройства

[in] cs калибровочные коэффициенты

9.1.4.39 result_t DCCONTROL_API set_edges_settings (device_t id, int border_flags, int enders_flags, int left, int right)

Запись настроек границ и концевых выключателей.

Аргументы

id идентификатор устройства

[in] border_flags флаги, определяющие тип границ и поведение мотора при их достижении

[in] enders_flags флаги, определяющие настройки концевых выключателей

[in] left позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER. Диапазон: -2147483647..2147483647

[in] right позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER. Диапазон: -2147483647..2147483647

9.1.4.40 result_t DCCONTROL_API set_engine_settings (device_t id, const engine_settings_t * engine_settings)

Запись настроек мотора.

См. также

get_engine_settings

Аргументы

id идентификатор устройства

[in] engine_settings структура с настройками мотора

9.1.4.41 result_t DCCONTROL_API set_feedback_settings (device_t id, unsigned int ips, unsigned int feedback_flags)

Запись настроек обратной связи.

Аргументы

id идентификатор устройства

[in] ips разрешение энкодера (импульсов на оборот). Диапазон: 1..65535

[in] feedback_flags флаги обратной связи

9.1.4.42 result_t DCCONTROL_API set_home_settings (device_t id, const home_settings_t home_settings)

Запись настроек калибровки позиции.

Эта функция записывает структуру настроек, использующихся для калибровки позиции, в память контроллера.

См. также

home_settings_t

Аргументы

id идентификатор устройства

[out] home_settings настройки калибровки позиции

9.1.4.43 result_t DCCONTROL_API set_move_settings (device_t id, unsigned int rpm, unsigned int accel, unsigned int tuneup_threshold)

Запись настроек движения.

Аргументы

id идентификатор устройства

[in] rpm скорость, RPM. Диапазон: 1..65535

[in] accel ускорение, RPM/с. Диапазон: 1..65535

[in] tuneup_threshold расстояние от заданной позиции, на котором включается режим tuneup. Диапазон: 1..65535

9.1.4.44 result_t DCCONTROL_API set_pid_settings (device_t id, const pid_settings_t * pid_settings)

Запись настроек ПИД контуров.

Эти настройки определяют поведение контуров позиции, скорости, напряжения и тока. Настройки различны для разных позиционеров.

См. также

get_pid_settings

Аргументы

id идентификатор устройства

[in] pid_settings настройки ПИД

9.1.4.45 result_t DCCONTROL_API set_pwm_freq (device_t id, unsigned int freq)

Запись частоты ШИМ.

Аргументы

id идентификатор устройства

[in] freq частота. Диапазон: 1000..65535

9.1.4.46 result_t DCCONTROL_API set_secure_settings (device_t id, unsigned int critical_curr, unsigned int critical_voltage, unsigned int critical_temp)

Запись критических значений напряжения, тока и температуры.

Если одно из этих значений превышено, контроллер отключает все силовые выходы и устанавливает соответствующий флаг в структуре state_t. Время реакции меньше 1 мс.

См. также

state_t::flags

Аргументы

id идентификатор устройства

[in] critical_curr критический ток, при превышении устанавливается флан STATE_OVERLOAD_CURRENT структуры state_t. Диапазон: 1..65535

[in] critical_voltage критическое напряжение, при превышении устанавливается флаг STATE_OVERLOAD_VOLTAGE структуры state_t. Диапазон: 1..65535

[in] critical_temp критическая температура, при превышении устанавливается флаг STATE_OVERHEAT структуры state_t. Диапазон: 0..65535

9.1.4.47 result_t DCCONTROL_API set_serial_number (device_t id, uint32_t serial, uint32_t key)

Запись серийного номера во flash память контроллера.

Функция используется только производителем.

Аргументы

id идентификатор устройства

[in] serial серийный номер. Диапазон: 0..4294967295

[in] key ключ защиты. Диапазон: 0..4294967295

9.1.4.48 result_t DCCONTROL_API set_sync_settings (device_t id, const sync_settings_t * sync_settings)

Запись настроек синхронизации.

Эта функция записывает структуру с настройками синхронизации, определяющими поведение вывода синхронизации, в память контроллера.

См. также

get_sync_settings

Аргументы

id идентификатор устройства

[in] sync_settings настройки синхронизации

9.1.4.49 result_t DCCONTROL_API write_key (device_t id, uint32_t key)

Запись ключа защиты Функция используется только производителем.

Аргументы

id идентификатор устройства

[in] key ключ защиты. Диапазон: 0..4294967295

Предметный указатель