libximc

3.1.1

Создано системой Doxygen 1.8.13

Оглавление

Глава 1

Библиотека libximc

Документация для библиотеки libximc.

Libximc - **потокобезопасная,** кросс-платформенная библиотека для работы с контроллерами 8SM↩ C4-USB и 8SMC5-USB.

Полная документация по контроллерам доступна по ссылке.

Полная документация по API libximc доступна на странице ximc.h.

Библиотека `libximc` теперь доступна на PyPI, и её можно установить напрямую через pip:

```
pip install libximc
```

Это упрощает использование библиотеки в Python-проектах без необходимости ручной сборки.

1.1 Что делает контроллер 8SMC4-USB и 8SMC5-USB

- Поддерживает входные и выходные сигналы синхронизации для обеспечения совместной работы нескольких устройств в рамках сложной системы;
- Работает со всеми компактными шаговыми двигателями с током обмотки до 3 А, без обратной связи, а так же с шаговыми двигателями, оснащенными энкодером в цепи обратной связи, в том числе линейным энкодером на позиционере;
- Управляет контроллером с помощью готового ПО XILab или с помощью примеров, которые позволяют быстро начать программирование с использованием C++, C#, .NET, Visual Basic, Xcode, Python, Matlab, Java и LabVIEW.

1.2 Что умеет библиотека libximc

- Libximc управляет контроллером с использованием интерфейсов: USB 2.0, RS232 и Ethernet, также использует распространенный и проверенный интерфейс виртуального последовательного порта, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми ОС, в том числе под Windows, Linux и Mac OS X.
- Библиотека libximc поддерживает подключение и отключение устройств "на лету". С одним устройством в каждый момент может работать не более одного экземпляра управляющей программы - **множественный доступ управляющих программ к одному и тому же устройству не допускается!**

Предупреждения

Библиотека открывает контроллер в режиме эксклюзивного доступа. Каждый контроллер, открытый библиотекой libximc (XiLab тоже использует эту библиотеку) должен быть закрыт, прежде чем может быть использован другим процессом. Поэтому прежде чем попытаться открыть контроллер заново, проверьте, что XILab или другое программное обеспечение, взаимодействующее с контроллером, закрыто.

Пожалуйста, прочитайте Введение для начала работы с библиотекой.

Для того, чтобы использовать libximc в проекте, ознакомьтесь со страницей Как использовать с...

1.3 Содействие

Большое спасибо всем, кто отправляет нам ошибки и предложения. Мы ценим ваше время и стараемся сделать наш продукт лучше!

Глава 2

Введение

2.1 О библиотеке

Этот документ содержит всю информацию о библиотеке libximc, за исключением инструкций по сборке, которые вы можете найти в файле README.md в корневом каталоге репозитория GitHub: https://github.com/Standa-Optomechanics/libximc. Библиотека libximc использует распространенный и проверенный интерфейс виртуального последовательного порта, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми ОС: Windows, Linux, macOS для Intel и Apple Silicon (с использованием Rosetta 2), в том числе с 64-битными версиями. Библиотека поддерживает подключение и отключение устройств "на лету".
С одним устройством в каждый момент может работать не более одного экземпляра управляющей программы - множественный доступ управляющих программ к одному и тому же устройству не допускается!

2.1.1 Поддерживаемые операционные системы и требования к окружению:

- macOS 10.15 или новее
- Windows 7 или новее
- Linux на основе Debian

Глава 3

Как использовать с...

Для приобретения первых навыков использования библиотеки создано простое тестовое приложение testappeasy_C.
Языки, отличные от C-подобных, поддерживаются с помощью вызовов с преобразованием аргументов типа stdcall.

Заметки

Для работы с SDK требуется Microsoft Visual C++ Redistributable Package (поставляется с SDK, файлы vcredist_x86 или vcredist_x64).
Для работы на Linux требуется установить оба пакета libximc7_x.x.x и libximc7-dev_x.x.x целевой архитектуры в указанном порядке. Для установки пакетов можно воспользоваться .deb командой:

```
sudo dpkg -i <имя_пакета>.deb
```

Тестовое приложение может быть собрано с помощью testapp.sln. Для компиляции необходимо использовать также MS Visual C++, mingw-library.

Убедитесь, что Microsoft Visual C++ Redistributable Package установлен. Откройте проект examples/test_C/testapp_C/testapp.sln, выполните сборку и запустите приложение из среды разработки.

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testapp.c перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enumerate_hints).

Тестовое приложение может быть собрано с помощью testappeasy_C.cbp или testapp_C.cbp. Для компиляции необходимо использовать также MS Visual C++, mingw-library.

Убедитесь, что Microsoft Visual C++ Redistributable Package установлен. Откройте проект examples/test_C/testappeasy_C/testappeasy_C/testappeasy_C.cbp или examples/test_C/testapp↩
_C/testapp_C.cbp, выполните сборку и запустите приложение из среды разработки.

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 libximc.dll в текущую директорию и запустите testapp.exe.

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testapp.c перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enumerate_hints).

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

```
implib libximc.lib libximc.def
```

Затем скомпилируйте тестовое приложение:

```
bcc32 -I..\..\ximc\win32 -L..\..\ximc\win32 -DWIN32 -DNDEBUG -D_WINDOWS testapp.c libximc.lib
```

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testapp.c перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enumerate_hints).

Также существует пример использования библиотеки libximc в проекте C++ Builder, **но он не поддерживается**.

testapp должен быть собран проектом XCode testapp.xcodeproj. Используйте конфигурацию Release. Библиотека поставляется в формате macOS framework, в той же директории находится собранное тестовое приложение testapp.app.

Запустите приложение testapp.app и проверьте его работу в Console.app.

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testapp.c перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enumerate_hints).

Убедитесь, что libximc (с помощью rpm или deb) установлена на вашей системе. Пакеты должны устанавливаться с помощью package manager'а вашей ОС. Для macOS предоставляется фреймворк.

Убедитесь, что пользователь принадлежит к группе, позволяющей доступ к COM-порту (например, dip или serial).

testapp может быть собран следующим образом с установленной библиотекой:

```
make
```

Для кросс-компиляции (архитектура целевой системы отличается от архитектуры хоста) следует передать флаг -m64 или -m32 компилятору. Для сборки universal binary на macOS необходимо использовать вместо этого флаг -arch. Обратитесь к документации компилятора.

Затем запустите приложение с помощью:

```
make run
```

Примечание: make run на macOS копирует библиотеку в текущую директорию. Если вы хотите использовать библиотеку из другой директории, пожалуйста укажите в LD_LIBRARY_PATH или DYLD_LIBRARY_PATH путь к директории с библиотекой.

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testapp.c перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enumerate_hints).

Для использования в .NET предлагается обертка ximc/winX/wrappers/csharp/ximcnet.dll. Она распространяется в двух различных архитектурах. Тестировалось на платформах .NET от 2.0 до 4.5.1.

Тестовые приложения на языке C# для Visual Studio 2013 расположены в директориях test_CSharp (для C#) и test_VBNET (для VB.NET). Откройте проекты и соберите их.

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testapp.cs или testapp.vb (в зависимости от языка) перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная enumerate_hints для C#, переменная enum_hints для VB).

Как запустить пример на Linux. Перейдите в examples/test_Java/compiled-winX/ и выполните

```
java -cp /usr/share/java/libjximc.jar:test_Java.jar ru.ximc.TestJava
```

Как запустить пример на Windows. Перейдите в examples/test_Java/compiled-winX/. Запустите:

```
java -classpath libjximc.jar -classpath test_Java.jar ru.ximc.TestJava
```

Как модифицировать и пересобрать пример. Исходный текст расположен внутри test_Java.jar. Перейдите в examples/test_Java/compiled. Распакуйте jar:

```
jar xvf test_Java.jar ru META-INF
```

Затем пересоберите исходные тексты:

```
javac -classpath /usr/share/java/libjximc.jar -Xlint ru/ximc/TestJava.java
```

или для Windows или macOS:

```
javac -classpath libjximc.jar -Xlint ru/ximc/TestJava.java
```

Затем соберите jar:

```
jar cmf MANIFEST.MF test_Java.jar ru
```

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле TestJava.java перед сборкой нужно прописать IP адрес Ethernet-адаптера (переменная ENUM_HINTS).

Измените текущую директорию на examples/test_Python/xxxxtest. NB: Для работы с библиотекой libximc в примере используется модуль-обёртка ximc/crossplatform/wrappers/python/libximc.

Для запуска:

```
python xxxx.py
```

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле test_Python.py перед запуском нужно прописать IP адрес Ethernet-адаптера (переменная enum_hints).

Тестовая программа на MATLAB testximc.m располагается в директории examples/test_MATLAB.

Перед запуском:

На macOS: скопируйте ximc/macosx/libximc.framework, ximc/macosx/wrappers/ximcm.h, ximc/ximc.h в директорию examples/test_MATLAB. Установите XCode, совместимый с Matlab

На Linux: установите libximc*deb и libximc-dev*deb нужной архитектуры. Далее скопируйте ximc/macosx/wrappers/ximcm.h в директорию examples/matlab. Установите gcc, совместимый с Matlab.

Для проверки совместимых XCode и gcc проверьте документы https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/support/sysreq/files/SystemRequirements-Release2014a_SupportedCompilers.pdf или похожие.

На Windows: перед запуском ничего делать не нужно

Измените текущую директорию в MATLAB на examples/test_MATLAB. Затем запустите в MATLAB:

```
testximc
```

В случае, если планируется использовать Ethernet-адаптер 8SMC4-USB-Eth1, в файле testximc.m перед запуском нужно прописать IP адрес Ethernet-адаптера (переменная enum_hints).

3.1 Логирование в файл

Если программа, использующая libximc, запущена с установленной переменной окружения XILOG, то это включит логирование в файл. Значение переменной XILOG будет использовано как имя файла. Файл будет открыт на запись при первом событии лога и закрыт при завершении программы, использующей libximc. В лог записываются события отправки данных в контроллер и приема данных из контроллера, а также открытия и закрытия порта.

3.2 Требуемые права доступа

Библиотеке не требуются особые права для выполнения, но нужены права доступа на чтение-запись в USB-COM устройства в системе. Исключением из этого правила является функция только для ОС Windows "fix_usbser_sys()" - если процесс использующий библиотеку не имеет повышенных прав, то при вызове этой функции программная переустановка устройства не будет работать.

3.3 Си-профили

Си-профили это набор заголовочных файлов, распространяемых вместе с библиотекой libximc. Они позволяют в программе на языке C/C++ загрузить в контроллер настройки одной из поддерживаемых подвижек вызовом всего одной функции. Пример использования си-профилей вы можете посмотреть в директории примеров "examples/test_C/testprofile_C".

3.4 Python-профили

Python-профили - это набор конфигурационных функций, распространяемых вместе с библиотекой libximc. Они позволяют в программе на языке Python загрузить в контроллер настройки одной из поддерживаемых подвижек вызовом всего одной функции.
Пример использования python-профилей вы можете посмотреть в директории примеров "examples/test_Python/profiletest/testpythonprofile.py".

Глава 4

Работа с пользовательскими единицами

Кроме работы в основных единицах (шагах, отчетах энкодера) библиотека позволяет работать с пользовательскими единицами. Для этого используются:

- Структура пересчета единиц calibration_t
- Функции дублеры для работы с пользовательскими единицами и структуры данных для них
- Таблица коррекции координат для более точного позиционирования

4.1 Структура пересчета единиц calibration_t

Для задания пересчета из основных единиц в пользовательские и обратно используется структура calibration_t. С помощью коэффициентов A и MicrostepMode, заданных в этой структуре, происходит пересчет из шагов и микрошагов являющихся целыми числами в пользовательское значение действительного типа и обратно.

Формулы пересчета:

- Пересчет в пользовательские единицы.

```
user_value = A*(step + mstep/pow(2,MicrostepMode-1))
```

- Пересчет из пользовательских единиц.

```
step = (int)(user_value/A)
mstep = (user_value/A - step)*pow(2,MicrostepMode-1)
```

4.2 Функции дублеры для работы с пользовательскими единицами и структуры данных для них

Структуры и функции для работы с пользовательскими единицами имеют постфикс _calb. Пользователь, используя данные функции, может выполнять все действия в собственных единицах, не беспокоясь о том, что и как считает контроллер. Для _calb функций отдельных описаний нет. Они выполняют те же действия, что и базовые функции. Разница между ними и базовыми функциями в типах данных положения, скоростей и ускорений, определенных как пользовательские. Если требуются уточнения для _calb функций - они оформлены в виде примечаний в описании базовых функций.

4.3 Таблица коррекции координат для более точного позиционирования

Некоторые функции для работы с пользовательскими единицами поддерживают преобразование координат с использованием корректировочной таблицы. Для загрузки таблицы из файла используется функция set_correction_table(). В ее описании описаны функции и их данные поддерживающие коррекцию движения.

Заметки

Для полей данных которые корректируются в случае загрузки таблицы в описании поля записано - корректируется таблицей.

Формат файла:

- два столбца разделенных табуляцией;
- заголовки столбцов строковые;
- данные действительные, разделитель - точка;
- первый столбец координата, второй - отклонение вызванное ошибкой механики;
- между координатами отклонение рассчитывается линейно;
- за диапазоном - константа равная отклонению на границе;
- максимальная длина таблицы 100 строк.

Пример файла:

X	dX
0	0
5.0	0.005
10.↩ 0	-0.01

Глава 5

Структуры данных

5.1 Структура accessories_settings_t

Устарело.

Поля данных

- char MagneticBrakeInfo [25]

  Производитель и номер магнитного тормоза, Максимальная длина строки: 24 символов.
- float MBRatedVoltage

  Номинальное напряжение для управления магнитным тормозом (В).
- float MBRatedCurrent

  Номинальный ток для управления магнитным тормозом (А).
- float MBTorque

  *Удерживающий момент (мН * м).*
- unsigned int MBSettings

  Флаги настроек энкодера.
- char TemperatureSensorInfo [25]

  Производитель и номер температурного датчика, Максимальная длина строки: 24 символов.
- float TSMin

  Минимальная измеряемая температура (градусов Цельсия).
- float TSMax

  Максимальная измеряемая температура (градусов Цельсия) Тип данных: float.
- float TSGrad

  Температурный градиент (В/градусов Цельсия).
- unsigned int TSSettings

  Флаги настроек температурного датчика.
- unsigned int LimitSwitchesSettings

  Флаги настроек температурного датчика.

5.1.1 Подробное описание

Устарело.

Информация о дополнительных аксессуарах.

См. также

set_accessories_settings
get_accessories_settings
get_accessories_settings, set_accessories_settings

5.1.2 Поля

5.1.2.1 LimitSwitchesSettings

```
unsigned int LimitSwitchesSettings
```

Флаги настроек температурного датчика.

5.1.2.2 MagneticBrakeInfo

```
char MagneticBrakeInfo[25]
```

Производитель и номер магнитного тормоза, Максимальная длина строки: 24 символов.

5.1.2.3 MBRatedCurrent

```
float MBRatedCurrent
```

Номинальный ток для управления магнитным тормозом (A).

Тип данных: float.

5.1.2.4 MBRatedVoltage

```
float MBRatedVoltage
```

Номинальное напряжение для управления магнитным тормозом (B).

Тип данных: float.

5.1.2.5 MBSettings

```
unsigned int MBSettings
```

Флаги настроек энкодера.

5.1.2.6 MBTorque

```
float MBTorque
```

Удерживающий момент (мН ∗ м).

Тип данных: float.

5.1.2.7 TemperatureSensorInfo

```
char TemperatureSensorInfo[25]
```

Производитель и номер температурного датчика, Максимальная длина строки: 24 символов.

5.1.2.8 TSGrad

```
float TSGrad
```

Температурный градиент (В/градусов Цельсия).

Тип данных: float.

5.1.2.9 TSMax

```
float TSMax
```

Максимальная измеряемая температура (градусов Цельсия) Тип данных: float.

5.1.2.10 TSMin

```
float TSMin
```

Минимальная измеряемая температура (градусов Цельсия).

Тип данных: float.

5.1.2.11 TSSettings

```
unsigned int TSSettings
```

Флаги настроек температурного датчика.

5.2 Структура analog_data_t

Аналоговые данные.

Поля данных

- unsigned int A1Voltage_ADC
  "Выходное напряжение на 1 выводе обмотки A" - необработанные данные с АЦП.
- unsigned int A2Voltage_ADC
  "Выходное напряжение на 2 выводе обмотки A" - необработанные данные с АЦП.
- unsigned int B1Voltage_ADC
  "Выходное напряжение на 1 выводе обмотки B" - необработанные данные с АЦП.
- unsigned int B2Voltage_ADC
  "Выходное напряжение на 2 выводе обмотки B" - необработанные данные с АЦП.
- unsigned int SupVoltage_ADC
  "Напряжение питания ключей H-моста" необработанные данные с АЦП.
- unsigned int ACurrent_ADC
  "Ток через обмотку A" - необработанные данные с АЦП.
- unsigned int BCurrent_ADC
  "Ток через обмотку B" - необработанные данные с АЦП.
- unsigned int FullCurrent_ADC
  "Полный ток" - необработанные данные с АЦП.
- unsigned int Temp_ADC
  Напряжение с датчика температуры, необработанные данные с АЦП.
- unsigned int Joy_ADC
  Необработанные данные с АЦП джойстика.
- unsigned int Pot_ADC
  Напряжение на аналоговом входе, необработанные данные с АЦП
- unsigned int Enc_Check_ADC
  Напряжение на линии проверки типа энкодера, необработанные данные АЦП.
- unsigned int **deprecated0**
- int A1Voltage
  "Выходное напряжение на 1 выводе обмотки A" - откалиброванные данные (в десятках мВ).
- int A2Voltage
  "Выходное напряжение на 2 выводе обмотки A" - откалиброванные данные (в десятках мВ).
- int B1Voltage
  "Выходное напряжение на 1 выводе обмотки B" - откалиброванные данные (в десятках мВ).
- int B2Voltage
  "Выходное напряжение на 2 выводе обмотки B" - откалиброванные данные (в десятках мВ).
- int SupVoltage
  "Напряжение питания ключей H-моста" - откалиброванные данные (в десятках мВ).

- int ACurrent

  "Ток через обмотку A" - откалиброванные данные (в мА).
- int BCurrent

  "Ток через обмотку B" - откалиброванные данные (в мА).
- int FullCurrent

  "Полный ток" - откалиброванные данные (в мА).
- int Temp

  Температура, откалиброванные данные (в десятых долях градуса Цельсия).
- int Joy

  Калиброванные данные джойстика в условных внутренних единицах.
- int Pot

  Аналоговый вход во внутренних единицах.
- int Enc_Check

  Напряжение на линии проверки типа энкодера, экспонениально сглаженные данные АЦП.
- unsigned int **deprecated1** [2]
- int R

  Сопротивление обмоток двигателя (для шагового двигателя), в мОм
- int L

  Псевдоиндуктивность обмоток двигателя (для шагового двигателя), в мкГн

5.2.1 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
get_analog_data

5.2.2 Поля

5.2.2.1 A1Voltage

```
int A1Voltage
```

"Выходное напряжение на 1 выводе обмотки A" - откалиброванные данные (в десятках мВ).

5.2.2.2 A1Voltage_ADC

```
unsigned int A1Voltage_ADC
```

"Выходное напряжение на 1 выводе обмотки A" - необработанные данные с АЦП.

5.2.2.3 A2Voltage

```
int A2Voltage
```

"Выходное напряжение на 2 выводе обмотки A" - откалиброванные данные (в десятках мВ).

5.2.2.4 A2Voltage_ADC

```
unsigned int A2Voltage_ADC
```

"Выходное напряжение на 2 выводе обмотки A" - необработанные данные с АЦП.

5.2.2.5 ACurrent

```
int ACurrent
```

"Ток через обмотку A" - откалиброванные данные (в мА).

5.2.2.6 ACurrent_ADC

```
unsigned int ACurrent_ADC
```

"Ток через обмотку A" - необработанные данные с АЦП.

5.2.2.7 B1Voltage

```
int B1Voltage
```

"Выходное напряжение на 1 выводе обмотки B" - откалиброванные данные (в десятках мВ).

5.2.2.8 B1Voltage_ADC

```
unsigned int B1Voltage_ADC
```

"Выходное напряжение на 1 выводе обмотки B" - необработанные данные с АЦП.

5.2.2.9 B2Voltage

```
int B2Voltage
```

"Выходное напряжение на 2 выводе обмотки B" - откалиброванные данные (в десятках мВ).

5.2.2.10 B2Voltage_ADC

```
unsigned int B2Voltage_ADC
```

"Выходное напряжение на 2 выводе обмотки B" - необработанные данные с АЦП.

5.2.2.11 BCurrent

```
int BCurrent
```

"Ток через обмотку B" - откалиброванные данные (в мА).

5.2.2.12 BCurrent_ADC

```
unsigned int BCurrent_ADC
```

"Ток через обмотку B" - необработанные данные с АЦП.

5.2.2.13 Enc_Check

```
int Enc_Check
```

Напряжение на линии проверки типа энкодера, экспонениально сглаженные данные АЦП.

Используется для определения типа энкодера: с однофазным выходом или дифференциальный.

5.2.2.14 Enc_Check_ADC

```
unsigned int Enc_Check_ADC
```

Напряжение на линии проверки типа энкодера, необработанные данные АЦП.

Используется для определения типа энкодера: с однофазным выходом или дифференциальный.

5.2.2.15 FullCurrent

```
int FullCurrent
```

"Полный ток" - откалиброванные данные (в мА).

5.2.2.16 FullCurrent_ADC

```
unsigned int FullCurrent_ADC
```

"Полный ток" - необработанные данные с АЦП.

5.2.2.17 Joy

```
int Joy
```

Калиброванные данные джойстика в условных внутренних единицах.

Диапазон: 0..10000

5.2.2.18 Joy_ADC

```
unsigned int Joy_ADC
```

Необработанные данные с АЦП джойстика.

5.2.2.19 Pot

```
int Pot
```

Аналоговый вход во внутренних единицах.

Диапазон: 0..10000

5.2.2.20 SupVoltage

```
int SupVoltage
```

"Напряжение питания ключей H-моста" - откалиброванные данные (в десятках мВ).

5.2.2.21 SupVoltage_ADC

```
unsigned int SupVoltage_ADC
```

"Напряжение питания ключей H-моста" необработанные данные с АЦП.

5.2.2.22 Temp

```
int Temp
```

Температура, откалиброванные данные (в десятых долях градуса Цельсия).

5.2.2.23 Temp_ADC

```
unsigned int Temp_ADC
```

Напряжение с датчика температуры, необработанные данные с АЦП.

5.3 Структура brake_settings_t

Настройки тормоза.

Поля данных

- unsigned int t1
  Время в мс между включением питания мотора и отключением тормоза.
- unsigned int t2
  Время в мс между отключением тормоза и готовностью к движению.
- unsigned int t3
  Время в мс между остановкой мотора и включением тормоза.
- unsigned int t4
  Время в мс между включением тормоза и отключением питания мотора.
- unsigned int BrakeFlags
  Флаги настроек тормоза.

5.3.1 Подробное описание

Настройки тормоза.

Эта структура содержит параметры управления тормозом.

См. также

set_brake_settings
get_brake_settings
get_brake_settings, set_brake_settings

5.3.2 Поля

5.3.2.1 BrakeFlags

```
unsigned int BrakeFlags
```

Флаги настроек тормоза.

5.3.2.2 t1

```
unsigned int t1
```

Время в мс между включением питания мотора и отключением тормоза.

5.3.2.3 t2

```
unsigned int t2
```

Время в мс между отключением тормоза и готовностью к движению.

Все команды движения начинают выполняться только по истечении этого времени.

5.3.2.4 t3

```
unsigned int t3
```

Время в мс между остановкой мотора и включением тормоза.

5.3.2.5 t4

```
unsigned int t4
```

Время в мс между включением тормоза и отключением питания мотора.

5.4 Структура calibration_settings_t

Калибровочные коэффициенты.

Поля данных

- float CSS1_A
  Коэффициент масштабирования для аналоговых измерений тока в обмотке A.
- float CSS1_B
  Коэффициент сдвига для аналоговых измерений тока в обмотке A.
- float CSS2_A
  Коэффициент масштабирования для аналоговых измерений тока в обмотке B.
- float CSS2_B
  Коэффициент сдвига для аналоговых измерений тока в обмотке B.
- float FullCurrent_A
  Коэффициент масштабирования для аналоговых измерений полного тока.
- float FullCurrent_B
  Коэффициент сдвига для аналоговых измерений полного тока.

5.4.1 Подробное описание

Калибровочные коэффициенты.

Эта структура содержит калибровочные коэффициенты. Эти коэффициенты используются для пересчёта кодов АЦП в токи обмоток и полный ток потребления. Коэффициенты сгруппированы в пары, XXX_A и XXX_B; пары представляют собой коэффициенты линейного уравнения. Первый коэффициент - тангенс угла наклона, второй - постоянное смещение. Таким образом, XXX_Current[mA] = XXX_A[mA/ADC]*XXX_ADC_CODE[ADC] + XXX_B[mA].

См. также

get_calibration_settings
set_calibration_settings
get_calibration_settings, set_calibration_settings

5.4.2 Поля

5.4.2.1 CSS1_A

```
float CSS1_A
```

Коэффициент масштабирования для аналоговых измерений тока в обмотке A.

5.4.2.2 CSS1_B

```
float CSS1_B
```

Коэффициент сдвига для аналоговых измерений тока в обмотке A.

5.4.2.3 CSS2_A

```
float CSS2_A
```

Коэффициент масштабирования для аналоговых измерений тока в обмотке B.

5.4.2.4 CSS2_B

```
float CSS2_B
```

Коэффициент сдвига для аналоговых измерений тока в обмотке B.

5.4.2.5 FullCurrent_A

```
float FullCurrent_A
```

Коэффициент масштабирования для аналоговых измерений полного тока.

5.4.2.6 FullCurrent_B

```
float FullCurrent_B
```

Коэффициент сдвига для аналоговых измерений полного тока.

5.5 Структура calibration_t

Структура калибровок

Поля данных

- double A

  Коэффициент преобразования, равный количеству миллиметров (или других пользовательских единиц) на один шаг.
- unsigned int MicrostepMode

  Настройка контроллера, определяющая режим пошагового деления.

5.5.1 Подробное описание

Структура калибровок

Где найти все значения для расчета?

- XILab (не забудьте загрузить профиль для вашего позиционера. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg):
  - В настройках XILab перейдите во вкладку user units. Разделите второе число на первое — это и будет коэффициент A
  - В настройках XILab, перейдите во вкладку DC motor/BLDC motor/Stepper motor (зависит от используемого типа двигателя). Подставьте значение из поля Encoder counts per turn в формулу подсчета B коэффициента
- Профиль (откройте профиль любым текстовым редактором. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg):
  - Найдите в файле профиля поля Step_multiplier= и Unit_multiplier=. Разделите второе число на первое — это и будет коэффициент A
  - Найдите в файле профиля поле Encoder_CPT=. Подставьте значение из этого поля в формулу подсчета B коэффициента вместо ENCODER_COUNTS_PER_TURN

Как посчитать Speed, Accel, Decel и AntiplaySpeed в пользовательских единицах при использовании шагового двигателя с энкодером или DC/BLDC двигателей?

1. Используя XILab, загрузите профиль для вашего позиционера. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg
2. Включите режим Feedback encoder, если он не был включен ранее
3. Вводите скорость в пользовательских единицах в поле Working speed
4. Во вкладке User units выключите флаг User units. Это позволит видеть значение в поле Working speed в RPM
5. Умножьте значение из поля Working speed (в RPM) на коэффициент B. Например: $480 * 0.0000009375 = 0.00045$. Значение 0.00045 для позиционера 8MT173-25-MEn1 в режиме Encoder будет равно скорости 2 мм/сек

Ускорение, замедление и скорость в режиме антилюфта считаются аналогично

5.5.2 Поля

5.5.2.1 A

`double A`

Коэффициент преобразования, равный количеству миллиметров (или других пользовательских единиц) на один шаг.

Должен быть отличным от нуля и положительным. Используется для пересчёта положений и перемещений.

- Для шагового двигателя без энкодера используется только один коэффициент A. Формула пересчёта: [user_unit/steps]. Пример: 800 шагов = 1 мм, тогда A = 1/800 = 0.00125
- При использовании шагового двигателя с энкодером или двигателей DC/BLDC позиция задаётся в counts, но скорость, ускорение/замедление и скорость в режиме антилюфта задаются в RPM, поэтому требуются два коэффициента:
  - A. Формула пересчёта для позиции: [user_unit/counts]. Пример: 16000 counts = 1 мм, тогда A = 1/16000 = 0.0000625
  - B. Формула пересчёта для скорости, ускорения/замедления и скорости в режиме антилюфта: [60/ENCODER_COUNTS_PER_TURN $*$ A]. Пример: 60 / 4000 $*$ 0.0000625 = 0.0000009375

5.5.2.2 MicrostepMode

`unsigned int MicrostepMode`

Настройка контроллера, определяющая режим пошагового деления.

5.6 Структура chart_data_t

Дополнительное состояние устройства.

Поля данных

- int WindingVoltageA

  В случае ШД напряжение на обмотке A (в десятках мВ); в случае бесщеточного - напряжение на первой обмотке; в случае DC - на единственной.
- int WindingVoltageB

  В случае ШД напряжение на обмотке B (в десятках мВ); в случае бесщеточного - напряжение на второй обмотке; в случае DC - не используется.
- int WindingVoltageC

  В случае бесщеточного - напряжение на третьей обмотке (в десятках мВ); в случае ШД и DC не используется.
- int WindingCurrentA

  В случае ШД - ток в обмотке A (в мА); в случае бесщеточного - ток в первой обмотке; в случае DC - в единственной.
- int WindingCurrentB

  В случае ШД - ток в обмотке B (в мА); в случае бесщеточного - ток в второй обмотке; в случае DC не используется.
- int WindingCurrentC

  В случае бесщеточного - ток в третьей обмотке (в мА); в случае ШД и DC не используется.
- unsigned int Pot

  Значение на аналоговом входе.
- unsigned int Joy

  Положение джойстика в десятитысячных долях.
- int AveragedPowerRatio

  Отношение подаваемой на мотор мощности к номинальной мощности, в процентах.

5.6.1 Подробное описание

Дополнительное состояние устройства.

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

См. также

get_chart_data
get_chart_data

5.6.2 Поля

5.6.2.1 AveragedPowerRatio

```
int AveragedPowerRatio
```

Отношение подаваемой на мотор мощности к номинальной мощности, в процентах.

5.6.2.2 Joy

```
unsigned int Joy
```

Положение джойстика в десятитысячных долях.

Диапазон: 0..10000

5.6.2.3 Pot

```
unsigned int Pot
```

Значение на аналоговом входе.

Диапазон: 0..10000

5.6.2.4 WindingCurrentA

```
int WindingCurrentA
```

В случае ШД - ток в обмотке A (в мА); в случае бесщеточного - ток в первой обмотке; в случае DC - в единственной.

5.6.2.5 WindingCurrentB

```
int WindingCurrentB
```

В случае ШД - ток в обмотке B (в мА); в случае бесщеточного - ток в второй обмотке; в случае DC не используется.

5.6.2.6 WindingCurrentC

```
int WindingCurrentC
```

В случае бесщеточного - ток в третьей обмотке (в мА); в случае ШД и DC не используется.

5.6.2.7 WindingVoltageA

```
int WindingVoltageA
```

В случае ШД напряжение на обмотке A (в десятках мВ); в случае бесщеточного - напряжение на первой обмотке; в случае DC - на единственной.

5.6.2.8 WindingVoltageB

```
int WindingVoltageB
```

В случае ШД напряжение на обмотке B (в десятках мВ); в случае бесщеточного - напряжение на второй обмотке; в случае DC - не используется.

5.6.2.9 WindingVoltageC

```
int WindingVoltageC
```

В случае бесщеточного - напряжение на третьей обмотке (в десятках мВ); в случае ШД и DC не используется.

5.7 Структура control_settings_calb_t

Настройки управления с использованием пользовательских единиц.

Поля данных

- float MaxSpeed [10]

  Массив скоростей, использующийся при управлении джойстиком или кнопками влево/вправо.
- unsigned int Timeout [9]

  timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).
- unsigned int MaxClickTime

  Максимальное время клика (в мс).
- unsigned int Flags

  Флаги управления.
- float DeltaPosition

  Смещение (дельта) позиции

5.7.1 Подробное описание

Настройки управления с использованием пользовательских единиц.

При выборе CTL_MODE= 1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE= 2 включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed [0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+ 1]. При переходе от MaxSpeed [i] на MaxSpeed [i+ 1] действует ускорение, как обычно.

См. также

set_control_settings_calb
get_control_settings_calb
get_control_settings, set_control_settings

5.7.2 Поля

5.7.2.1 Flags

```
unsigned int Flags
```

Флаги управления.

5.7.2.2 MaxClickTime

```
unsigned int MaxClickTime
```

Максимальное время клика (в мс).

До истечения этого времени первая скорость не включается.

5.7.2.3 MaxSpeed

```
float MaxSpeed[10]
```

Массив скоростей, использующийся при управлении джойстиком или кнопками влево/вправо.

5.7.2.4 Timeout

```
unsigned int Timeout[9]
```

timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).

5.8 Структура control_settings_t

Настройки управления.

Поля данных

- unsigned int MaxSpeed [10]

  Массив скоростей (в полных шагах), использующийся при управлении джойстиком или кнопками влево/вправо.
- unsigned int uMaxSpeed [10]

  Массив скоростей (в микрошагах), использующийся при управлении джойстиком или кнопками влево/вправо.
- unsigned int Timeout [9]

  timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).
- unsigned int MaxClickTime

  Максимальное время клика (в мс).
- unsigned int Flags

  Флаги управления.
- int DeltaPosition

  Смещение (дельта) позиции (в полных шагах)
- int uDeltaPosition

  Дробная часть смещения в микрошагах.

5.8.1 Подробное описание

Настройки управления.

При выборе CTL_MODE= 1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE= 2 включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed [0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+ 1]. При переходе от MaxSpeed [i] на MaxSpeed [i+ 1] действует ускорение, как обычно.

См. также

set_control_settings
get_control_settings
get_control_settings, set_control_settings

5.8.2 Поля

5.8.2.1 Flags

```
unsigned int Flags
```

Флаги управления.

5.8.2.2 MaxClickTime

```
unsigned int MaxClickTime
```

Максимальное время клика (в мс).

До истечения этого времени первая скорость не включается.

5.8.2.3 MaxSpeed

```
unsigned int MaxSpeed[10]
```

Массив скоростей (в полных шагах), использующийся при управлении джойстиком или кнопками влево/вправо.

Диапазон: 0..100000.

5.8.2.4 Timeout

```
unsigned int Timeout[9]
```

timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+ 1] (используется только при управлении кнопками).

5.8.2.5 uDeltaPosition

```
int uDeltaPosition
```

Дробная часть смещения в микрошагах.

Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_↩ settings).

5.8.2.6 uMaxSpeed

```
unsigned int uMaxSpeed[10]
```

Массив скоростей (в микрошагах), использующийся при управлении джойстиком или кнопками влево/вправо.

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.9 Структура controller_name_t

Пользовательское имя контроллера и флаги настройки.

Поля данных

- char ControllerName [17]

  Пользовательское имя контроллера.
- unsigned int CtrlFlags

  Флаги настроек контроллера.

5.9.1 Подробное описание

Пользовательское имя контроллера и флаги настройки.

См. также

get_controller_name, set_controller_name

5.9.2 Поля

5.9.2.1 ControllerName

```
char ControllerName[17]
```

Пользовательское имя контроллера.

Может быть установлено пользователем для его удобства. Максимальная длина строки: 16 символов.

5.9.2.2 CtrlFlags

```
unsigned int CtrlFlags
```

Флаги настроек контроллера.

5.10 Структура ctp_settings_t

Настройки контроля позиции(для шагового двигателя).

Поля данных

- unsigned int CTPMinError

  Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг STATE_RT_↩ ERROR.
- unsigned int CTPFlags

  Флаги контроля позиции.

5.10.1 Подробное описание

Настройки контроля позиции(для шагового двигателя).

При управлении ШД с энкодером (CTP_BASE 0) появляется возможность обнаруживать потерю шагов. Контроллер знает количество шагов на оборот (GENG::StepsPerRev) и разрешение энкодера (GFBS::IPT). При включении контроля (флаг CTP_ENABLED), контроллер запоминает текущую позицию в шагах ШД и текущую позицию энкодера. Далее, на каждом шаге позиция энкодера преобразовывается в шаги и если разница оказывается больше CTPMinError, устанавливается флаг S↩TATE_CTP_ERROR и устанавливается состояние ALARM.

При управлении ШД с датчиком оборотов (CTP_BASE 1), позиция контролируется по нему. По активному фронту на входе синхронизации контроллер запоминает текущее значение шагов. Далее, при каждом обороте проверяет, на сколько шагов сместились. При рассогласовании более CTPMin↩Error устанавливается флаг STATE_CTP_ERROR и устанавливается состояние ALARM.

См. также

set_ctp_settings
get_ctp_settings
get_ctp_settings, set_ctp_settings

5.10.2 Поля

5.10.2.1 CTPFlags

```
unsigned int CTPFlags
```

Флаги контроля позиции.

5.10.2.2 CTPMinError

```
unsigned int CTPMinError
```

Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг STATE_RT_E↩RROR.

Измеряется в шагах ШД.

5.11 Структура debug_read_t

Отладочные данные.

Поля данных

- uint8_t DebugData [128]
  Отладочные данные.

5.11.1 Подробное описание

Отладочные данные.

Эти данные используются в сервисных целях для тестирования и отладки устройства.

См. также

get_debug_read

5.11.2 Поля

5.11.2.1 DebugData

```
uint8_t DebugData[128]
```

Отладочные данные.

5.12 Структура debug_write_t

Отладочные данные.

Поля данных

- uint8_t DebugData [128]
  Отладочные данные.

5.12.1 Подробное описание

Отладочные данные.

Эти данные используются в сервисных целях для тестирования и отладки устройства.

См. также

set_debug_write

5.12.2 Поля

5.12.2.1 DebugData

```
uint8_t DebugData[128]
```

Отладочные данные.

5.13 Структура device_information_t

Информации о контроллере.

Поля данных

- char Manufacturer [5]
  Производитель
- char ManufacturerId [3]
  Идентификатор производителя
- char ProductDescription [9]
  Описание продукта
- unsigned int Major
  Основной номер версии железа.
- unsigned int Minor
  Второстепенный номер версии железа.
- unsigned int Release
  Номер правок этой версии железа.

5.13.1 Подробное описание

Информации о контроллере.

См. также

get_device_information
get_device_information_impl

5.13.2 Поля

5.13.2.1 Major

```
unsigned int Major
```

Основной номер версии железа.

5.13.2.2 Minor

```
unsigned int Minor
```

Второстепенный номер версии железа.

5.13.2.3 Release

```
unsigned int Release
```

Номер правок этой версии железа.

5.14 Структура device_network_information_t

Структура данных с информацией о сетевом устройстве.

Поля данных

- uint32_t ipv4
  IPv4-адрес, передаваемый в сетевом байтовом порядке (big-endian byte order)
- char nodename [16]
  имя узла Bindy, на котором размещено устройство
- uint32_t axis_state
  флаги, представляющие состояние устройства
- char locker_username [16]
  имя пользователя, заблокировавшего устройство (если таковое имеется)
- char locker_nodename [16]
  имя узла Bindy, которое использовалось для блокировки устройства (если таковое имеется)
- time_t locked_time
  время, в которое была получена блокировка (UTC, микросекунды с момента начала эпохи)

5.14.1 Подробное описание

Структура данных с информацией о сетевом устройстве.

5.15 Структура edges_settings_calb_t

Настройки границ с использованием пользовательских единиц.

Поля данных

- unsigned int BorderFlags

  Флаги границ.
- unsigned int EnderFlags

  Флаги концевых выключателей.
- float LeftBorder

  Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.
- float RightBorder

  Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.

5.15.1 Подробное описание

Настройки границ с использованием пользовательских единиц.

Эта структура содержит настройки границ и концевых выключателей. Пожалуйста, загружайте новые настройки когда вы меняете позиционер. Помните, что неправильные настройки мотора могут повредить оборудование.

См. также

set_edges_settings_calb
get_edges_settings_calb
get_edges_settings, set_edges_settings

5.15.2 Поля

5.15.2.1 BorderFlags

```
unsigned int BorderFlags
```

Флаги границ.

5.15.2.2 EnderFlags

```
unsigned int EnderFlags
```

Флаги концевых выключателей.

5.15.2.3 LeftBorder

```
float LeftBorder
```

Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.

Корректируется таблицей.

5.15.2.4 RightBorder

```
float RightBorder
```

Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.

Корректируется таблицей.

5.16 Структура edges_settings_t

Настройки границ.

Поля данных

- unsigned int BorderFlags
  Флаги границ.
- unsigned int EnderFlags
  Флаги концевых выключателей.
- int LeftBorder
  Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.
- int uLeftBorder
  Позиция левой границы в микрошагах (используется только с шаговым двигателем).
- int RightBorder
  Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.
- int uRightBorder
  Позиция правой границы в микрошагах (используется только с шаговым двигателем).

5.16.1 Подробное описание

Настройки границ.

Эта структура содержит настройки границ и концевых выключателей. Пожалуйста, загружайте новые настройки когда вы меняете позиционер. Помните, что неправильные настройки мотора могут повредить оборудование.

См. также

set_edges_settings
get_edges_settings
get_edges_settings, set_edges_settings

5.16.2 Поля

5.16.2.1 BorderFlags

```
unsigned int BorderFlags
```

Флаги границ.

5.16.2.2 EnderFlags

```
unsigned int EnderFlags
```

Флаги концевых выключателей.

5.16.2.3 LeftBorder

```
int LeftBorder
```

Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.

5.16.2.4 RightBorder

```
int RightBorder
```

Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.

5.16.2.5 uLeftBorder

```
int uLeftBorder
```

Позиция левой границы в микрошагах (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.16.2.6 uRightBorder

```
int uRightBorder
```

Позиция правой границы в микрошагах (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.17 Структура emf_settings_t

Настройки EMF.

Поля данных

- float L

  Индуктивность обмоток двигателя.
- float R

  Сопротивление обмоток двигателя.
- float Km

  Электромеханический коэффициент двигателя.
- unsigned int BackEMFFlags

  Флаги автоопределения характеристик обмоток двигателя.

5.17.1 Подробное описание

Настройки EMF.

Эта структура содержит данные электромеханических характеристик(EMF) двигателя. Они определяют индуктивность, сопротивление и электромеханический коэффициент двигателя. Эти данные хранятся во flash-памяти контроллера. Пожалуйста, загружайте новые настройки, когда вы меняете мотор. Помните, что неправильные настройки EMF могут повредить оборудование.

См. также

set_emf_settings
get_emf_settings
get_emf_settings, set_emf_settings

5.17.2 Поля

5.17.2.1 BackEMFFlags

```
unsigned int BackEMFFlags
```

Флаги автоопределения характеристик обмоток двигателя.

5.17.2.2 Km

```
float Km
```

Электромеханический коэффициент двигателя.

5.17.2.3 L

```
float L
```

Индуктивность обмоток двигателя.

5.17.2.4 R

```
float R
```

Сопротивление обмоток двигателя.

5.18 Структура encoder_information_t

Устарело.

Поля данных

- char Manufacturer [17]

  Производитель.
- char PartNumber [25]

  Серия и номер модели.

5.18.1 Подробное описание

Устарело.

Информация об энкодере.

См. также

set_encoder_information
get_encoder_information
get_encoder_information, set_encoder_information

5.18.2 Поля

5.18.2.1 Manufacturer

```
char Manufacturer[17]
```

Производитель.

Максимальная длина строки: 16 символов.

5.18.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

5.19 Структура encoder_settings_t

Устарело.

Поля данных

- float MaxOperatingFrequency
  Максимальная частота (кГц).
- float SupplyVoltageMin
  Минимальное напряжение питания (В).
- float SupplyVoltageMax
  Максимальное напряжение питания (В).
- float MaxCurrentConsumption
  Максимальное потребление тока (мА).
- unsigned int PPR
  Количество отсчётов на оборот
- unsigned int EncoderSettings
  Флаги настроек энкодера.

5.19.1 Подробное описание

Устарело.

Настройки энкодера.

См. также

set_encoder_settings
get_encoder_settings
get_encoder_settings, set_encoder_settings

5.19.2 Поля

5.19.2.1 EncoderSettings

```
unsigned int EncoderSettings
```

Флаги настроек энкодера.

5.19.2.2 MaxCurrentConsumption

```
float MaxCurrentConsumption
```

Максимальное потребление тока (мА).

Тип данных: float.

5.19.2.3 MaxOperatingFrequency

```
float MaxOperatingFrequency
```

Максимальная частота (кГц).

Тип данных: float.

5.19.2.4 SupplyVoltageMax

```
float SupplyVoltageMax
```

Максимальное напряжение питания (В).

Тип данных: float.

5.19.2.5 SupplyVoltageMin

```
float SupplyVoltageMin
```

Минимальное напряжение питания (В).

Тип данных: float.

5.20 Структура engine_advanced_setup_t

Настройки EAS.

Поля данных

- unsigned int stepcloseloop_Kw
  Используется только производителем.
- unsigned int stepcloseloop_Kp_low
  Используется только производителем.
- unsigned int stepcloseloop_Kp_high
  Используется только производителем.

5.20.1 Подробное описание

Настройки EAS.

Используется только производителем. Эта структура предназначена для настройки параметров алгоритмов, которые невозможно отнести к стандартным Kp, Ki, Kd и L, R, Km. Эти данные хранятся во flash-памяти контроллера.

См. также

set_engine_advanced_setup
get_engine_advanced_setup
get_engine_advanced_setup, set_engine_advanced_setup

5.20.2 Поля

5.20.2.1 stepcloseloop_Kp_high

```
unsigned int stepcloseloop_Kp_high
```

Используется только производителем.

Обратная связь по позиции в зоне больших скоростей, диапазон [0, 65535], значение по умолчанию 33.

5.20.2.2 stepcloseloop_Kp_low

```
unsigned int stepcloseloop_Kp_low
```

Используется только производителем.

Обратная связь по позиции в зоне малых скоростей, диапазон [0, 65535], значение по умолчанию 1000.

5.20.2.3 stepcloseloop_Kw

```
unsigned int stepcloseloop_Kw
```

Используется только производителем.

Коэффициент смешения реальной и заданной скорости, диапазон [0, 100], значение по умолчанию 50.

5.21 Структура engine_settings_calb_t

Ограничения и настройки движения, связанные с двигателем, с использованием пользовательских единиц.

Поля данных

- unsigned int NomVoltage

  Номинальное напряжение мотора в десятках мВ.
- unsigned int NomCurrent

  Номинальный ток через мотор (в мА).
- float NomSpeed

  Номинальная скорость.
- unsigned int EngineFlags

  Флаги параметров мотора.
- float Antiplay

  Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны.
- unsigned int MicrostepMode

  Флаги параметров микрошагового режима.
- unsigned int StepsPerRev

  Количество полных шагов на оборот (используется только с шаговым двигателем).
- unsigned int PeakCurrent

  Номинальный пиковый ток (в мА).

5.21.1 Подробное описание

Ограничения и настройки движения, связанные с двигателем, с использованием пользовательских единиц.

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_calb
get_engine_settings_calb
get_engine_settings, set_engine_settings

5.21.2 Поля

5.21.2.1 Antiplay

```
float Antiplay
```

Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны.

Используется, если установлен флаг ENGINE_ANTIPLAY.

5.21.2.2 EngineFlags

```
unsigned int EngineFlags
```

Флаги параметров мотора.

5.21.2.3 MicrostepMode

```
unsigned int MicrostepMode
```

Флаги параметров микрошагового режима.

5.21.2.4 NomCurrent

```
unsigned int NomCurrent
```

Номинальный ток через мотор (в мА).

Ток стабилизируется для шаговых и может быть ограничен для DC (если установлен флаг ENGIN↩ E_LIMIT_CURR). Диапазон: 15..8000

5.21.2.5 NomSpeed

```
float NomSpeed
```

Номинальная скорость.

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGIN↩ E_LIMIT_RPM.

5.21.2.6 NomVoltage

```
unsigned int NomVoltage
```

Номинальное напряжение мотора в десятках мВ.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT (используется только с DC-двигателем).

5.21.2.7 PeakCurrent

```
unsigned int PeakCurrent
```

Номинальный пиковый ток (в мА).

Контроллеру разрешается подавать пиковый ток, поддерживая при этом средний ток за 10 секунд ниже номинального, если установлен флаг USE_PEAK_CURRENT.

5.21.2.8 StepsPerRev

```
unsigned int StepsPerRev
```

Количество полных шагов на оборот (используется только с шаговым двигателем).

Диапазон: 1..65535.

5.22 Структура engine_settings_t

Ограничения и настройки движения, связанные с двигателем.

Поля данных

- unsigned int NomVoltage

  Номинальное напряжение мотора в десятках мВ.
- unsigned int NomCurrent

  Номинальный ток через мотор (в мА).
- unsigned int NomSpeed

  Номинальная (максимальная) скорость (в целых шагах/с или rpm для DC- и шагового двигателя в режиме ведущего энкодера).
- unsigned int uNomSpeed

  Микрошаговая часть номинальной скорости мотора (используется только с шаговым двигателем).
- unsigned int EngineFlags

  Флаги параметров мотора.
- int Antiplay

  Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны.
- unsigned int MicrostepMode

  Флаги параметров микрошагового режима.
- unsigned int StepsPerRev

  Количество полных шагов на оборот (используется только с шаговым двигателем).
- unsigned int PeakCurrent

  Номинальный пиковый ток (в мА).

5.22.1 Подробное описание

Ограничения и настройки движения, связанные с двигателем.

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
get_engine_settings, set_engine_settings

5.22.2 Поля

5.22.2.1 Antiplay

```
int Antiplay
```

Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны.

Используется, если установлен флаг ENGINE_ANTIPLAY.

5.22.2.2 EngineFlags

```
unsigned int EngineFlags
```

Флаги параметров мотора.

5.22.2.3 MicrostepMode

```
unsigned int MicrostepMode
```

Флаги параметров микрошагового режима.

5.22.2.4 NomCurrent

```
unsigned int NomCurrent
```

Номинальный ток через мотор (в мА).

Ток стабилизируется для шаговых и может быть ограничен для DC (если установлен флаг ENGIN↩ E_LIMIT_CURR). Диапазон: 15..8000

5.22.2.5 NomSpeed

```
unsigned int NomSpeed
```

Номинальная (максимальная) скорость (в целых шагах/с или rpm для DC- и шагового двигателя в режиме ведущего энкодера).

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGIN↩ E_LIMIT_RPM. Диапазон: 1..100000.

5.22.2.6 NomVoltage

```
unsigned int NomVoltage
```

Номинальное напряжение мотора в десятках мВ.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT (используется только с DC-двигателем).

5.22.2.7 PeakCurrent

```
unsigned int PeakCurrent
```

Номинальный пиковый ток (в мА).

Контроллеру разрешается подавать пиковый ток, поддерживая при этом средний ток за 10 секунд ниже номинального, если установлен флаг USE_PEAK_CURRENT.

5.22.2.8 StepsPerRev

```
unsigned int StepsPerRev
```

Количество полных шагов на оборот (используется только с шаговым двигателем).

Диапазон: 1..65535.

5.22.2.9 uNomSpeed

```
unsigned int uNomSpeed
```

Микрошаговая часть номинальной скорости мотора (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.23 Структура entype_settings_t

Настройки типа мотора и типа силового драйвера.

Поля данных

- unsigned int EngineType

  Флаги, определяющие тип мотора.
- unsigned int DriverType

  Флаги, определяющие тип силового драйвера.

5.23.1 Подробное описание

Настройки типа мотора и типа силового драйвера.

Эта структура содержит настройки типа мотора и типа силового драйвера.

Аргументы

id	идентификатор устройства
EngineType	тип мотора
DriverType	тип силового драйвера

См. также

get_entype_settings, set_entype_settings

5.23.2 Поля

5.23.2.1 DriverType

```
unsigned int DriverType
```

Флаги, определяющие тип силового драйвера.

5.23.2.2 EngineType

```
unsigned int EngineType
```

Флаги, определяющие тип мотора.

5.24 Структура extended_settings_t

Настройки EST.

Поля данных

- unsigned int **Param1**

5.24.1 Подробное описание

Настройки EST.

Эти данные хранятся во flash-памяти контроллера. Эта структура на будущее. В настоящее время не используется.

См. также

set_extended_settings
get_extended_settings
get_extended_settings, set_extended_settings

5.25 Структура extio_settings_t

Настройки EXTIO.

Поля данных

- unsigned int EXTIOSetupFlags

  Флаги настройки работы внешнего ввода/вывода.
- unsigned int EXTIOModeFlags

  Флаги настройки режимов внешнего ввода/вывода.

5.25.1 Подробное описание

Настройки EXTIO.

Эта структура содержит все настройки, определяющие поведение ножки EXTIO. Входные события обрабатываются по фронту. Выходные состояния сигнализируются логическим состоянием. По умолчанию нарастающий фронт считается моментом подачи входного сигнала, а единичное состояние считается активным выходом.

См. также

get_extio_settings
set_extio_settings
get_extio_settings, set_extio_settings

5.25.2 Поля

5.25.2.1 EXTIOModeFlags

```
unsigned int EXTIOModeFlags
```

Флаги настройки режимов внешнего ввода/вывода.

5.25.2.2 EXTIOSetupFlags

```
unsigned int EXTIOSetupFlags
```

Флаги настройки работы внешнего ввода/вывода.

5.26 Структура feedback_settings_t

Настройки обратной связи.

Поля данных

- unsigned int IPS

  Количество отсчётов энкодера на оборот вала.
- unsigned int FeedbackType

  Тип обратной связи.
- unsigned int FeedbackFlags

  Флаги обратной связи.
- unsigned int CountsPerTurn

  Количество отсчётов энкодера на оборот вала.

5.26.1 Подробное описание

Настройки обратной связи.

Эта структура содержит настройки обратной связи.

См. также

get_feedback_settings, set_feedback_settings

5.26.2 Поля

5.26.2.1 CountsPerTurn

```
unsigned int CountsPerTurn
```

Количество отсчётов энкодера на оборот вала.

Диапазон: 1..4294967295. Для использования поля CountsPerTurn нужно записать 0 в поле IPS, иначе будет использоваться значение из поля IPS.

5.26.2.2 FeedbackFlags

```
unsigned int FeedbackFlags
```

Флаги обратной связи.

5.26.2.3 FeedbackType

```
unsigned int FeedbackType
```

Тип обратной связи.

5.26.2.4 IPS

```
unsigned int IPS
```

Количество отсчётов энкодера на оборот вала.

Диапазон: 1..65535. Поле устарело, рекомендуется записывать 0 в IPS и использовать расширенное поле CountsPerTurn. Может потребоваться обновление микропрограммы контроллера до последней версии.

5.27 Структура gear_information_t

Устарело.

Поля данных

- char Manufacturer [17]
  Производитель.
- char PartNumber [25]
  Серия и номер модели.

5.27.1 Подробное описание

Устарело.

Информация о редукторе.

См. также

set_gear_information
get_gear_information
get_gear_information, set_gear_information

5.27.2 Поля

5.27.2.1 Manufacturer

```
char Manufacturer[17]
```

Производитель.

Максимальная длина строки: 16 символов.

5.27.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

5.28 Структура gear_settings_t

Устарело.

Поля данных

- float ReductionIn

  Входной коэффициент редуктора.
- float ReductionOut

  Выходной коэффициент редуктора.
- float RatedInputTorque

  *Максимальный крутящий момент (Н * м).*
- float RatedInputSpeed

  Максимальная скорость на входном валу редуктора (об/мин).
- float MaxOutputBacklash

  Выходной люфт редуктора (градус).
- float InputInertia

  *Эквивалентная входная инерция редуктора(г * см2).*
- float Efficiency

  КПД редуктора (%).

5.28.1 Подробное описание

Устарело.

Настройки редуктора.

См. также

set_gear_settings
get_gear_settings
get_gear_settings, set_gear_settings

5.28.2 Поля

5.28.2.1 Efficiency

```
float Efficiency
```

КПД редуктора (%).

Тип данных: float.

5.28.2.2 InputInertia

```
float InputInertia
```

Эквивалентная входная инерция редуктора(г ∗ см2).

Тип данных: float.

5.28.2.3 MaxOutputBacklash

```
float MaxOutputBacklash
```

Выходной люфт редуктора (градус).

Тип данных: float.

5.28.2.4 RatedInputSpeed

```
float RatedInputSpeed
```

Максимальная скорость на входном валу редуктора (об/мин).

Тип данных: float.

5.28.2.5 RatedInputTorque

```
float RatedInputTorque
```

Максимальный крутящий момент (Н ∗ м).

Тип данных: float.

5.28.2.6 ReductionIn

```
float ReductionIn
```

Входной коэффициент редуктора.

(Выход = (ReductionOut/ReductionIn) * вход) Тип данных: float.

5.28.2.7 ReductionOut

```
float ReductionOut
```

Выходной коэффициент редуктора.

(Выход = (ReductionOut/ReductionIn) * вход) Тип данных: float.

5.29 Структура get_position_calb_t

Данные о позиции.

Поля данных

- float Position
  Позиция двигателя.
- long_t EncPosition
  Позиция энкодера.

5.29.1 Подробное описание

Данные о позиции.

Структура содержит значение положения в пользовательских единицах для шагового двигателя и в шагах энкодера всех двигателей.

См. также

get_position

5.29.2 Поля

5.29.2.1 EncPosition

```
long_t EncPosition
```

Позиция энкодера.

5.29.2.2 Position

```
float Position
```

Позиция двигателя.

Корректируется таблицей.

5.30 Структура get_position_t

Данные о позиции.

Поля данных

- int Position
  Позиция в основных шагах двигателя
- int uPosition
  Позиция в микрошагах (используется только с шаговыми двигателями).
- long_t EncPosition
  Позиция энкодера.

5.30.1 Подробное описание

Данные о позиции.

Структура содержит значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.

См. также

get_position

5.30.2 Поля

5.30.2.1 EncPosition

```
long_t EncPosition
```

Позиция энкодера.

5.30.2.2 uPosition

```
int uPosition
```

Позиция в микрошагах (используется только с шаговыми двигателями).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.31 Структура globally_unique_identifier_t

Глобальный уникальный идентификатор.

Поля данных

- unsigned int UniqueID0
  Уникальный ID 0.
- unsigned int UniqueID1
  Уникальный ID 1.
- unsigned int UniqueID2
  Уникальный ID 2.
- unsigned int UniqueID3
  Уникальный ID 3.

5.31.1 Подробное описание

Глобальный уникальный идентификатор.

Используется только производителем.

См. также

get_globally_unique_identifier

5.31.2 Поля

5.31.2.1 UniqueID0

```
unsigned int UniqueID0
```

Уникальный ID 0.

5.31.2.2 UniqueID1

```
unsigned int UniqueID1
```

Уникальный ID 1.

5.31.2.3 UniqueID2

```
unsigned int UniqueID2
```

Уникальный ID 2.

5.31.2.4 UniqueID3

```
unsigned int UniqueID3
```

Уникальный ID 3.

5.32 Структура hallsensor_information_t

Устарело.

Поля данных

- char Manufacturer [17]

  Производитель.
- char PartNumber [25]

  Серия и номер модели.

5.32.1 Подробное описание

Устарело.

Информация о датчиках Холла.

См. также

set_hallsensor_information
get_hallsensor_information
get_hallsensor_information, set_hallsensor_information

5.32.2 Поля

5.32.2.1 Manufacturer

```
char Manufacturer[17]
```

Производитель.

Максимальная длина строки: 16 символов.

5.32.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

5.33 Структура hallsensor_settings_t

Устарело.

Поля данных

- float MaxOperatingFrequency
  Максимальная частота (кГц).
- float SupplyVoltageMin
  Минимальное напряжение питания (В).
- float SupplyVoltageMax
  Максимальное напряжение питания (В).
- float MaxCurrentConsumption
  Максимальное потребление тока (мА).
- unsigned int PPR
  Количество отсчётов на оборот

5.33.1 Подробное описание

Устарело.

Настройки датчиков Холла.

См. также

set_hallsensor_settings
get_hallsensor_settings
get_hallsensor_settings, set_hallsensor_settings

5.33.2 Поля

5.33.2.1 MaxCurrentConsumption

```
float MaxCurrentConsumption
```

Максимальное потребление тока (мА).

Тип данных: float.

5.33.2.2 MaxOperatingFrequency

```
float MaxOperatingFrequency
```

Максимальная частота (кГц).

Тип данных: float.

5.33.2.3 SupplyVoltageMax

```
float SupplyVoltageMax
```

Максимальное напряжение питания (В).

Тип данных: float.

5.33.2.4 SupplyVoltageMin

```
float SupplyVoltageMin
```

Минимальное напряжение питания (В).

Тип данных: float.

5.34 Структура home_settings_calb_t

Настройки калибровки позиции с использованием пользовательских единиц.

Поля данных

- float FastHome

  Скорость первого движения.
- float SlowHome

  Скорость второго движения.
- float HomeDelta

  Расстояние отхода от точки останова.
- unsigned int HomeFlags

  Флаги настроек команды home.

5.34.1 Подробное описание

Настройки калибровки позиции с использованием пользовательских единиц.

Эта структура содержит настройки, использующиеся при калибровке позиции.

См. также

get_home_settings_calb
set_home_settings_calb
command_home
get_home_settings, set_home_settings

5.34.2 Поля

5.34.2.1 FastHome

```
float FastHome
```

Скорость первого движения.

5.34.2.2 HomeDelta

```
float HomeDelta
```

Расстояние отхода от точки останова.

5.34.2.3 HomeFlags

```
unsigned int HomeFlags
```

Флаги настроек команды home.

5.34.2.4 SlowHome

```
float SlowHome
```

Скорость второго движения.

5.35 Структура home_settings_t

Настройки калибровки позиции.

Поля данных

- unsigned int FastHome

  Скорость первого движения (в полных шагах).
- unsigned int uFastHome

  Дробная часть скорости первого движения в микрошагах (используется только с шаговым двигателем).
- unsigned int SlowHome

  Скорость второго движения (в полных шагах).
- unsigned int uSlowHome

  Дробная часть скорости второго движения в микрошагах (используется только с шаговым двигателем).
- int HomeDelta

  Расстояние отхода от точки останова (в полных шагах).
- int uHomeDelta

  Дробная часть расстояния отхода от точки останова в микрошагах (используется только с шаговым двигателем).
- unsigned int HomeFlags

  Флаги настроек команды home.

5.35.1 Подробное описание

Настройки калибровки позиции.

Эта структура содержит настройки, использующиеся при калибровке позиции.

См. также

get_home_settings
set_home_settings
command_home
get_home_settings, set_home_settings

5.35.2 Поля

5.35.2.1 FastHome

```
unsigned int FastHome
```

Скорость первого движения (в полных шагах).

Диапазон: 0..100000

5.35.2.2 HomeDelta

```
int HomeDelta
```

Расстояние отхода от точки останова (в полных шагах).

5.35.2.3 HomeFlags

```
unsigned int HomeFlags
```

Флаги настроек команды home.

5.35.2.4 SlowHome

```
unsigned int SlowHome
```

Скорость второго движения (в полных шагах).

Диапазон: 0..100000.

5.35.2.5 uFastHome

```
unsigned int uFastHome
```

Дробная часть скорости первого движения в микрошагах (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.35.2.6 uHomeDelta

```
int uHomeDelta
```

Дробная часть расстояния отхода от точки останова в микрошагах (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.35.2.7 uSlowHome

```
unsigned int uSlowHome
```

Дробная часть скорости второго движения в микрошагах (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.36 Структура init_random_t

Случайный ключ.

Поля данных

- uint8_t key [16]
  Случайный ключ.

5.36.1 Подробное описание

Случайный ключ.

Используется только производителем. Структура которая содержит случайный ключ, использующийся для шифрования содержимого команд WKEY и SSER.

См. также

get_init_random

5.36.2 Поля

5.36.2.1 key

```
uint8_t key[16]
```

Случайный ключ.

5.37 Структура joystick_settings_t

Настройки джойстика.

Поля данных

- unsigned int JoyLowEnd

  Значение в шагах джойстика, соответствующее нижней границе диапазона отклонения устройства.
- unsigned int JoyCenter

  Значение в шагах джойстика, соответствующее неотклонённому устройству.
- unsigned int JoyHighEnd

  Значение в шагах джойстика, соответствующее верхней границе диапазона отклонения устройства.
- unsigned int ExpFactor

  Фактор экспоненциальной нелинейности отклика джойстика.
- unsigned int DeadZone

  Отклонение от среднего положения, которое не вызывает начала движения (в десятых долях процента).
- unsigned int JoyFlags

  Флаги джойстика.

5.37.1 Подробное описание

Настройки джойстика.

Команда чтения настроек и калибровки джойстика. При отклонении джойстика более чем на Dead↩ Zone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость (при этом постоянно выполняется команда "soft stop"), а 100% отклонения соответствует MaxSpeed i, где i=0, если предыдущим использованием этого режима не было выбрано другое i. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. Первая скорость в списке не должна быть нулевой. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Зависимость между отклонением и скоростью экспоненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность и точность.

См. также

set_joystick_settings
get_joystick_settings
get_joystick_settings, set_joystick_settings

5.37.2 Поля

5.37.2.1 DeadZone

```
unsigned int DeadZone
```

Отклонение от среднего положения, которое не вызывает начала движения (в десятых долях процента).

Максимальное мёртвое отклонение +-25.5%, что составляет половину рабочего диапазона джойстика.

5.37.2.2 ExpFactor

```
unsigned int ExpFactor
```

Фактор экспоненциальной нелинейности отклика джойстика.

5.37.2.3 JoyCenter

```
unsigned int JoyCenter
```

Значение в шагах джойстика, соответствующее неотклонённому устройству.

Должно лежать в диапазоне 0..10000.

5.37.2.4 JoyFlags

```
unsigned int JoyFlags
```

Флаги джойстика.

5.37.2.5 JoyHighEnd

```
unsigned int JoyHighEnd
```

Значение в шагах джойстика, соответствующее верхней границе диапазона отклонения устройства.

Должно лежать в диапазоне 0..10000.

5.37.2.6 JoyLowEnd

```
unsigned int JoyLowEnd
```

Значение в шагах джойстика, соответствующее нижней границе диапазона отклонения устройства.

Должно лежать в диапазоне 0..10000.

5.38 Структура measurements_t

Структура содержит последовательность измеренных параметров движения оси – скоростей и ошибок по позиции.

Поля данных

- int Speed [25]

  Последовательность измеренных скоростей (в отсчётах энкодера/сек или микрошагах/сек, в зависимости от типа двигателя и режима управления)
- int Error [25]

  Последовательность измеренных ошибок позиции (в отсчётах энкодера или микрошагах, в зависимости от типа двигателя и режима управления)
- unsigned int Length

  Фактическая длина последовательности.

5.38.1 Подробное описание

Структура содержит последовательность измеренных параметров движения оси – скоростей и ошибок по позиции.

Шаг по времени между двумя последовательными измерениями - 1 мс.

См. также

measurements
get_measurements

5.38.2 Поля

5.38.2.1 Length

```
unsigned int Length
```

Фактическая длина последовательности.

Значения, содержащиеся в ячейках Length, Length + 1, ...24, не должны интерпретироваться в качестве результатов измерений.

5.39 Структура motor_information_t

Устарело.

Поля данных

- char Manufacturer [17]
  Производитель.
- char PartNumber [25]
  Серия и номер модели.

5.39.1 Подробное описание

Устарело.

Информация о двигателе.

См. также

set_motor_information
get_motor_information
get_motor_information, set_motor_information

5.39.2 Поля

5.39.2.1 Manufacturer

```
char Manufacturer[17]
```

Производитель.

Максимальная длина строки: 16 символов.

5.39.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

5.40 Структура motor_settings_t

Устарело.

Поля данных

- unsigned int MotorType

  Флаги типа двигателя.
- unsigned int ReservedField

  Зарезервировано
- unsigned int Poles

  Количество пар полюсов у DC- или BLDC-двигателя или количество шагов на оборот для шагового двигателя.
- unsigned int Phases

  Количество фаз у BLDC-двигателя.
- float NominalVoltage

  Номинальное напряжение на обмотке (В).
- float NominalCurrent

  Максимальный постоянный ток в обмотке для DC- и BLDC-двигателей, номинальный ток в обмотке для шаговых двигателей (А).
- float NominalSpeed

  Не используется.
- float NominalTorque

  *Номинальный крутящий момент (мН * м).*
- float NominalPower

  Номинальная мощность(Вт).
- float WindingResistance

  Сопротивление обмотки DC-двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC-двигателя (Ом).
- float WindingInductance

  Индуктивность обмотки DC-двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC-двигателя (мГн).
- float RotorInertia

  *Инерция ротора (г * см2).*
- float StallTorque

  *Крутящий момент удержания позиции для шагового двигателя или крутящий момент при неподвижном роторе для других типов двигателей (мН * м).*
- float DetentTorque

  *Момент удержания позиции с незапитанными обмотками (мН * м).*
- float TorqueConstant

  *Константа крутящего момента, определяющая коэффициент пропорциональности максимального момента силы ротора от протекающего в обмотке тока (мН * м/А).*
- float SpeedConstant

  Константа скорости, определяющая значение или амплитуду напряжения наведённой индукции при вращении ротора DC- или BLDC-двигателя (об/мин / В) или шагового двигателя (шаг/с / В).
- float SpeedTorqueGradient

  *Градиент крутящего момента (об/мин / мН * м).*
- float MechanicalTimeConstant

  Механическая постоянная времени (мс).
- float MaxSpeed

  Максимальная разрешённая скорость для шаговых двигателей (шаг/с) или для DC- и BLDC-двигателей (об/мин).
- float MaxCurrent

  Максимальный ток в обмотке (А).
- float MaxCurrentTime

Безопасная длительность максимального тока в обмотке (мс).

- float NoLoadCurrent

  Ток потребления в холостом режиме (А).

- float NoLoadSpeed

  Скорость в холостом режиме (об/мин).

5.40.1 Подробное описание

Устарело.

Физический характеристики и ограничения мотора.

См. также

set_motor_settings
get_motor_settings
get_motor_settings, set_motor_settings

5.40.2 Поля

5.40.2.1 DetentTorque

```
float DetentTorque
```

Момент удержания позиции с незапитанными обмотками (мН * м).

Тип данных: float.

5.40.2.2 MaxCurrent

```
float MaxCurrent
```

Максимальный ток в обмотке (А).

Тип данных: float.

5.40.2.3 MaxCurrentTime

```
float MaxCurrentTime
```

Безопасная длительность максимального тока в обмотке (мс).

Тип данных: float.

5.40.2.4 MaxSpeed

```
float MaxSpeed
```

Максимальная разрешённая скорость для шаговых двигателей (шаг/с) или для DC- и BLDC-двигателей (об/мин).

Тип данных: float.

5.40.2.5 MechanicalTimeConstant

```
float MechanicalTimeConstant
```

Механическая постоянная времени (мс).

Тип данных: float.

5.40.2.6 MotorType

```
unsigned int MotorType
```

Флаги типа двигателя.

5.40.2.7 NoLoadCurrent

```
float NoLoadCurrent
```

Ток потребления в холостом режиме (A).

Применяется для DC- и BLDC-двигателей. Тип данных: float.

5.40.2.8 NoLoadSpeed

```
float NoLoadSpeed
```

Скорость в холостом режиме (об/мин).

Применяется для DC- и BLDC-двигателей. Тип данных: float.

5.40.2.9 NominalCurrent

```
float NominalCurrent
```

Максимальный постоянный ток в обмотке для DC- и BLDC-двигателей, номинальный ток в обмотке для шаговых двигателей (A).

Тип данных: float.

5.40.2.10 NominalPower

```
float NominalPower
```

Номинальная мощность(Вт).

Применяется для DC- и BLDC-двигателей. Тип данных: float.

5.40.2.11 NominalSpeed

```
float NominalSpeed
```

Не используется.

Номинальная скорость (об/мин). Применяется для DC- и BLDC-двигателей. Тип данных: float.

5.40.2.12 NominalTorque

```
float NominalTorque
```

Номинальный крутящий момент (мН $*$ м).

Применяется для DC- и BLDC-двигателей. Тип данных: float.

5.40.2.13 NominalVoltage

```
float NominalVoltage
```

Номинальное напряжение на обмотке (В).

Тип данных: float.

5.40.2.14 Phases

```
unsigned int Phases
```

Количество фаз у BLDC-двигателя.

5.40.2.15 Poles

```
unsigned int Poles
```

Количество пар полюсов у DC- или BLDC-двигателя или количество шагов на оборот для шагового двигателя.

5.40.2.16 RotorInertia

```
float RotorInertia
```

Инерция ротора (г ∗ см2).

Тип данных: float.

5.40.2.17 SpeedConstant

```
float SpeedConstant
```

Константа скорости, определяющая значение или амплитуду напряжения наведённой индукции при вращении ротора DC- или BLDC-двигателя (об/мин / В) или шагового двигателя (шаг/с / В).

Тип данных: float.

5.40.2.18 SpeedTorqueGradient

```
float SpeedTorqueGradient
```

Градиент крутящего момента (об/мин / мН ∗ м).

Тип данных: float.

5.40.2.19 StallTorque

```
float StallTorque
```

Крутящий момент удержания позиции для шагового двигателя или крутящий момент при неподвижном роторе для других типов двигателей (мН ∗ м).

Тип данных: float.

5.40.2.20 TorqueConstant

```
float TorqueConstant
```

Константа крутящего момента, определяющая коэффициент пропорциональности максимального момента силы ротора от протекающего в обмотке тока (мН ∗ м/А).

Используется в основном для DC-двигателей. Тип данных: float.

5.40.2.21 WindingInductance

```
float WindingInductance
```

Индуктивность обмотки DC-двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC-двигателя (мГн).

Тип данных: float.

5.40.2.22 WindingResistance

```
float WindingResistance
```

Сопротивление обмотки DC-двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC-двигателя (Ом).

Тип данных: float.

5.41 Структура move_settings_calb_t

Настройки движения с использованием пользовательских единиц.

Поля данных

- float Speed
  Скорость
- float Accel
  Ускорение
- float Decel
  Торможение
- float AntiplaySpeed
  Скорость в режиме антилюфта
- unsigned int MoveFlags
  Флаги параметров движения.

5.41.1 Подробное описание

Настройки движения с использованием пользовательских единиц.

См. также

set_move_settings_calb
get_move_settings_calb
get_move_settings, set_move_settings

5.41.2 Поля

5.41.2.1 Accel

```
float Accel
```

Ускорение

- для шагового двигателя без энкодера — в шагах в секунду^2.
- при использовании энкодера (включая DC/BLDC) — в оборотах в минуту (RPM).

5.41.2.2 AntiplaySpeed

```
float AntiplaySpeed
```

Скорость в режиме антилюфта

- для шагового двигателя без энкодера — в шагах в секунду.
- при использовании энкодера (включая DC/BLDC) — в оборотах в минуту (RPM).

5.41.2.3 Decel

```
float Decel
```

Торможение

- для шагового двигателя без энкодера — в шагах в секунду^2.
- при использовании энкодера (включая DC/BLDC) — в оборотах в минуту (RPM).

5.41.2.4 MoveFlags

```
unsigned int MoveFlags
```

Флаги параметров движения.

5.41.2.5 Speed

```
float Speed
```

Скорость

- для шагового двигателя без энкодера — в шагах в секунду.
- при использовании энкодера (включая DC/BLDC) — в оборотах в минуту (RPM).

5.42 Структура move_settings_t

Настройки движения.

Поля данных

- unsigned int Speed

  Заданная скорость (для ШД: шагов/с, для DC: rpm).
- unsigned int uSpeed

  Заданная скорость в единицах деления микрошага в секунду.
- unsigned int Accel

  Ускорение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC).
- unsigned int Decel

  Торможение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC).
- unsigned int AntiplaySpeed

  Скорость в режиме антилюфта, заданная в целых шагах/с (ШД) или в оборотах/с(DC).
- unsigned int uAntiplaySpeed

  Скорость в режиме антилюфта, выраженная в микрошагах в секунду.
- unsigned int MoveFlags

  Флаги параметров движения.

5.42.1 Подробное описание

Настройки движения.

См. также

set_move_settings
get_move_settings
get_move_settings, set_move_settings

5.42.2 Поля

5.42.2.1 Accel

```
unsigned int Accel
```

Ускорение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC).

Диапазон: 1..65535.

5.42.2.2 AntiplaySpeed

```
unsigned int AntiplaySpeed
```

Скорость в режиме антилюфта, заданная в целых шагах/с (ШД) или в оборотах/с(DC).

Диапазон: 0..100000.

5.42.2.3 Decel

```
unsigned int Decel
```

Торможение, заданное в шагах в секунду^2 (ШД) или в оборотах в минуту за секунду (DC).

Диапазон: 1..65535.

5.42.2.4 MoveFlags

```
unsigned int MoveFlags
```

Флаги параметров движения.

5.42.2.5 Speed

```
unsigned int Speed
```

Заданная скорость (для ШД: шагов/с, для DC: rpm).

Диапазон: 0..100000.

5.42.2.6 uAntiplaySpeed

```
unsigned int uAntiplaySpeed
```

Скорость в режиме антилюфта, выраженная в микрошагах в секунду.

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.

5.42.2.7 uSpeed

```
unsigned int uSpeed
```

Заданная скорость в единицах деления микрошага в секунду.

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.

5.43 Структура network_settings_t

Настройки сети.

Поля данных

- unsigned int DHCPEnabled

  Определяет способ получения IP-адреса каналов.
- unsigned int IPv4Address [4]

  IP-адрес устройства в формате x.x.x.x.
- unsigned int SubnetMask [4]

  Маска подсети в формате x.x.x.x.
- unsigned int DefaultGateway [4]

  Шлюз сети по умолчанию в формате x.x.x.x.

5.43.1 Подробное описание

Настройки сети.

Используется только производителем. Эта структура содержит настройки сети.

См. также

get_network_settings
set_network_settings
get_network_settings, set_network_settings

5.43.2 Поля

5.43.2.1 DefaultGateway

```
unsigned int DefaultGateway[4]
```

Шлюз сети по умолчанию в формате x.x.x.x.

5.43.2.2 DHCPEnabled

```
unsigned int DHCPEnabled
```

Определяет способ получения IP-адреса каналов.

Может принимать значения: 0 — статически, 1 — через DHCP

5.43.2.3 IPv4Address

```
unsigned int IPv4Address[4]
```

IP-адрес устройства в формате x.x.x.x.

5.43.2.4 SubnetMask

```
unsigned int SubnetMask[4]
```

Маска подсети в формате x.x.x.x.

5.44 Структура nonvolatile_memory_t

Пользовательские данные для сохранения во FRAM.

Поля данных

- unsigned int UserData [7]
  Пользовательские данные.

5.44.1 Подробное описание

Пользовательские данные для сохранения во FRAM.

См. также

get_nonvolatile_memory, set_nonvolatile_memory

5.44.2 Поля

5.44.2.1 UserData

```
unsigned int UserData[7]
```

Пользовательские данные.

Могут быть установлены пользователем для его удобства. Каждый элемент массива хранит только 32 бита пользовательских данных. Это важно на системах где тип int содержит больше чем 4 байта. Например это все системы amd64.

5.45 Структура password_settings_t

Пароль.

Поля данных

- char UserPassword [21]

  Строчка-пароль для доступа к веб-странице, который пользователь может поменять с помощью USB команды или на веб-странице.

5.45.1 Подробное описание

Пароль.

Используется только производителем. Эта структура содержит пароль к веб-странице.

См. также

get_password_settings
set_password_settings
get_password_settings, set_password_settings

5.45.2 Поля

5.45.2.1 UserPassword

```
char UserPassword[21]
```

Строчка-пароль для доступа к веб-странице, который пользователь может поменять с помощью USB команды или на веб-странице.

5.46 Структура pid_settings_t

Настройки ПИД.

Поля данных

- unsigned int KpU

  Пропорциональный коэффициент ПИД контура по напряжению
- unsigned int KiU

  Интегральный коэффициент ПИД контура по напряжению
- unsigned int KdU

  Дифференциальный коэффициент ПИД контура по напряжению
- float Kpf

  Пропорциональный коэффициент ПИД контура по позиции для BLDC.
- float Kif

  Интегральный коэффициент ПИД контура по позиции для BLDC.
- float Kdf

  Дифференциальный коэффициент ПИД контура по позиции для BLDC.

5.46.1 Подробное описание

Настройки ПИД.

Эта структура содержит коэффициенты для ПИД регулятора. Они определяют работу ПИД контура напряжения. Эти коэффициенты хранятся во flash-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
get_pid_settings, set_pid_settings

5.47 Структура power_settings_t

Настройки питания шагового мотора.

Поля данных

- unsigned int HoldCurrent

  Ток мотора в режиме удержания, в процентах от номинального.
- unsigned int CurrReductDelay

  Время в мс от перехода в состояние STOP до уменьшения тока.
- unsigned int PowerOffDelay

  Время в с от перехода в состояние STOP до отключения питания мотора.
- unsigned int CurrentSetTime

  Время в мс, требуемое для набора номинального тока от 0% до 100%.
- unsigned int PowerFlags

  Флаги параметров питания шагового мотора.

5.47.1 Подробное описание

Настройки питания шагового мотора.

См. также

set_move_settings
get_move_settings
get_power_settings, set_power_settings

5.47.2 Поля

5.47.2.1 CurrentSetTime

```
unsigned int CurrentSetTime
```

Время в мс, требуемое для набора номинального тока от 0% до 100%.

5.47.2.2 CurrReductDelay

```
unsigned int CurrReductDelay
```

Время в мс от перехода в состояние STOP до уменьшения тока.

5.47.2.3 HoldCurrent

```
unsigned int HoldCurrent
```

Ток мотора в режиме удержания, в процентах от номинального.

Диапазон: 0..100.

5.47.2.4 PowerFlags

```
unsigned int PowerFlags
```

Флаги параметров питания шагового мотора.

5.47.2.5 PowerOffDelay

```
unsigned int PowerOffDelay
```

Время в с от перехода в состояние STOP до отключения питания мотора.

5.48 Структура secure_settings_t

Структура содержит параметры защиты: критические значения электрических характеристик, флаги управления алгоритмами защиты.

Поля данных

- unsigned int LowUpwrOff

  Нижний порог напряжения на силовой части для выключения, десятки мВ.
- unsigned int CriticalIpwr

  Максимальный ток силовой части, вызывающий состояние ALARM, в мА.
- unsigned int CriticalUpwr

  Максимальное напряжение на силовой части, вызывающее состояние ALARM, десятки мВ.
- unsigned int CriticalT

  Максимальная температура контроллера, вызывающая состояние ALARM, в десятых долях градуса Цельсия.
- unsigned int CriticalIusb

  Устарело.
- unsigned int CriticalUusb

  Устарело.
- unsigned int MinimumUusb

  Устарело.
- unsigned int Flags

  Флаги критических параметров.

5.48.1 Подробное описание

Структура содержит параметры защиты: критические значения электрических характеристик, флаги управления алгоритмами защиты.

См. также

get_secure_settings
set_secure_settings
get_secure_settings, set_secure_settings

5.48.2 Поля

5.48.2.1 CriticalIpwr

```
unsigned int CriticalIpwr
```

Максимальный ток силовой части, вызывающий состояние ALARM, в мА.

5.48.2.2 CriticalIusb

```
unsigned int CriticalIusb
```

Устарело.

Максимальный ток USB, вызывающий состояние ALARM, в мА.

5.48.2.3 CriticalUpwr

```
unsigned int CriticalUpwr
```

Максимальное напряжение на силовой части, вызывающее состояние ALARM, десятки мВ.

5.48.2.4 CriticalUusb

```
unsigned int CriticalUusb
```

Устарело.

Максимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.

5.48.2.5 Flags

```
unsigned int Flags
```

Флаги критических параметров.

5.48.2.6 LowUpwrOff

```
unsigned int LowUpwrOff
```

Нижний порог напряжения на силовой части для выключения, десятки мВ.

5.48.2.7 MinimumUusb

```
unsigned int MinimumUusb
```

Устарело.

Минимальное напряжение на USB, вызывающее состояние ALARM, десятки мВ.

5.49 Структура serial_number_t

Структура с серийным номером и версией железа.

Поля данных

- unsigned int SN

  Новый серийный номер платы.
- uint8_t Key [32]

  Ключ защиты для установки серийного номера (256 бит).
- unsigned int Major

  Основной номер версии железа.
- unsigned int Minor

  Второстепенный номер версии железа.
- unsigned int Release

  Номер правок этой версии железа.

5.49.1 Подробное описание

Структура с серийным номером и версией железа.

Вместе с новым серийным номером и версией железа передаётся "Ключ", только при совпадении которого происходит изменение и сохранение. Функция используется только производителем. Используется только загрузчиком.

См. также

set_serial_number

5.49.2 Поля

5.49.2.1 Key

```
uint8_t Key[32]
```

Ключ защиты для установки серийного номера (256 бит).

5.49.2.2 Major

```
unsigned int Major
```

Основной номер версии железа.

5.49.2.3 Minor

```
unsigned int Minor
```

Второстепенный номер версии железа.

5.49.2.4 Release

```
unsigned int Release
```

Номер правок этой версии железа.

5.49.2.5 SN

```
unsigned int SN
```

Новый серийный номер платы.

5.50 Структура set_position_calb_t

Данные о позиции с использованием пользовательских единиц.

Поля данных

- float Position
  Позиция двигателя.
- long_t EncPosition
  Позиция энкодера.
- unsigned int PosFlags
  Флаги установки положения.

5.50.1 Подробное описание

Данные о позиции с использованием пользовательских единиц.

Структура содержит значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.

См. также

set_position

5.50.2 Поля

5.50.2.1 EncPosition

```
long_t EncPosition
```

Позиция энкодера.

5.50.2.2 PosFlags

```
unsigned int PosFlags
```

Флаги установки положения.

5.50.2.3 Position

```
float Position
```

Позиция двигателя.

5.51 Структура set_position_t

Данные о позиции.

Поля данных

- int Position
  Позиция в основных шагах двигателя
- int uPosition
  Позиция в микрошагах (используется только с шаговыми двигателями).
- long_t EncPosition
  Позиция энкодера.
- unsigned int PosFlags
  Флаги установки положения.

5.51.1 Подробное описание

Данные о позиции.

Структура содержит значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.

См. также

set_position

5.51.2 Поля

5.51.2.1 EncPosition

```
long_t EncPosition
```

Позиция энкодера.

5.51.2.2 PosFlags

```
unsigned int PosFlags
```

Флаги установки положения.

5.51.2.3 uPosition

```
int uPosition
```

Позиция в микрошагах (используется только с шаговыми двигателями).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.52 Структура stage_information_t

Устарело.

Поля данных

- char Manufacturer [17]

  Производитель.
- char PartNumber [25]

  Серия и номер модели.

5.52.1 Подробное описание

Устарело.

Информация о позиционере.

См. также

set_stage_information
get_stage_information
get_stage_information, set_stage_information

5.52.2 Поля

5.52.2.1 Manufacturer

```
char Manufacturer[17]
```

Производитель.

Максимальная длина строки: 16 символов.

5.52.2.2 PartNumber

```
char PartNumber[25]
```

Серия и номер модели.

Максимальная длина строки: 24 символа.

5.53 Структура stage_name_t

Пользовательское имя подвижки.

Поля данных

- char PositionerName [17]

  Пользовательское имя подвижки.

5.53.1 Подробное описание

Пользовательское имя подвижки.

См. также

get_stage_name, set_stage_name

5.53.2 Поля

5.53.2.1 PositionerName

```
char PositionerName[17]
```

Пользовательское имя подвижки.

Может быть установлено пользователем для его удобства. Максимальная длина строки: 16 символов.

5.54 Структура stage_settings_t

Устарело.

Поля данных

- float LeadScrewPitch

  Шаг ходового винта в мм.
- char Units [9]

  Единицы измерения расстояния, используемые в полях MaxSpeed и TravelRange (шаги, градусы, мм, ...), Максимальная длина строки: 8 символов.
- float MaxSpeed

  Максимальная скорость (Units/c).
- float TravelRange

  Диапазон перемещения (Units).
- float SupplyVoltageMin

  Минимальное напряжение питания (В).
- float SupplyVoltageMax

  Максимальное напряжение питания (В).
- float MaxCurrentConsumption

  Максимальный ток потребления (А).
- float HorizontalLoadCapacity

  Горизонтальная грузоподъемность (кг).
- float VerticalLoadCapacity

  Вертикальная грузоподъемность (кг).

5.54.1 Подробное описание

Устарело.

Настройки позиционера.

См. также

set_stage_settings
get_stage_settings
get_stage_settings, set_stage_settings

5.54.2 Поля

5.54.2.1 HorizontalLoadCapacity

```
float HorizontalLoadCapacity
```

Горизонтальная грузоподъемность (кг).

Тип данных: float.

5.54.2.2 LeadScrewPitch

```
float LeadScrewPitch
```

Шаг ходового винта в мм.

Тип данных: float.

5.54.2.3 MaxCurrentConsumption

```
float MaxCurrentConsumption
```

Максимальный ток потребления (A).

Тип данных: float.

5.54.2.4 MaxSpeed

```
float MaxSpeed
```

Максимальная скорость (Units/c).

Тип данных: float.

5.54.2.5 SupplyVoltageMax

```
float SupplyVoltageMax
```

Максимальное напряжение питания (В).

Тип данных: float.

5.54.2.6 SupplyVoltageMin

```
float SupplyVoltageMin
```

Минимальное напряжение питания (В).

Тип данных: float.

5.54.2.7 TravelRange

```
float TravelRange
```

Диапазон перемещения (Units).

Тип данных: float.

5.54.2.8 Units

```
char Units[9]
```

Единицы измерения расстояния, используемые в полях MaxSpeed и TravelRange (шаги, градусы, мм, ...), Максимальная длина строки: 8 символов.

5.54.2.9 VerticalLoadCapacity

```
float VerticalLoadCapacity
```

Вертикальная грузоподъемность (кг).

Тип данных: float.

5.55 Структура status_calb_t

Состояние устройства с использованием пользовательских единиц.

Поля данных

- unsigned int MoveSts

  Флаги состояния движения.
- unsigned int MvCmdSts

  Состояние команды движения.
- unsigned int PWRSts

  Флаги состояния питания шагового мотора.
- unsigned int EncSts

  Состояние энкодера.
- unsigned int WindSts

  Состояние обмоток.
- float CurPosition

  Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.
- long_t EncPosition

  Текущая позиция по данным с энкодера в импульсах энкодера используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзования.
- float CurSpeed

  Текущая скорость.
- int Ipwr

  Ток потребления силовой части, мА.
- int Upwr

  Напряжение на силовой части, десятки мВ.
- int Iusb

  Ток потребления по USB, мА.
- int Uusb

  Напряжение на USB, десятки мВ.
- int CurT

  Температура процессора в десятых долях градусов Цельсия.
- unsigned int Flags

  Флаги состояния.
- unsigned int GPIOFlags

  Флаги состояния GPIO входов.
- unsigned int CmdBufFreeSpace

  Данное поле служебное.

5.55.1 Подробное описание

Состояние устройства с использованием пользовательских единиц.

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

См. также

get_status_impl

5.55.2 Поля

5.55.2.1 CmdBufFreeSpace

```
unsigned int CmdBufFreeSpace
```

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

5.55.2.2 CurPosition

```
float CurPosition
```

Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.

В случае работы с DC-мотором в этом поле находится текущая позиция по данным с энкодера, в случае работы с ШД-мотором в режиме, когда первичными являются импульсы, подаваемые на мотор. Корректируется таблицей.

5.55.2.3 CurSpeed

```
float CurSpeed
```

Текущая скорость.

5.55.2.4 CurT

```
int CurT
```

Температура процессора в десятых долях градусов Цельсия.

5.55.2.5 EncPosition

```
long_t EncPosition
```

Текущая позиция по данным с энкодера в импульсах энкодера используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзования.

5.55.2.6 EncSts

```
unsigned int EncSts
```

Состояние энкодера.

5.55.2.7 Flags

```
unsigned int Flags
```

Флаги состояния.

5.55.2.8 GPIOFlags

```
unsigned int GPIOFlags
```

Флаги состояния GPIO входов.

5.55.2.9 Ipwr

```
int Ipwr
```

Ток потребления силовой части, мА.

5.55.2.10 Iusb

```
int Iusb
```

Ток потребления по USB, мА.

5.55.2.11 MoveSts

```
unsigned int MoveSts
```

Флаги состояния движения.

5.55.2.12 MvCmdSts

```
unsigned int MvCmdSts
```

Состояние команды движения.

5.55.2.13 PWRSts

```
unsigned int PWRSts
```

Флаги состояния питания шагового мотора.

5.55.2.14 Upwr

```
int Upwr
```

Напряжение на силовой части, десятки мВ.

5.55.2.15 Uusb

```
int Uusb
```

Напряжение на USB, десятки мВ.

5.55.2.16 WindSts

```
unsigned int WindSts
```

Состояние обмоток.

5.56 Структура status_t

Состояние устройства.

Поля данных

- unsigned int MoveSts

  Флаги состояния движения.
- unsigned int MvCmdSts

  Состояние команды движения.
- unsigned int PWRSts

  Флаги состояния питания шагового мотора.
- unsigned int EncSts

  Состояние энкодера.
- unsigned int WindSts

  Состояние обмоток.
- int CurPosition

  Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.
- int uCurPosition

  Дробная часть текущей позиции в микрошагах.
- long_t EncPosition

  Текущая позиция по данным с энкодера в импульсах энкодера используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзования.
- int CurSpeed

  Текущая скорость.
- int uCurSpeed

  Дробная часть текущей скорости в микрошагах.
- int Ipwr

  Ток потребления силовой части, мА.
- int Upwr

  Напряжение на силовой части, десятки мВ.
- int Iusb

  Ток потребления по USB, мА.
- int Uusb

  Напряжение на USB, десятки мВ.
- int CurT

  Температура процессора в десятых долях градусов Цельсия.
- unsigned int Flags

  Флаги состояния.
- unsigned int GPIOFlags

  Флаги состояния GPIO входов.
- unsigned int CmdBufFreeSpace

  Данное поле служебное.

5.56.1 Подробное описание

Состояние устройства.

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

См. также

get_status_impl

5.56.2 Поля

5.56.2.1 CmdBufFreeSpace

```
unsigned int CmdBufFreeSpace
```

Данное поле служебное.

Оно показывает количество свободных ячеек буфера цепочки синхронизации.

5.56.2.2 CurPosition

```
int CurPosition
```

Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.

В случае работы с DC-мотором в этом поле находится текущая позиция по данным с энкодера, в случае работы с ШД-мотором в режиме, когда первичными являются импульсы, подаваемые на мотор, в этом поле содержится целое значение шагов текущей позиции.

5.56.2.3 CurSpeed

```
int CurSpeed
```

Текущая скорость.

5.56.2.4 CurT

```
int CurT
```

Температура процессора в десятых долях градусов Цельсия.

5.56.2.5 EncPosition

```
long_t EncPosition
```

Текущая позиция по данным с энкодера в импульсах энкодера используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзования.

5.56.2.6 EncSts

```
unsigned int EncSts
```

Состояние энкодера.

5.56.2.7 Flags

```
unsigned int Flags
```

Флаги состояния.

5.56.2.8 GPIOFlags

```
unsigned int GPIOFlags
```

Флаги состояния GPIO входов.

5.56.2.9 Ipwr

```
int Ipwr
```

Ток потребления силовой части, мА.

5.56.2.10 Iusb

```
int Iusb
```

Ток потребления по USB, мА.

5.56.2.11 MoveSts

```
unsigned int MoveSts
```

Флаги состояния движения.

5.56.2.12 MvCmdSts

```
unsigned int MvCmdSts
```

Состояние команды движения.

5.56.2.13 PWRSts

```
unsigned int PWRSts
```

Флаги состояния питания шагового мотора.

5.56.2.14 uCurPosition

```
int uCurPosition
```

Дробная часть текущей позиции в микрошагах.

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым двигателем.

5.56.2.15 uCurSpeed

```
int uCurSpeed
```

Дробная часть текущей скорости в микрошагах.

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым двигателем.

5.56.2.16 Upwr

```
int Upwr
```

Напряжение на силовой части, десятки мВ.

5.56.2.17 Uusb

```
int Uusb
```

Напряжение на USB, десятки мВ.

5.56.2.18 WindSts

```
unsigned int WindSts
```

Состояние обмоток.

5.57 Структура sync_in_settings_calb_t

Настройки входной синхронизации с использованием пользовательских единиц.

Поля данных

- unsigned int SyncInFlags
  Флаги настроек синхронизации входа.
- unsigned int ClutterTime
  Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).
- float Position
  Желаемая позиция или смещение.
- float Speed
  Заданная скорость.
- unsigned int **reserved0**

5.57.1 Подробное описание

Настройки входной синхронизации с использованием пользовательских единиц.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

get_sync_in_settings_calb
set_sync_in_settings_calb
get_sync_in_settings, set_sync_in_settings

5.57.2 Поля

5.57.2.1 ClutterTime

```
unsigned int ClutterTime
```

Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).

5.57.2.2 Position

```
float Position
```

Желаемая позиция или смещение.

5.57.2.3 Speed

```
float Speed
```

Заданная скорость.

5.57.2.4 SyncInFlags

```
unsigned int SyncInFlags
```

Флаги настроек синхронизации входа.

5.58 Структура sync_in_settings_t

Настройки входной синхронизации.

Поля данных

- unsigned int SyncInFlags
  Флаги настроек синхронизации входа.
- unsigned int ClutterTime
  Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).
- int Position
  Желаемая позиция или смещение (в полных шагах)
- int uPosition
  Дробная часть позиции или смещения в микрошагах.
- unsigned int Speed
  Заданная скорость (для ШД: шагов/с, для DC: rpm).
- unsigned int uSpeed
  Заданная скорость в микрошагах в секунду.
- unsigned int **reserved0**

5.58.1 Подробное описание

Настройки входной синхронизации.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

get_sync_in_settings
set_sync_in_settings
get_sync_in_settings, set_sync_in_settings

5.58.2 Поля

5.58.2.1 ClutterTime

```
unsigned int ClutterTime
```

Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).

5.58.2.2 Speed

```
unsigned int Speed
```

Заданная скорость (для ШД: шагов/с, для DC: rpm).

Диапазон: 0..100000.

5.58.2.3 SyncInFlags

```
unsigned int SyncInFlags
```

Флаги настроек синхронизации входа.

5.58.2.4 uPosition

```
int uPosition
```

Дробная часть позиции или смещения в микрошагах.

Используется только с шаговым двигателем. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_↩ settings).

5.58.2.5 uSpeed

```
unsigned int uSpeed
```

Заданная скорость в микрошагах в секунду.

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings). Используется только с шаговым мотором.

5.59 Структура sync_out_settings_calb_t

Настройки выходной синхронизации с использованием пользовательских единиц.

Поля данных

- unsigned int SyncOutFlags

  Флаги настроек синхронизации выхода.
- unsigned int SyncOutPulseSteps

  Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг SYNCOUT_IN_STEPS или в микросекундах, если флаг сброшен.
- unsigned int SyncOutPeriod

  Период генерации импульсов (в шагах/отсчетах энкодера) используется при установленном флаге SYNCOUT_ONPERIOD.
- float Accuracy

  Это окрестность вокруг целевой координаты (в шагах/отсчетах энкодера), попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.

5.59.1 Подробное описание

Настройки выходной синхронизации с использованием пользовательских единиц.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

get_sync_out_settings_calb
set_sync_out_settings_calb
get_sync_out_settings, set_sync_out_settings

5.59.2 Поля

5.59.2.1 Accuracy

```
float Accuracy
```

Это окрестность вокруг целевой координаты (в шагах/отсчетах энкодера), попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.

5.59.2.2 SyncOutFlags

```
unsigned int SyncOutFlags
```

Флаги настроек синхронизации выхода.

5.59.2.3 SyncOutPeriod

```
unsigned int SyncOutPeriod
```

Период генерации импульсов (в шагах/отсчетах энкодера) используется при установленном флаге SYNCOUT_ONPERIOD.

5.59.2.4 SyncOutPulseSteps

```
unsigned int SyncOutPulseSteps
```

Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг SYNCOUT_IN_STEPS или в микросекундах, если флаг сброшен.

5.60 Структура sync_out_settings_t

Настройки выходной синхронизации.

Поля данных

- unsigned int SyncOutFlags

  Флаги настроек синхронизации выхода.
- unsigned int SyncOutPulseSteps

  Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг SYNCOUT_IN_STEPS или в микросекундах, если флаг сброшен.
- unsigned int SyncOutPeriod

  Период генерации импульсов (в шагах/отсчетах энкодера) используется при установленном флаге SYNCOUT_ONPERIOD.
- unsigned int Accuracy

  Это окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.
- unsigned int uAccuracy

  Это окрестность вокруг целевой координаты в микрошагах (используется только с шаговым двигателем).

5.60.1 Подробное описание

Настройки выходной синхронизации.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

get_sync_out_settings
set_sync_out_settings
get_sync_out_settings, set_sync_out_settings

5.60.2 Поля

5.60.2.1 Accuracy

```
unsigned int Accuracy
```

Это окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.

5.60.2.2 SyncOutFlags

```
unsigned int SyncOutFlags
```

Флаги настроек синхронизации выхода.

5.60.2.3 SyncOutPeriod

```
unsigned int SyncOutPeriod
```

Период генерации импульсов (в шагах/отсчетах энкодера) используется при установленном флаге SYNCOUT_ONPERIOD.

5.60.2.4 SyncOutPulseSteps

```
unsigned int SyncOutPulseSteps
```

Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг SYNCOUT_IN_STEPS или в микросекундах, если флаг сброшен.

5.60.2.5 uAccuracy

```
unsigned int uAccuracy
```

Это окрестность вокруг целевой координаты в микрошагах (используется только с шаговым двигателем).

Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

5.61 Структура uart_settings_t

Настройки UART.

Поля данных

- unsigned int Speed
  Скорость UART (в бодах)
- unsigned int UARTSetupFlags
  Флаги настроек четности команды UART.

5.61.1 Подробное описание

Настройки UART.

Эта структура содержит настройки UART.

См. также

get_uart_settings
set_uart_settings
get_uart_settings, set_uart_settings

5.61.2 Поля

5.61.2.1 UARTSetupFlags

```
unsigned int UARTSetupFlags
```

Флаги настроек четности команды UART.

Глава 6

Файлы

6.1 Файл ximc.h

Заголовочный файл для библиотеки libximc.

Структуры данных

- struct calibration_t

  Структура калибровок
- struct device_network_information_t

  Структура данных с информацией о сетевом устройстве.
- struct feedback_settings_t

  Настройки обратной связи.
- struct home_settings_t

  Настройки калибровки позиции.
- struct home_settings_calb_t

  Настройки калибровки позиции с использованием пользовательских единиц.
- struct move_settings_t

  Настройки движения.
- struct move_settings_calb_t

  Настройки движения с использованием пользовательских единиц.
- struct engine_settings_t

  Ограничения и настройки движения, связанные с двигателем.
- struct engine_settings_calb_t

  Ограничения и настройки движения, связанные с двигателем, с использованием пользовательских единиц.
- struct entype_settings_t

  Настройки типа мотора и типа силового драйвера.
- struct power_settings_t

  Настройки питания шагового мотора.
- struct secure_settings_t

  Структура содержит параметры защиты: критические значения электрических характеристик, флаги управления алгоритмами защиты.
- struct edges_settings_t

Настройки границ.

- struct edges_settings_calb_t

  Настройки границ с использованием пользовательских единиц.
- struct pid_settings_t

  Настройки ПИД.
- struct sync_in_settings_t

  Настройки входной синхронизации.
- struct sync_in_settings_calb_t

  Настройки входной синхронизации с использованием пользовательских единиц.
- struct sync_out_settings_t

  Настройки выходной синхронизации.
- struct sync_out_settings_calb_t

  Настройки выходной синхронизации с использованием пользовательских единиц.
- struct extio_settings_t

  Настройки EXTIO.
- struct brake_settings_t

  Настройки тормоза.
- struct control_settings_t

  Настройки управления.
- struct control_settings_calb_t

  Настройки управления с использованием пользовательских единиц.
- struct joystick_settings_t

  Настройки джойстика.
- struct ctp_settings_t

  Настройки контроля позиции(для шагового двигателя).
- struct uart_settings_t

  Настройки UART.
- struct network_settings_t

  Настройки сети.
- struct password_settings_t

  Пароль.
- struct calibration_settings_t

  Калибровочные коэффициенты.
- struct controller_name_t

  Пользовательское имя контроллера и флаги настройки.
- struct nonvolatile_memory_t

  Пользовательские данные для сохранения во FRAM.
- struct emf_settings_t

  Настройки EMF.
- struct engine_advanced_setup_t

  Настройки EAS.
- struct extended_settings_t

  Настройки EST.
- struct get_position_t

  Данные о позиции.
- struct get_position_calb_t

  Данные о позиции.
- struct set_position_t

  Данные о позиции.

- struct set_position_calb_t

  Данные о позиции с использованием пользовательских единиц.
- struct status_t

  Состояние устройства.
- struct status_calb_t

  Состояние устройства с использованием пользовательских единиц.
- struct measurements_t

  Структура содержит последовательность измеренных параметров движения оси – скоростей и ошибок по позиции.
- struct chart_data_t

  Дополнительное состояние устройства.
- struct device_information_t

  Информации о контроллере.
- struct serial_number_t

  Структура с серийным номером и версией железа.
- struct analog_data_t

  Аналоговые данные.
- struct debug_read_t

  Отладочные данные.
- struct debug_write_t

  Отладочные данные.
- struct stage_name_t

  Пользовательское имя подвижки.
- struct stage_information_t

  Устарело.
- struct stage_settings_t

  Устарело.
- struct motor_information_t

  Устарело.
- struct motor_settings_t

  Устарело.
- struct encoder_information_t

  Устарело.
- struct encoder_settings_t

  Устарело.
- struct hallsensor_information_t

  Устарело.
- struct hallsensor_settings_t

  Устарело.
- struct gear_information_t

  Устарело.
- struct gear_settings_t

  Устарело.
- struct accessories_settings_t

  Устарело.
- struct init_random_t

  Случайный ключ.
- struct globally_unique_identifier_t

  Глобальный уникальный идентификатор.

Макросы

- #define XIMC_API
  Макрос импорта библиотеки.
- #define XIMC_CALLCONV
  Библиотека вызывающая условные макросы.
- #define XIMC_RETTYPE void*
  Возвращаеемый тип потока.
- #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_value_error -3
  ошибка записи значения
- #define result_nodevice -4
  устройство не подключено

Уровень логирования

- #define LOGLEVEL_ERROR 0x01
  Уровень логирования - ошибка
- #define LOGLEVEL_WARNING 0x02
  Уровень логирования - предупреждение
- #define LOGLEVEL_INFO 0x03
  Уровень логирования - информация
- #define LOGLEVEL_DEBUG 0x04
  Уровень логирования - отладка

Флаги поиска устройств

Это битовая маска для побитовых операций.

- #define ENUMERATE_PROBE 0x01
  Проверять, является ли устройство XIMC-совместимым.
- #define ENUMERATE_ALL_COM 0x02
  Проверять все COM-устройства
- #define ENUMERATE_NETWORK 0x04
  Проверять сетевые устройства

Флаги состояния движения

Это битовая маска для побитовых операций. Возвращаются командой get_status.

См. также

get_status
status_t::MoveSts, *get_status_impl*

- #define MOVE_STATE_MOVING 0x01
  Если флаг установлен, то контроллер пытается вращать двигателем.
- #define MOVE_STATE_TARGET_SPEED 0x02
  Флаг устанавливается при достижении заданной скорости.
- #define MOVE_STATE_ANTIPLAY 0x04
  Выполняется компенсация люфта, если флаг установлен.

Флаги настроек контроллера

Это битовая маска для побитовых операций.

См. также

set_controller_name
get_controller_name
controller_name_t::CtrlFlags, *get_controller_name*, *set_controller_name*

- #define EEPROM_PRECEDENCE 0x01
  Если флаг установлен, то настройки в EEPROM подвижки имеют приоритет над текущими настройками и заменяют их при обнаружении EEPROM.

Флаги состояния питания шагового мотора

Это битовая маска для побитовых операций. Возвращаются командой get_status.

См. также

get_status
status_t::PWRSts, *get_status_impl*

- #define PWR_STATE_UNKNOWN 0x00
  Неизвестное состояние, которое не должно никогда реализовываться.
- #define PWR_STATE_OFF 0x01
  Обмотки мотора разомкнуты и не управляются драйвером.
- #define PWR_STATE_NORM 0x03
  Обмотки запитаны номинальным током.
- #define PWR_STATE_REDUCT 0x04
  Обмотки намеренно запитаны уменьшенным током от рабочего для снижения потребляемой мощности.
- #define PWR_STATE_MAX 0x05
  Обмотки двигателя питаются от максимального тока, который драйвер может обеспечить при этом напряжении.

Флаги состояния

Это битовая маска для побитовых операций. Содержат бинарные значения состояния контроллера. Могут быть объединены с помощью логического ИЛИ.

См. также

get_status
status_t::Flags, *get_status_impl*

- #define STATE_CONTR 0x000003F
  Флаги состояния контроллера.
- #define STATE_ERRC 0x0000001
  Недопустимая команда.
- #define STATE_ERRD 0x0000002
  Обнаружена ошибка целостности данных.
- #define STATE_ERRV 0x0000004
  Недопустимое значение данных.
- #define STATE_EEPROM_CONNECTED 0x0000010
  Подключена память EEPROM с настройками.
- #define STATE_IS_HOMED 0x0000020
  Калибровка выполнена.
- #define STATE_SECUR 0x1B3FFC0
  Флаги безопасности.
- #define STATE_ALARM 0x0000040
  Контроллер находится в состоянии ALARM, показывая, что случилась какая-то опасная ситуация.
- #define STATE_CTP_ERROR 0x0000080
  Контроль позиции нарушен (используется только с шаговым двигателем).
- #define STATE_POWER_OVERHEAT 0x0000100
  Перегрев силового драйвера.
- #define STATE_CONTROLLER_OVERHEAT 0x0000200
  Перегрелась микросхема контроллера.
- #define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400
  Превышено напряжение на силовой части.
- #define STATE_OVERLOAD_POWER_CURRENT 0x0000800
  Превышен максимальный ток потребления силовой части.
- #define STATE_OVERLOAD_USB_VOLTAGE 0x0001000
  Устарело.
- #define STATE_LOW_USB_VOLTAGE 0x0002000
  Устарело.
- #define STATE_OVERLOAD_USB_CURRENT 0x0004000
  Устарело.
- #define STATE_BORDERS_SWAP_MISSET 0x0008000
  Достижение неверной границы.
- #define STATE_LOW_POWER_VOLTAGE 0x0010000
  Напряжение на силовой части ниже, чем напряжение Low Voltage Protection.
- #define STATE_H_BRIDGE_FAULT 0x0020000
  Получен сигнал от драйвера о неисправности
- #define STATE_WINDING_RES_MISMATCH 0x0100000
  Сопротивления обмоток слишком сильно отличаются друг от друга.
- #define STATE_ENCODER_FAULT 0x0200000
  Получен сигнал от энкодера о неисправности
- #define STATE_ENGINE_RESPONSE_ERROR 0x0800000
  Ошибка реакции двигателя на управляющее воздействие.
- #define STATE_EXTIO_ALARM 0x1000000
  Ошибка вызвана внешним входным сигналом EXTIO.

Флаги состояния GPIO входов

Это битовая маска для побитовых операций. Содержат бинарные значения состояния контроллера. Могут быть объединены с помощью логического ИЛИ.

См. также

get_status
status_t::GPIOFlags, *get_status_impl*

- #define STATE_DIG_SIGNAL 0xFFFF
  Флаги цифровых сигналов.
- #define STATE_RIGHT_EDGE 0x0001
  Достижение правой границы.
- #define STATE_LEFT_EDGE 0x0002
  Достижение левой границы.
- #define STATE_BUTTON_RIGHT 0x0004
  Состояние кнопки "вправо" (1, если нажата).
- #define STATE_BUTTON_LEFT 0x0008
  Состояние кнопки "влево" (1, если нажата).
- #define STATE_GPIO_PINOUT 0x0010
  Если флаг установлен, ввод/вывод общего назначения работает как выход; если флаг сброшен, ввод/вывод работает как вход.
- #define STATE_GPIO_LEVEL 0x0020
  Состояние ввода/вывода общего назначения.
- #define STATE_BRAKE 0x0200
  Состояние вывода управления тормозом.
- #define STATE_REV_SENSOR 0x0400
  Состояние вывода датчика оборотов (флаг "1", если датчик активен).
- #define STATE_SYNC_INPUT 0x0800
  Состояние входа синхронизации (1, если вход синхронизации активен).
- #define STATE_SYNC_OUTPUT 0x1000
  Состояние выхода синхронизации (1, если выход синхронизации активен).
- #define STATE_ENC_A 0x2000
  Состояние ножки A энкодера (флаг "1", если энкодер активен).
- #define STATE_ENC_B 0x4000
  Состояние ножки B энкодера (флаг "1", если энкодер активен).

Состояние энкодера

Это битовая маска для побитовых операций. Состояние энкодера, подключенного к контроллеру.

Заметки

Флаг ENCD работает только в режимах None и EMF. В других режимах, таких как Encoder и Encoder Mediated, он не используется, потому что эти режимы невозможно использовать без энкодера. Сразу после включения контроллера флаг будет в состоянии unknown — чтобы определить положение, нужно совершить движение. По мере движения значение флага может меняться.

См. также

get_status
status_t::EncSts, *get_status_impl*

- #define ENC_STATE_ABSENT 0x00
  Энкодер не подключен.
- #define ENC_STATE_UNKNOWN 0x01
  Состояние энкодера неизвестно.
- #define ENC_STATE_MALFUNC 0x02
  Энкодер подключен и неисправен.
- #define ENC_STATE_REVERS 0x03
  Энкодер подключен и исправен, но считает в другую сторону.
- #define ENC_STATE_OK 0x04
  Энкодер подключен и работает должным образом.

Состояние обмоток

Это битовая маска для побитовых операций. Состояние обмоток двигателя, подключенного к контроллеру.

См. также

get_status
status_t::WindSts, *get_status_impl*

- #define WIND_A_STATE_ABSENT 0x00
  Обмотка A не подключена.
- #define WIND_A_STATE_UNKNOWN 0x01
  Состояние обмотки A неизвестно.
- #define WIND_A_STATE_MALFUNC 0x02
  Короткое замыкание на обмотке A.
- #define WIND_A_STATE_OK 0x03
  Обмотка A работает адекватно.
- #define WIND_B_STATE_ABSENT 0x00
  Обмотка B не подключена.
- #define WIND_B_STATE_UNKNOWN 0x10
  Состояние обмотки B неизвестно.
- #define WIND_B_STATE_MALFUNC 0x20
  Короткое замыкание на обмотке B.
- #define WIND_B_STATE_OK 0x30
  Обмотка B работает адекватно.

Состояние команды движения

Это битовая маска для побитовых операций. Состояние команды движения (касается command_move, command_movr, command_left, command_right, command_stop, command_home, command_loft, command_sstp) и статуса её выполнения (выполняется, завершено, ошибка)

См. также

get_status
status_t::MvCmdSts, *get_status_impl*

- #define MVCMD_NAME_BITS 0x3F
  Битовая маска активной команды.
- #define MVCMD_UKNWN 0x00
  Неизвестная команда.
- #define MVCMD_MOVE 0x01
  Команда move.
- #define MVCMD_MOVR 0x02
  Команда movr.
- #define MVCMD_LEFT 0x03
  Команда left.
- #define MVCMD_RIGHT 0x04
  Команда rigt.
- #define MVCMD_STOP 0x05
  Команда stop.
- #define MVCMD_HOME 0x06
  Команда home.
- #define MVCMD_LOFT 0x07
  Команда loft.
- #define MVCMD_SSTP 0x08
  Команда плавной остановки(SSTP).
- #define MVCMD_ERROR 0x40

Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно).

- #define MVCMD_RUNNING 0x80

  Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).

Флаги параметров движения

Это битовая маска для побитовых операций. Определяют настройки параметров движения. Возвращаются командой get_move_settings.

См. также

set_move_settings
get_move_settings
move_settings_t::MoveFlags, *get_move_settings*, *set_move_settings*

- #define RPM_DIV_1000 0x01

  Флаг указывает на то что рабочая скорость указанная в команде задана в милли rpm.

Флаги параметров мотора

Это битовая маска для побитовых операций. Определяют настройки движения и работу ограничителей. Возвращаются командой get_engine_settings. Могут быть объединены с помощью логического ИЛИ.

См. также

set_engine_settings
get_engine_settings
engine_settings_t::EngineFlags, *get_engine_settings*, *set_engine_settings*

- #define ENGINE_REVERSE 0x001

  Флаг реверса.
- #define ENGINE_CURRENT_AS_RMS 0x002

  Флаг интерпретации значения тока.
- #define ENGINE_MAX_SPEED 0x004

  Флаг максимальной скорости.
- #define ENGINE_ANTIPLAY 0x008

  Компенсация люфта.
- #define ENGINE_ACCEL_ON 0x010

  Ускорение.
- #define ENGINE_LIMIT_VOLT 0x020

  Номинальное напряжение мотора.
- #define ENGINE_LIMIT_CURR 0x040

  Номинальный ток мотора.
- #define ENGINE_LIMIT_RPM 0x080

  Номинальная частота вращения мотора.
- #define ENGINE_USE_PEAK_CURRENT 0x100

  Разрешить использование пикового тока.

Флаги параметров микрошагового режима

Это битовая маска для побитовых операций. Определяют деление шага в микрошаговом режиме. Используются с шаговыми моторами. Возвращаются командой get_engine_settings. Могут быть объединены с помощью логического ИЛИ.

См. также

engine_settings_t::flags
set_engine_settings
get_engine_settings
engine_settings_t::MicrostepMode, *get_engine_settings*, *set_engine_settings*

- #define MICROSTEP_MODE_FULL 0x01
  Полношаговый режим.
- #define MICROSTEP_MODE_FRAC_2 0x02
  Деление шага 1/2.
- #define MICROSTEP_MODE_FRAC_4 0x03
  Деление шага 1/4.
- #define MICROSTEP_MODE_FRAC_8 0x04
  Деление шага 1/8.
- #define MICROSTEP_MODE_FRAC_16 0x05
  Деление шага 1/16.
- #define MICROSTEP_MODE_FRAC_32 0x06
  Деление шага 1/32.
- #define MICROSTEP_MODE_FRAC_64 0x07
  Деление шага 1/64.
- #define MICROSTEP_MODE_FRAC_128 0x08
  Деление шага 1/128.
- #define MICROSTEP_MODE_FRAC_256 0x09
  Деление шага 1/256.

Флаги, определяющие тип мотора

Это битовая маска для побитовых операций. Определяют тип мотора. Возвращаются командой get_entype_settings.

См. также

engine_settings_t::flags
set_entype_settings
get_entype_settings
entype_settings_t::EngineType, *get_entype_settings*, *set_entype_settings*

- #define ENGINE_TYPE_NONE 0x00
  Это значение не нужно использовать.
- #define ENGINE_TYPE_DC 0x01
  Мотор постоянного тока.
- #define ENGINE_TYPE_2DC 0x02
  Два мотора постоянного тока, что приводит к эмуляции двух контроллеров.
- #define ENGINE_TYPE_STEP 0x03
  Шаговый мотор.
- #define ENGINE_TYPE_TEST 0x04
  Продолжительность включения фиксирована.
- #define ENGINE_TYPE_BRUSHLESS 0x05
  Бесщеточный мотор.

Флаги, определяющие тип силового драйвера

Это битовая маска для побитовых операций. Определяют тип силового драйвера. Возвращаются командой get_entype_settings.

См. также

engine_settings_t::flags
set_entype_settings
get_entype_settings
entype_settings_t::DriverType, *get_entype_settings*, *set_entype_settings*

- #define DRIVER_TYPE_INTEGRATE 0x02
  Силовой драйвер с использованием ключей, интегрированных в микросхему.
- #define DRIVER_TYPE_EXTERNAL 0x03
  Внешний силовой драйвер.

Флаги параметров питания шагового мотора

Это битовая маска для побитовых операций. Возвращаются командой get_power_settings.

См. также

get_power_settings
set_power_settings
power_settings_t::PowerFlags, *get_power_settings*, *set_power_settings*

- #define POWER_REDUCT_ENABLED 0x01
  Если флаг установлен, уменьшить ток по прошествии CurrReductDelay.
- #define POWER_OFF_ENABLED 0x02
  Если флаг установлен, снять напряжение с обмоток по прошествии PowerOffDelay.
- #define POWER_SMOOTH_CURRENT 0x04
  Если установлен, то запитывание обмоток, снятие питания или снижение/повышение тока происходят плавно со скоростью CurrentSetTime, а только потом выполняется та задача, которая вызвала это плавное изменение.

Флаги критических параметров.

Это битовая маска для побитовых операций. Возвращаются командой get_secure_settings.

См. также

get_secure_settings
set_secure_settings
secure_settings_t::Flags, *get_secure_settings*, *set_secure_settings*

- #define ALARM_ON_DRIVER_OVERHEATING 0x01
  Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера.
- #define LOW_UPWR_PROTECTION 0x02
  Если флаг установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.
- #define H_BRIDGE_ALERT 0x04
  Если флаг установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.
- #define ALARM_ON_BORDERS_SWAP_MISSET 0x08
  Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевого выключателя.
- #define ALARM_FLAGS_STICKING 0x10
  Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.
- #define BRAKING_OVERVOLTAGE_PROTECTION 0x20
  Если флаг установлен, то микропрограмма контроллера будет замыкать нижние ключи H-моста, отсоединяя мотор от цепи питания, при перенапряжении.
- #define ALARM_WINDING_MISMATCH 0x40
  Если флаг установлен, то войти в состояние Alarm при получении сигнала рассогласования обмоток

- #define ALARM_ENGINE_RESPONSE 0x80
  Если флаг установлен, то войти в состояние Alarm при получении сигнала ошибки реакции двигателя на управляющее воздействие

Флаги установки положения

Это битовая маска для побитовых операций. Возвращаются командой get_position.

См. также

get_position
set_position
set_position_t::PosFlags, *set_position*

- #define SETPOS_IGNORE_POSITION 0x01
  Если установлен, то позиция в шагах и микрошагах не обновляется.
- #define SETPOS_IGNORE_ENCODER 0x02
  Если установлен, то счётчик энкодера не обновляется.

Тип обратной связи.

Это битовая маска для побитовых операций.

См. также

set_feedback_settings
get_feedback_settings
feedback_settings_t::FeedbackType, *get_feedback_settings*, *set_feedback_settings*

- #define FEEDBACK_ENCODER 0x01
  Обратная связь с помощью энкодера.
- #define FEEDBACK_EMF 0x04
  Обратная связь по ЭДС.
- #define FEEDBACK_NONE 0x05
  Обратная связь отсутствует.
- #define FEEDBACK_ENCODER_MEDIATED 0x06
  Обратная связь по энкодеру, опосредованному относительно двигателя механической передачей (например, винтовой передачей).

Флаги обратной связи.

Это битовая маска для побитовых операций.

См. также

set_feedback_settings
get_feedback_settings
feedback_settings_t::FeedbackFlags, *get_feedback_settings*, *set_feedback_settings*

- #define FEEDBACK_ENC_REVERSE 0x01
  Обратный счет у энкодера.
- #define FEEDBACK_ENC_ADAPTIVE_HOLDING 0x02
  Включает алгоритм адаптивного удержания.
- #define FEEDBACK_ENC_FILTER_NONE 0x00
  Выключает внутренний фильтр сигнала энкодера.
- #define FEEDBACK_ENC_FILTER_WEAK 0x10
  Слабая фильтрация шумов: максимальная частота сигнала энкодера 3 МГц.
- #define FEEDBACK_ENC_FILTER_MEDIUM 0x20
  Средняя фильтрация шумов: максимальная частота сигнала энкодера 1 МГц.
- #define FEEDBACK_ENC_FILTER_STRONG 0x30

Сильная фильтрация шумов: максимальная частота сигнала энкодера 300 кГц.

- #define FEEDBACK_ENC_FILTER_BITS 0x30
  Биты, отвечающие за настройку внутреннего фильтра энкодерного сигнала.
- #define FEEDBACK_ENC_TYPE_AUTO 0x00
  Определяет тип энкодера автоматически.
- #define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40
  Недифференциальный энкодер.
- #define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80
  Дифференциальный энкодер.
- #define FEEDBACK_ENC_TYPE_BITS 0xC0
  Биты, отвечающие за тип энкодера.

Флаги настроек синхронизации входа

Это битовая маска для побитовых операций.

См. также

sync_in_settings_t::SyncInFlags, *get_sync_in_settings*, *set_sync_in_settings*

- #define SYNCIN_ENABLED 0x01
  Включение необходимости импульса синхронизации для начала движения.
- #define SYNCIN_INVERT 0x02
  Если установлен - срабатывает на спадающем фронте из 1 в 0.
- #define SYNCIN_GOTOPOSITION 0x04
  Если флаг установлен, то двигатель смещается к позиции, установленной в Position и uPosition, иначе двигатель смещается на Position и uPosition.

Флаги настроек синхронизации выхода

Это битовая маска для побитовых операций.

См. также

sync_out_settings_t::SyncOutFlags, *get_sync_out_settings*, *set_sync_out_settings*

- #define SYNCOUT_ENABLED 0x01
  Синхронизация выхода работает согласно настройкам, если флаг установлен.
- #define SYNCOUT_STATE 0x02
  Когда значение выхода управляется напрямую (см.
- #define SYNCOUT_INVERT 0x04
  Нулевой логический уровень является активным, если флаг установлен, а единичный - если флаг сброшен.
- #define SYNCOUT_IN_STEPS 0x08
  Если флаг установлен использовать шаги/импульсы энкодера для выходных импульсов синхронизации вместо миллисекунд.
- #define SYNCOUT_ONSTART 0x10
  Генерация синхронизирующего импульса при начале движения.
- #define SYNCOUT_ONSTOP 0x20
  Генерация синхронизирующего импульса при остановке.
- #define SYNCOUT_ONPERIOD 0x40
  Выдает импульс синхронизации после прохождения SyncOutPeriod отсчётов.

Флаги настройки работы внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

get_extio_settings
set_extio_settings
extio_settings_t::EXTIOSetupFlags, *get_extio_settings*, *set_extio_settings*

- #define EXTIO_SETUP_OUTPUT 0x01
  Если флаг установлен, то ножка в состоянии вывода, иначе - ввода.
- #define EXTIO_SETUP_INVERT 0x02
  Если флаг установлен, то нули считаются активным состоянием выхода, а спадающие фронты - как момент подачи входного сигнала.

Флаги настройки режимов внешнего ввода/вывода

Это битовая маска для побитовых операций.

См. также

extio_settings_t::extio_mode_flags
get_extio_settings
set_extio_settings
extio_settings_t::EXTIOModeFlags, *get_extio_settings*, *set_extio_settings*

- #define EXTIO_SETUP_MODE_IN_BITS 0x0F
  Биты, отвечающие за поведение при переходе сигнала в активное состояние.
- #define EXTIO_SETUP_MODE_IN_NOP 0x00
  Ничего не делать.
- #define EXTIO_SETUP_MODE_IN_STOP 0x01
  По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).
- #define EXTIO_SETUP_MODE_IN_PWOF 0x02
  Выполняет команду PWOF, обесточивая обмотки двигателя.
- #define EXTIO_SETUP_MODE_IN_MOVR 0x03
  Выполняется команда MOVR с последними настройками.
- #define EXTIO_SETUP_MODE_IN_HOME 0x04
  Выполняется команда HOME.
- #define EXTIO_SETUP_MODE_IN_ALARM 0x05
  Войти в состояние ALARM при переходе сигнала в активное состояние.
- #define EXTIO_SETUP_MODE_OUT_BITS 0xF0
  Биты выбора поведения на выходе.
- #define EXTIO_SETUP_MODE_OUT_OFF 0x00
  Ножка всегда в неактивном состоянии.
- #define EXTIO_SETUP_MODE_OUT_ON 0x10
  Ножка всегда в активном состоянии.
- #define EXTIO_SETUP_MODE_OUT_MOVING 0x20
  Ножка находится в активном состоянии при движении.
- #define EXTIO_SETUP_MODE_OUT_ALARM 0x30
  Ножка находится в активном состоянии при нахождении в состоянии ALARM.
- #define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40
  Ножка находится в активном состоянии при подаче питания на обмотки.

Флаги границ

Это битовая маска для побитовых операций. Типы границ и поведение позиционера на границах. Могут быть объединены с помощью побитового ИЛИ.

См. также

get_edges_settings
set_edges_settings
edges_settings_t::BorderFlags, *get_edges_settings*, *set_edges_settings*

- #define BORDER_IS_ENCODER 0x01
  Если флаг установлен, границы определяются предустановленными точками на шкале позиции.
- #define BORDER_STOP_LEFT 0x02
  Если флаг установлен, мотор останавливается при достижении левой границы.
- #define BORDER_STOP_RIGHT 0x04
  Если флаг установлен, мотор останавливается при достижении правой границы.
- #define BORDERS_SWAP_MISSET_DETECTION 0x08
  Если флаг установлен, мотор останавливается по достижении любой из границ.

Флаги концевых выключателей

Это битовая маска для побитовых операций. Определяют направление и состояние границ. Могут быть объединены с помощью побитового ИЛИ.

См. также

get_edges_settings
set_edges_settings
edges_settings_t::EnderFlags, *get_edges_settings*, *set_edges_settings*

- #define ENDER_SWAP 0x01
  Если флаг установлен, первый концевой выключатель находится справа; иначе - слева.
- #define ENDER_SW1_ACTIVE_LOW 0x02
  1 - Концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте.
- #define ENDER_SW2_ACTIVE_LOW 0x04
  1 - Концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте.

Флаги настроек тормоза

Это битовая маска для побитовых операций. Определяют поведение тормоза. Могут быть объединены с помощью побитового ИЛИ.

См. также

get_brake_settings
set_brake_settings
brake_settings_t::BrakeFlags, *get_brake_settings*, *set_brake_settings*

- #define BRAKE_ENABLED 0x01
  Управление тормозом включено, если флаг установлен.
- #define BRAKE_ENG_PWROFF 0x02
  Тормоз отключает питание шагового мотора, если флаг установлен.

Флаги управления

Это битовая маска для побитовых операций. Определяют параметры управления мотором с помощью джойстика или кнопок. Могут быть объединены с помощью побитового ИЛИ.

См. также

get_control_settings
set_control_settings
control_settings_t::Flags, *get_control_settings*, *set_control_settings*

- #define CONTROL_MODE_BITS 0x03
  Биты управления мотором с помощью джойстика или кнопок влево/вправо.
- #define CONTROL_MODE_OFF 0x00
  Управление отключено.
- #define CONTROL_MODE_JOY 0x01
  Управление с помощью джойстика.
- #define CONTROL_MODE_LR 0x02
  Управление с помощью кнопок влево/вправо.
- #define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04
  Нажатая левая кнопка соответствует открытому контакту, если этот флаг установлен.
- #define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08
  Нажатая правая кнопка соответствует открытому контакту, если этот флаг установлен.

Флаги джойстика

Это битовая маска для побитовых операций. Управляют состояниями джойстика.

См. также

set_joystick_settings
get_joystick_settings
joystick_settings_t::JoyFlags, *get_joystick_settings*, *set_joystick_settings*

- #define JOY_REVERSE 0x01
  Реверс воздействия джойстика.

Флаги контроля позиции

Это битовая маска для побитовых операций. Определяют настройки контроля позиции. Могут быть объединены с помощью побитового ИЛИ.

См. также

get_ctp_settings
set_ctp_settings
ctp_settings_t::CTPFlags, *get_ctp_settings*, *set_ctp_settings*

- #define CTP_ENABLED 0x01
  Контроль позиции включен, если флаг установлен.
- #define CTP_BASE 0x02
  Управление положением основано на датчике вращения, если установлен этот флаг; в противном случае - на энкодере.
- #define CTP_ALARM_ON_ERROR 0x04
  Войти в состояние ALARM при расхождении позиции, если флаг установлен.
- #define REV_SENS_INV 0x08
  Сенсор считается активным, когда на нём 0, инвертирование делает активным уровень 1.
- #define CTP_ERROR_CORRECTION 0x10
  Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

Флаги настроек команды home

Это битовая маска для побитовых операций. Определяют поведение для команды home. Могут быть объединены с помощью побитового ИЛИ.

См. также

get_home_settings
set_home_settings
command_home
home_settings_t::HomeFlags, *get_home_settings*, *set_home_settings*

- #define HOME_DIR_FIRST 0x001
  Определяет направление первоначального движения мотора после поступления команды HOME.
- #define HOME_DIR_SECOND 0x002
  Определяет направление второго движения мотора.
- #define HOME_MV_SEC_EN 0x004
  Если флаг установлен, реализуется второй этап доводки в домашнюю позицию; иначе - этап пропускается.
- #define HOME_HALF_MV 0x008
  Если флаг установлен, сигналы остановки игнорируются в первой половине второго движения.
- #define HOME_STOP_FIRST_BITS 0x030
  Биты, отвечающие за выбор сигнала завершения первого движения.
- #define HOME_STOP_FIRST_REV 0x010
  Первое движение завершается по сигналу с Revolution sensor.
- #define HOME_STOP_FIRST_SYN 0x020
  Первое движение завершается по сигналу со входа синхронизации.
- #define HOME_STOP_FIRST_LIM 0x030
  Первое движение завершается по сигналу с концевого переключателя.
- #define HOME_STOP_SECOND_BITS 0x0C0
  Биты, отвечающие за выбор сигнала завершения второго движения.
- #define HOME_STOP_SECOND_REV 0x040
  Второе движение завершается по сигналу с Revolution sensor.
- #define HOME_STOP_SECOND_SYN 0x080
  Второе движение завершается по сигналу со входа синхронизации.
- #define HOME_STOP_SECOND_LIM 0x0C0
  Второе движение завершается по сигналу с концевого переключателя.
- #define HOME_USE_FAST 0x100
  Если флаг установлен, используется быстрый поиск домашней позиции; иначе - традиционный.

Флаги настроек четности команды UART

Это битовая маска для побитовых операций.

См. также

uart_settings_t::UARTSetupFlags, *get_uart_settings*, *set_uart_settings*

- #define UART_PARITY_BITS 0x03
  Биты, отвечающие за выбор четности.
- #define UART_PARITY_BIT_EVEN 0x00
  Бит 1, если четный
- #define UART_PARITY_BIT_ODD 0x01
  Бит 1, если нечетный
- #define UART_PARITY_BIT_SPACE 0x02
  Бит четности всегда 0.
- #define UART_PARITY_BIT_MARK 0x03
  Бит четности всегда 1.
- #define UART_PARITY_BIT_USE 0x04
  Бит чётности не используется, если "0"; бит четности используется, если "1".
- #define UART_STOP_BIT 0x08
  Если установлен, один стоповый бит; иначе - 2 стоповых бита

Флаги типа двигателя

Это битовая маска для побитовых операций.

См. также

motor_settings_t::MotorType, *get_motor_settings*, *set_motor_settings*

- #define MOTOR_TYPE_UNKNOWN 0x00
  Неизвестный двигатель
- #define MOTOR_TYPE_STEP 0x01
  Шаговый двигатель
- #define MOTOR_TYPE_DC 0x02
  DC-двигатель
- #define MOTOR_TYPE_BLDC 0x03
  BLDC-двигатель

Флаги настроек энкодера

Это битовая маска для побитовых операций.

См. также

accessories_settings_t::MBSettings, *get_accessories_settings*, *set_accessories_settings*

- #define ENCSET_DIFFERENTIAL_OUTPUT 0x001
  Если флаг установлен, то энкодер имеет дифференциальный выход, иначе - несимметричный выход
- #define ENCSET_PUSHPULL_OUTPUT 0x004
  Если флаг установлен, то энкодер имеет двухтактный выход, иначе - выход с открытым коллектором
- #define ENCSET_INDEXCHANNEL_PRESENT 0x010
  Если флаг установлен, то энкодер имеет дополнительный индексный канал, иначе - он отсутствует
- #define ENCSET_REVOLUTIONSENSOR_PRESENT 0x040
  Если флаг установлен, то энкодер имеет датчик оборотов, иначе - он отсутствует
- #define ENCSET_REVOLUTIONSENSOR_ACTIVE_HIGH 0x100
  Если флаг установлен, то активное состояние датчика оборотов соответствует логической 1, иначе - логическому 0.
- #define MB_AVAILABLE 0x01
  Если флаг установлен, то магнитный тормоз доступен
- #define MB_POWERED_HOLD 0x02
  Если флаг установлен, то магнитный тормоз находится в режиме удержания (активен) при подаче питания

Флаги настроек температурного датчика

Это битовая маска для побитовых операций.

См. также

accessories_settings_t::LimitSwitchesSettings, *get_accessories_settings*, *set_accessories_settings*

- #define TS_TYPE_BITS 0x07
  Биты, отвечающие за тип температурного датчика.
- #define TS_TYPE_UNKNOWN 0x00
  Неизвестный сенсор
- #define TS_TYPE_THERMOCOUPLE 0x01
  Термопара
- #define TS_TYPE_SEMICONDUCTOR 0x02
  Полупроводниковый температурный датчик
- #define TS_AVAILABLE 0x08
  Если флаг установлен, то датчик температуры доступен
- #define LS_ON_SW1_AVAILABLE 0x01

Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, доступен

- #define LS_ON_SW2_AVAILABLE 0x02
  Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, доступен
- #define LS_SW1_ACTIVE_LOW 0x04
  Если флаг установлен, то концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте
- #define LS_SW2_ACTIVE_LOW 0x08
  Если флаг установлен, то концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте
- #define LS_SHORTED 0x10
  Если флаг установлен, то концевые переключатели замкнуты.

Флаги автоопределения характеристик обмоток двигателя.

Это битовая маска для побитовых операций.

См. также

set_emf_settings
get_emf_settings
emf_settings_t::BackEMFFlags, get_emf_settings, set_emf_settings

- #define BACK_EMF_INDUCTANCE_AUTO 0x01
  Флаг автоопределения индуктивности обмоток двигателя.
- #define BACK_EMF_RESISTANCE_AUTO 0x02
  Флаг автоопределения сопротивления обмоток двигателя.
- #define BACK_EMF_KM_AUTO 0x04
  Флаг автоопределения электромеханического коэффициента двигателя.

Определения типов

- typedef unsigned long long **ulong_t**
- typedef long long **long_t**
- typedef int device_t
  Тип идентификатора устройства
- typedef int result_t
  Тип, определяющий результат выполнения команды.
- typedef uint32_t device_enumeration_t
  Тип, определяющий структуру данных о всех контроллерах, обнаруженных при опросе устройств.
- typedef struct calibration_t calibration_t
  Структура калибровок
- typedef struct device_network_information_t device_network_information_t
  Структура данных с информацией о сетевом устройстве.

Функции

Группа команд настройки контроллера

Функции для чтения/записи большинства настроек контроллера.

- result_t XIMC_API set_feedback_settings (device_t id, const feedback_settings_t *feedback_settings)
  Запись настроек обратной связи.
- result_t XIMC_API get_feedback_settings (device_t id, feedback_settings_t *feedback_settings)

Чтение настроек обратной связи

- result_t XIMC_API set_home_settings (device_t id, const home_settings_t *home_settings)
  Команда записи настроек для подхода в home position.
- result_t XIMC_API set_home_settings_calb (device_t id, const home_settings_calb_t *home_settings_calb, const calibration_t *calibration)
  Команда записи настроек для подхода в home position с использованием пользовательских единиц.
- result_t XIMC_API get_home_settings (device_t id, home_settings_t *home_settings)
  Команда чтения настроек для подхода в home position.
- result_t XIMC_API get_home_settings_calb (device_t id, home_settings_calb_t *home_settings_calb, const calibration_t *calibration)
  Команда чтения настроек для подхода в home position с использованием пользовательских единиц.
- result_t XIMC_API set_move_settings (device_t id, const move_settings_t *move_settings)
  Команда записи настроек перемещения (скорость, ускорение, порог и т.
- result_t XIMC_API set_move_settings_calb (device_t id, const move_settings_calb_t *move_settings_calb, const calibration_t *calibration)
  Команда записи настроек перемещения, с использованием пользовательских единиц (скорость, ускорение, порог и т.
- result_t XIMC_API get_move_settings (device_t id, move_settings_t *move_settings)
  Команда чтения настроек перемещения (скорость, ускорение, порог и т.
- result_t XIMC_API get_move_settings_calb (device_t id, move_settings_calb_t *move_settings_calb, const calibration_t *calibration)
  Команда чтения настроек перемещения с использованием пользовательских единиц (скорость, ускорение, порог и т.
- result_t XIMC_API set_engine_settings (device_t id, const engine_settings_t *engine_settings)
  Запись настроек мотора.
- result_t XIMC_API set_engine_settings_calb (device_t id, const engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)
  Запись настроек мотора с использованием пользовательских единиц.
- result_t XIMC_API get_engine_settings (device_t id, engine_settings_t *engine_settings)
  Чтение настроек мотора.
- result_t XIMC_API get_engine_settings_calb (device_t id, engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)
  Чтение настроек мотора с использованием пользовательских единиц.
- result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t *entype_settings)
  Запись информации о типе мотора и типе силового драйвера.
- result_t XIMC_API get_entype_settings (device_t id, entype_settings_t *entype_settings)
  Возвращает информацию о типе мотора и силового драйвера.
- result_t XIMC_API set_power_settings (device_t id, const power_settings_t *power_settings)
  Команда записи параметров питания мотора.
- result_t XIMC_API get_power_settings (device_t id, power_settings_t *power_settings)
  Команда чтения параметров питания мотора.
- result_t XIMC_API set_secure_settings (device_t id, const secure_settings_t *secure_settings)
  Команда записи установок защит.
- result_t XIMC_API get_secure_settings (device_t id, secure_settings_t *secure_settings)
  Команда записи установок защит.
- result_t XIMC_API set_edges_settings (device_t id, const edges_settings_t *edges_settings)
  Запись настроек границ и концевых выключателей.
- result_t XIMC_API set_edges_settings_calb (device_t id, const edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)
  Запись настроек границ и концевых выключателей с использованием пользовательских единиц.
- result_t XIMC_API get_edges_settings (device_t id, edges_settings_t *edges_settings)
  Чтение настроек границ и концевых выключателей.
- result_t XIMC_API get_edges_settings_calb (device_t id, edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)

Чтение настроек границ и концевых выключателей с использованием пользовательских единиц.

- result_t XIMC_API set_pid_settings (device_t id, const pid_settings_t *pid_settings)
  Запись ПИД-коэффициентов.
- result_t XIMC_API get_pid_settings (device_t id, pid_settings_t *pid_settings)
  Чтение ПИД-коэффициентов.
- result_t XIMC_API set_sync_in_settings (device_t id, const sync_in_settings_t *sync_in_settings)
  Запись настроек для входного импульса синхронизации.
- result_t XIMC_API set_sync_in_settings_calb (device_t id, const sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)
  Запись настроек для входного импульса синхронизации с использованием пользовательских единиц.
- result_t XIMC_API get_sync_in_settings (device_t id, sync_in_settings_t *sync_in_settings)
  Чтение настроек для входного импульса синхронизации.
- result_t XIMC_API get_sync_in_settings_calb (device_t id, sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)
  Чтение настроек для входного импульса синхронизации с использованием пользовательских единиц.
- result_t XIMC_API set_sync_out_settings (device_t id, const sync_out_settings_t *sync_out_settings)
  Запись настроек для выходного импульса синхронизации.
- result_t XIMC_API set_sync_out_settings_calb (device_t id, const sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)
  Запись настроек для выходного импульса синхронизации с использованием пользовательских единиц.
- result_t XIMC_API get_sync_out_settings (device_t id, sync_out_settings_t *sync_out_settings)
  Чтение настроек для выходного импульса синхронизации.
- result_t XIMC_API get_sync_out_settings_calb (device_t id, sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)
  Чтение настроек для выходного импульса синхронизации с использованием пользовательских единиц.
- result_t XIMC_API set_extio_settings (device_t id, const extio_settings_t *extio_settings)
  Команда записи параметров настройки режимов внешнего ввода/вывода.
- result_t XIMC_API get_extio_settings (device_t id, extio_settings_t *extio_settings)
  Команда чтения параметров настройки режимов внешнего ввода/вывода.
- result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t *brake_settings)
  Запись настроек управления тормозом.
- result_t XIMC_API get_brake_settings (device_t id, brake_settings_t *brake_settings)
  Чтение настроек управления тормозом.
- result_t XIMC_API set_control_settings (device_t id, const control_settings_t *control_settings)
  Запись настроек управления мотором.
- result_t XIMC_API set_control_settings_calb (device_t id, const control_settings_calb_t *control_settings_calb, const calibration_t *calibration)
  Запись настроек управления мотором с использованием пользовательских единиц.
- result_t XIMC_API get_control_settings (device_t id, control_settings_t *control_settings)
  Чтение настроек управления мотором.
- result_t XIMC_API get_control_settings_calb (device_t id, control_settings_calb_t *control_settings_calb, const calibration_t *calibration)
  Чтение настроек управления мотором с использованием пользовательских единиц.
- result_t XIMC_API set_joystick_settings (device_t id, const joystick_settings_t *joystick_settings)
  Запись настроек джойстика.
- result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t *joystick_settings)
  Чтение настроек джойстика.
- result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t *ctp_settings)

Запись настроек контроля позиции(для шагового двигателя).

- result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t *ctp_settings)
  Чтение настроек контроля позиции(для шагового двигателя).
- result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t *uart_settings)
  Команда записи настроек UART.
- result_t XIMC_API get_uart_settings (device_t id, uart_settings_t *uart_settings)
  Команда чтения настроек UART.
- result_t XIMC_API set_network_settings (device_t id, const network_settings_t *network_settings)
  Команда записи сетевых настроек.
- result_t XIMC_API get_network_settings (device_t id, network_settings_t *network_settings)
  Команда чтения сетевых настроек.
- result_t XIMC_API set_password_settings (device_t id, const password_settings_t *password_settings)
  Команда записи пароля к веб-странице.
- result_t XIMC_API get_password_settings (device_t id, password_settings_t *password_settings)
  Команда чтения пароля к веб-странице.
- result_t XIMC_API set_calibration_settings (device_t id, const calibration_settings_t *calibration_settings)
  Команда записи калибровочных коэффициентов.
- result_t XIMC_API get_calibration_settings (device_t id, calibration_settings_t *calibration_settings)
  Команда чтения калибровочных коэффициентов.
- result_t XIMC_API set_controller_name (device_t id, const controller_name_t *controller_name)
  Запись пользовательского имени контроллера и настроек в FRAM.
- result_t XIMC_API get_controller_name (device_t id, controller_name_t *controller_name)
  Чтение пользовательского имени контроллера и настроек из FRAM.
- result_t XIMC_API set_nonvolatile_memory (device_t id, const nonvolatile_memory_t *nonvolatile_memory)
  Запись пользовательских данных во FRAM.
- result_t XIMC_API get_nonvolatile_memory (device_t id, nonvolatile_memory_t *nonvolatile_memory)
  Чтение пользовательских данных из FRAM.
- result_t XIMC_API set_emf_settings (device_t id, const emf_settings_t *emf_settings)
  Запись электромеханических настроек шагового двигателя.
- result_t XIMC_API get_emf_settings (device_t id, emf_settings_t *emf_settings)
  Чтение электромеханических настроек шагового двигателя.
- result_t XIMC_API set_engine_advanced_setup (device_t id, const engine_advanced_setup_t *engine_advanced_setup)
  Запись расширенных настроек.
- result_t XIMC_API get_engine_advanced_setup (device_t id, engine_advanced_setup_t *engine_advanced_setup)
  Чтение расширенных настроек.
- result_t XIMC_API set_extended_settings (device_t id, const extended_settings_t *extended_settings)
  Запись расширенных настроек.
- result_t XIMC_API get_extended_settings (device_t id, extended_settings_t *extended_settings)
  Чтение расширенных настроек.

Группа команд управления движением

- result_t XIMC_API command_stop (device_t id)

Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим "удержания" дезактивируется для DC-двигателей, удержание тока в обмотках для шаговых двигателей (с учётом Power management настроек).

- result_t XIMC_API command_power_off (device_t id)

  Немедленное отключение питания двигателя вне зависимости от его состояния.
- result_t XIMC_API command_move (device_t id, int Position, int uPosition)

  При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в полях Position, uPosition.
- result_t XIMC_API command_move_calb (device_t id, float Position, const calibration_t *calibration)

  Перемещение в позицию с использованием пользовательских единиц.
- result_t XIMC_API command_movr (device_t id, int DeltaPosition, int uDeltaPosition)

  Перемещение на заданное смещение.
- result_t XIMC_API command_movr_calb (device_t id, float DeltaPosition, const calibration_t *calibration)

  Перемещение на заданное смещение с использованием пользовательских единиц.
- result_t XIMC_API command_home (device_t id)

  Движение в домашнюю позицию.
- result_t XIMC_API command_left (device_t id)

  При получении команды "left" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.
- result_t XIMC_API command_right (device_t id)

  При получении команды "rigt" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.
- result_t XIMC_API command_loft (device_t id)

  При получении команды "loft" двигатель смещается из текущей точки на расстояние Antiplay, заданное в настройках мотора (engine_settings), затем двигается в ту же точку.
- result_t XIMC_API command_sstp (device_t id)

  Плавная остановка.
- result_t XIMC_API get_position (device_t id, get_position_t *the_get_position)

  Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.
- result_t XIMC_API get_position_calb (device_t id, get_position_calb_t *the_get_position_calb, const calibration_t *calibration)

  Считывает значение положения в пользовательских единицах для шагового двигателя и в шагах энкодера всех двигателей.
- result_t XIMC_API set_position (device_t id, const set_position_t *the_set_position)

  Устанавливает произвольное значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера для всех двигателей.
- result_t XIMC_API set_position_calb (device_t id, const set_position_calb_t *the_set_position_calb, const calibration_t *calibration)

  Устанавливает произвольное значение положения и значение энкодера всех двигателей с использованием пользовательских единиц.
- result_t XIMC_API command_zero (device_t id)

  Устанавливает текущую позицию равной 0.

Группа команд сохранения и загрузки настроек

- result_t XIMC_API command_save_settings (device_t id)

  При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.
- result_t XIMC_API command_read_settings (device_t id)

  Чтение всех настроек контроллера из flash-памяти в оперативную, заменяя текущие настройки.
- result_t XIMC_API command_save_robust_settings (device_t id)

  При получении команды контроллер выполняет операцию сохранения важных настроек (калибровочные коэффициенты и т.
- result_t XIMC_API command_read_robust_settings (device_t id)

Чтение важных настроек (калибровочные коэффициенты и т.
- result_t XIMC_API command_eesave_settings (device_t id)
  Запись настроек контроллера в EEPROM память позиционера.
- result_t XIMC_API command_eeread_settings (device_t id)
  Чтение настроек контроллера из EEPROM памяти позиционера.
- result_t XIMC_API command_start_measurements (device_t id)
  Старт измерения и буферизации скорости, ошибки следования.
- result_t XIMC_API get_measurements (device_t id, measurements_t *measurements)
  Команда чтения буфера данных для построения графиков скорости и ошибки следования.
- result_t XIMC_API get_chart_data (device_t id, chart_data_t *chart_data)
  Команда чтения состояния обмоток и других нечасто используемых данных.
- result_t XIMC_API get_serial_number (device_t id, unsigned int *SerialNumber)
  Чтение серийного номера контроллера.
- result_t XIMC_API get_firmware_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)
  Чтение номера версии прошивки контроллера.
- result_t XIMC_API service_command_updf (device_t id)
  Команда переводит контроллер в режим обновления прошивки.

Группа сервисных команд

- result_t XIMC_API set_serial_number (device_t id, const serial_number_t *serial_number)
  Запись серийного номера и версии железа во flash память контроллера.
- result_t XIMC_API get_analog_data (device_t id, analog_data_t *analog_data)
  Чтение аналоговых данных, содержащих данные с АЦП и нормированные значения величин.
- result_t XIMC_API get_debug_read (device_t id, debug_read_t *debug_read)
  Чтение данных из прошивки для отладки и поиска неисправностей.
- result_t XIMC_API set_debug_write (device_t id, const debug_write_t *debug_write)
  Запись данных в прошивку для отладки и поиска неисправностей.

Группа команд работы с EEPROM подвижки

- result_t XIMC_API set_stage_name (device_t id, const stage_name_t *stage_name)
  Запись пользовательского имени подвижки в EEPROM.
- result_t XIMC_API get_stage_name (device_t id, stage_name_t *stage_name)
  Чтение пользовательского имени подвижки из EEPROM.
- result_t XIMC_API set_stage_information (device_t id, const stage_information_t *stage_information)
  Устарело.
- result_t XIMC_API get_stage_information (device_t id, stage_information_t *stage_information)
  Устарело.
- result_t XIMC_API set_stage_settings (device_t id, const stage_settings_t *stage_settings)
  Устарело.
- result_t XIMC_API get_stage_settings (device_t id, stage_settings_t *stage_settings)
  Устарело.
- result_t XIMC_API set_motor_information (device_t id, const motor_information_t *motor_information)
  Устарело.
- result_t XIMC_API get_motor_information (device_t id, motor_information_t *motor_information)
  Устарело.
- result_t XIMC_API set_motor_settings (device_t id, const motor_settings_t *motor_settings)
  Устарело.
- result_t XIMC_API get_motor_settings (device_t id, motor_settings_t *motor_settings)
  Устарело.

- result_t XIMC_API set_encoder_information (device_t id, const encoder_information_t *encoder_information)
  Устарело.
- result_t XIMC_API get_encoder_information (device_t id, encoder_information_t *encoder_information)
  Устарело.
- result_t XIMC_API set_encoder_settings (device_t id, const encoder_settings_t *encoder_settings)
  Устарело.
- result_t XIMC_API get_encoder_settings (device_t id, encoder_settings_t *encoder_settings)
  Устарело.
- result_t XIMC_API set_hallsensor_information (device_t id, const hallsensor_information_t *hallsensor_information)
  Устарело.
- result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t *hallsensor_information)
  Устарело.
- result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t *hallsensor_settings)
  Устарело.
- result_t XIMC_API get_hallsensor_settings (device_t id, hallsensor_settings_t *hallsensor_settings)
  Устарело.
- result_t XIMC_API set_gear_information (device_t id, const gear_information_t *gear_information)
  Устарело.
- result_t XIMC_API get_gear_information (device_t id, gear_information_t *gear_information)
  Устарело.
- result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t *gear_settings)
  Устарело.
- result_t XIMC_API get_gear_settings (device_t id, gear_settings_t *gear_settings)
  Устарело.
- result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t *accessories_settings)
  Устарело.
- result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t *accessories_settings)
  Устарело.
- result_t XIMC_API get_bootloader_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)
  Чтение номера версии загрузчика контроллера.
- result_t XIMC_API get_init_random (device_t id, init_random_t *init_random)
  Чтение случайного числа из контроллера.
- result_t XIMC_API get_globally_unique_identifier (device_t id, globally_unique_identifier_t *globally_unique_identifier)
  Считывает уникальный идентификатор каждого чипа, это значение не является случайным.
- result_t XIMC_API goto_firmware (device_t id, uint8_t *ret)
  Перезагрузка в прошивку в контроллере
- result_t XIMC_API has_firmware (const char *uri, uint8_t *ret)
  Проверка наличия прошивки в контроллере
- result_t XIMC_API command_update_firmware (const char *uri, const uint8_t *data, uint32_t data_size)
  Обновление прошивки.
- result_t XIMC_API write_key (const char *uri, uint8_t *key)
  Запись ключа защиты Функция используется только производителем.
- result_t XIMC_API command_reset (device_t id)
  Перезагрузка контроллера.
- result_t XIMC_API command_clear_fram (device_t id)
  Очистка FRAM памяти контроллера.

Управление устройством

Функции поиска и открытия/закрытия устройств

- typedef char * pchar

  Не обращайте на меня внимание
- typedef void(XIMC_CALLCONV * logging_callback_t) (int loglevel, const wchar_t *message, void *user_data)

  Прототип функции обратного вызова для логирования
- device_t XIMC_API open_device (const char *uri)

  Открывает устройство по имени uri и возвращает идентификатор, который будет использоваться для обращения к устройству.
- result_t XIMC_API close_device (device_t *id)

  Закрывает устройство
- result_t XIMC_API set_correction_table (device_t id, const char *namefile)

  Команда загрузки корректирующей таблицы из текстового файла.
- result_t XIMC_API probe_device (const char *uri)

  Проверяет, является ли устройство с уникальным идентификатором uri XIMC-совместимым.
- device_enumeration_t XIMC_API enumerate_devices (int enumerate_flags, const char *hints)

  Поиск и составление списка доступных устройств.
- result_t XIMC_API free_enumerate_devices (device_enumeration_t device_enumeration)

  Освобождает память, выделенную enumerate_devices.
- int XIMC_API get_device_count (device_enumeration_t device_enumeration)

  Возвращает количество подключенных устройств.
- pchar XIMC_API get_device_name (device_enumeration_t device_enumeration, int device_index)

  Возвращает имя подключенного устройства из перечисления устройств.
- result_t XIMC_API get_enumerate_device_serial (device_enumeration_t device_enumeration, int device_index, uint32_t *serial)

  Возвращает серийный номер подключенного устройства из перечисления устройств.
- result_t XIMC_API get_enumerate_device_information (device_enumeration_t device_enumeration, int device_index, device_information_t *device_information)

  Возвращает информацию о подключенном устройстве из перечисления устройств.
- result_t XIMC_API get_enumerate_device_controller_name (device_enumeration_t device_enumeration, int device_index, controller_name_t *controller_name)

  Возвращает имя подключенного устройства из перечисления устройств.
- result_t XIMC_API get_enumerate_device_stage_name (device_enumeration_t device_enumeration, int device_index, stage_name_t *stage_name)

  Возвращает имя подвижки для подключенного устройства из перечисления устройств.
- result_t XIMC_API get_enumerate_device_network_information (device_enumeration_t device_enumeration, int device_index, device_network_information_t *device_network_information)

  Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.
- result_t XIMC_API reset_locks ()

  Сбрасывает ошибку неправильной передачи данных.
- void XIMC_API msec_sleep (unsigned int msec)

  Приостанавливает работу на указанное время
- void XIMC_API ximc_version (char *version)

  Возвращает версию библиотеки
- void XIMC_API logging_callback_stderr_wide (int loglevel, const wchar_t *message, void *user_data)

  Простая функция логирования на stderr в широких символах

- void XIMC_API logging_callback_stderr_narrow (int loglevel, const wchar_t *message, void *user_data)

  Простая функция логирования на stderr в узких (однобайтных) символах
- void XIMC_API set_logging_callback (logging_callback_t logging_callback, void *user_data)

  Устанавливает функцию обратного вызова для логирования.
- result_t XIMC_API get_status (device_t id, status_t *status)

  Возвращает информацию о текущем состоянии устройства.
- result_t XIMC_API get_status_calb (device_t id, status_calb_t *status, const calibration_t *calibration)

  Возвращает информацию о текущем состоянии устройства.
- result_t XIMC_API get_device_information (device_t id, device_information_t *device_information)

  Возвращает информацию об устройстве.
- result_t XIMC_API command_wait_for_stop (device_t id, uint32_t refresh_interval_ms)

  Ожидание остановки контроллера
- result_t XIMC_API command_homezero (device_t id)

  Запустить процедуру поиска домашней позиции, подождать её завершения и обнулить позицию в конце.

6.1.1 Подробное описание

Заголовочный файл для библиотеки libximc.

6.1.2 Макросы

6.1.2.1 ALARM_ON_DRIVER_OVERHEATING

```
#define ALARM_ON_DRIVER_OVERHEATING 0x01
```

Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера.

Иначе - игнорировать подступающий перегрев с драйвера.

6.1.2.2 BACK_EMF_INDUCTANCE_AUTO

```
#define BACK_EMF_INDUCTANCE_AUTO 0x01
```

Флаг автоопределения индуктивности обмоток двигателя.

6.1.2.3 BACK_EMF_KM_AUTO

```
#define BACK_EMF_KM_AUTO 0x04
```

Флаг автоопределения электромеханического коэффициента двигателя.

6.1.2.4 BACK_EMF_RESISTANCE_AUTO

```
#define BACK_EMF_RESISTANCE_AUTO 0x02
```

Флаг автоопределения сопротивления обмоток двигателя.

6.1.2.5 BORDER_IS_ENCODER

```
#define BORDER_IS_ENCODER 0x01
```

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

Если флаг сброшен, границы определяются концевыми выключателями.

6.1.2.6 BORDER_STOP_LEFT

```
#define BORDER_STOP_LEFT 0x02
```

Если флаг установлен, мотор останавливается при достижении левой границы.

6.1.2.7 BORDER_STOP_RIGHT

```
#define BORDER_STOP_RIGHT 0x04
```

Если флаг установлен, мотор останавливается при достижении правой границы.

6.1.2.8 BORDERS_SWAP_MISSET_DETECTION

```
#define BORDERS_SWAP_MISSET_DETECTION 0x08
```

Если флаг установлен, мотор останавливается по достижении любой из границ.

Нужен для предотвращения поломки двигателя при неправильных настройках концевых выключателей

6.1.2.9 BRAKE_ENABLED

```
#define BRAKE_ENABLED 0x01
```

Управление тормозом включено, если флаг установлен.

6.1.2.10 BRAKE_ENG_PWROFF

```
#define BRAKE_ENG_PWROFF 0x02
```

Тормоз отключает питание шагового мотора, если флаг установлен.

6.1.2.11 BRAKING_OVERVOLTAGE_PROTECTION

```
#define BRAKING_OVERVOLTAGE_PROTECTION 0x20
```

Если флаг установлен, то микропрограмма контроллера будет замыкать нижние ключи H-моста, отсоединяя мотор от цепи питания, при перенапряжении.

6.1.2.12 CONTROL_BTN_LEFT_PUSHED_OPEN

```
#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04
```

Нажатая левая кнопка соответствует открытому контакту, если этот флаг установлен.

6.1.2.13 CONTROL_BTN_RIGHT_PUSHED_OPEN

```
#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08
```

Нажатая правая кнопка соответствует открытому контакту, если этот флаг установлен.

6.1.2.14 CONTROL_MODE_BITS

```
#define CONTROL_MODE_BITS 0x03
```

Биты управления мотором с помощью джойстика или кнопок влево/вправо.

6.1.2.15 CONTROL_MODE_JOY

```
#define CONTROL_MODE_JOY 0x01
```

Управление с помощью джойстика.

6.1.2.16 CONTROL_MODE_LR

```
#define CONTROL_MODE_LR 0x02
```

Управление с помощью кнопок влево/вправо.

6.1.2.17 CONTROL_MODE_OFF

```
#define CONTROL_MODE_OFF 0x00
```

Управление отключено.

6.1.2.18 CTP_ALARM_ON_ERROR

```
#define CTP_ALARM_ON_ERROR 0x04
```

Войти в состояние ALARM при расхождении позиции, если флаг установлен.

6.1.2.19 CTP_BASE

```
#define CTP_BASE 0x02
```

Управление положением основано на датчике вращения, если установлен этот флаг; в противном случае - на энкодере.

6.1.2.20 CTP_ENABLED

```
#define CTP_ENABLED 0x01
```

Контроль позиции включен, если флаг установлен.

6.1.2.21 CTP_ERROR_CORRECTION

```
#define CTP_ERROR_CORRECTION 0x10
```

Корректировать ошибки, возникающие при проскальзывании, если флаг установлен.

Работает только с энкодером. Несовместимо с флагом CTP_ALARM_ON_ERROR.

6.1.2.22 DRIVER_TYPE_EXTERNAL

```
#define DRIVER_TYPE_EXTERNAL 0x03
```

Внешний силовой драйвер.

Поддержка этой функции зависит от модификации контроллера.

6.1.2.23 DRIVER_TYPE_INTEGRATE

```
#define DRIVER_TYPE_INTEGRATE 0x02
```

Силовой драйвер с использованием ключей, интегрированных в микросхему.

6.1.2.24 EEPROM_PRECEDENCE

```
#define EEPROM_PRECEDENCE 0x01
```

Если флаг установлен, то настройки в EEPROM подвижки имеют приоритет над текущими настройками и заменяют их при обнаружении EEPROM.

6.1.2.25 ENC_STATE_ABSENT

```
#define ENC_STATE_ABSENT 0x00
```

Энкодер не подключен.

6.1.2.26 ENC_STATE_MALFUNC

```
#define ENC_STATE_MALFUNC 0x02
```

Энкодер подключен и неисправен.

6.1.2.27 ENC_STATE_OK

```
#define ENC_STATE_OK 0x04
```

Энкодер подключен и работает должным образом.

6.1.2.28 ENC_STATE_REVERS

```
#define ENC_STATE_REVERS 0x03
```

Энкодер подключен и исправен, но считает в другую сторону.

6.1.2.29 ENC_STATE_UNKNOWN

```
#define ENC_STATE_UNKNOWN 0x01
```

Состояние энкодера неизвестно.

6.1.2.30 ENDER_SW1_ACTIVE_LOW

```
#define ENDER_SW1_ACTIVE_LOW 0x02
```

1 - Концевой переключатель, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте.

6.1.2.31 ENDER_SW2_ACTIVE_LOW

```
#define ENDER_SW2_ACTIVE_LOW 0x04
```

1 - Концевой переключатель, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте.

6.1.2.32 ENDER_SWAP

```
#define ENDER_SWAP 0x01
```

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

6.1.2.33 ENGINE_ACCEL_ON

```
#define ENGINE_ACCEL_ON 0x010
```

Ускорение.

Если флаг установлен, движение происходит с ускорением.

6.1.2.34 ENGINE_ANTIPLAY

```
#define ENGINE_ANTIPLAY 0x008
```

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.

6.1.2.35 ENGINE_CURRENT_AS_RMS

```
#define ENGINE_CURRENT_AS_RMS 0x002
```

Флаг интерпретации значения тока.

Если флаг снят, то задаваемое значение тока интерпретируется как максимальная амплитуда тока. Если флаг установлен, то задаваемое значение тока интерпретируется как среднеквадратичное значение тока (для шагового) или как значение тока, посчитанное из максимального тепловыделения (BLDC).

6.1.2.36 ENGINE_LIMIT_CURR

```
#define ENGINE_LIMIT_CURR 0x040
```

Номинальный ток мотора.

Если флаг установлен, ток через мотор ограничивается заданным номинальным значением (используется только с DC-двигателем).

6.1.2.37 ENGINE_LIMIT_RPM

```
#define ENGINE_LIMIT_RPM 0x080
```

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

6.1.2.38 ENGINE_LIMIT_VOLT

```
#define ENGINE_LIMIT_VOLT 0x020
```

Номинальное напряжение мотора.

Если флаг установлен, напряжение на моторе ограничивается заданным номинальным значением (используется только с DC-двигателем).

6.1.2.39 ENGINE_MAX_SPEED

```
#define ENGINE_MAX_SPEED 0x004
```

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

6.1.2.40 ENGINE_REVERSE

```
#define ENGINE_REVERSE 0x001
```

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.

6.1.2.41 ENGINE_TYPE_2DC

```
#define ENGINE_TYPE_2DC 0x02
```

Два мотора постоянного тока, что приводит к эмуляции двух контроллеров.

6.1.2.42 ENGINE_TYPE_BRUSHLESS

```
#define ENGINE_TYPE_BRUSHLESS 0x05
```

Бесщеточный мотор.

6.1.2.43 ENGINE_TYPE_DC

```
#define ENGINE_TYPE_DC 0x01
```

Мотор постоянного тока.

6.1.2.44 ENGINE_TYPE_NONE

```
#define ENGINE_TYPE_NONE 0x00
```

Это значение не нужно использовать.

6.1.2.45 ENGINE_TYPE_STEP

```
#define ENGINE_TYPE_STEP 0x03
```

Шаговый мотор.

6.1.2.46 ENGINE_TYPE_TEST

```
#define ENGINE_TYPE_TEST 0x04
```

Продолжительность включения фиксирована.

Используется только производителем.

6.1.2.47 ENGINE_USE_PEAK_CURRENT

```
#define ENGINE_USE_PEAK_CURRENT 0x100
```

Разрешить использование пикового тока.

6.1.2.48 ENUMERATE_PROBE

```
#define ENUMERATE_PROBE 0x01
```

Проверять, является ли устройство XIMC-совместимым.

Будьте осторожны с этим флагом, т.к. он отправляет данные в устройство.

6.1.2.49 EXTIO_SETUP_INVERT

```
#define EXTIO_SETUP_INVERT 0x02
```

Если флаг установлен, то нули считаются активным состоянием выхода, а спадающие фронты - как момент подачи входного сигнала.

6.1.2.50 EXTIO_SETUP_MODE_IN_ALARM

```
#define EXTIO_SETUP_MODE_IN_ALARM 0x05
```

Войти в состояние ALARM при переходе сигнала в активное состояние.

6.1.2.51 EXTIO_SETUP_MODE_IN_BITS

```
#define EXTIO_SETUP_MODE_IN_BITS 0x0F
```

Биты, отвечающие за поведение при переходе сигнала в активное состояние.

6.1.2.52 EXTIO_SETUP_MODE_IN_HOME

```
#define EXTIO_SETUP_MODE_IN_HOME 0x04
```

Выполняется команда HOME.

6.1.2.53 EXTIO_SETUP_MODE_IN_MOVR

```
#define EXTIO_SETUP_MODE_IN_MOVR 0x03
```

Выполняется команда MOVR с последними настройками.

6.1.2.54 EXTIO_SETUP_MODE_IN_NOP

```
#define EXTIO_SETUP_MODE_IN_NOP 0x00
```

Ничего не делать.

6.1.2.55 EXTIO_SETUP_MODE_IN_PWOF

```
#define EXTIO_SETUP_MODE_IN_PWOF 0x02
```

Выполняет команду PWOF, обесточивая обмотки двигателя.

6.1.2.56 EXTIO_SETUP_MODE_IN_STOP

```
#define EXTIO_SETUP_MODE_IN_STOP 0x01
```

По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).

6.1.2.57 EXTIO_SETUP_MODE_OUT_ALARM

```
#define EXTIO_SETUP_MODE_OUT_ALARM 0x30
```

Ножка находится в активном состоянии при нахождении в состоянии ALARM.

6.1.2.58 EXTIO_SETUP_MODE_OUT_BITS

```
#define EXTIO_SETUP_MODE_OUT_BITS 0xF0
```

Биты выбора поведения на выходе.

6.1.2.59 EXTIO_SETUP_MODE_OUT_MOTOR_ON

```
#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40
```

Ножка находится в активном состоянии при подаче питания на обмотки.

6.1.2.60 EXTIO_SETUP_MODE_OUT_MOVING

```
#define EXTIO_SETUP_MODE_OUT_MOVING 0x20
```

Ножка находится в активном состоянии при движении.

6.1.2.61 EXTIO_SETUP_MODE_OUT_OFF

```
#define EXTIO_SETUP_MODE_OUT_OFF 0x00
```

Ножка всегда в неактивном состоянии.

6.1.2.62 EXTIO_SETUP_MODE_OUT_ON

```
#define EXTIO_SETUP_MODE_OUT_ON 0x10
```

Ножка всегда в активном состоянии.

6.1.2.63 EXTIO_SETUP_OUTPUT

```
#define EXTIO_SETUP_OUTPUT 0x01
```

Если флаг установлен, то ножка в состоянии вывода, иначе - ввода.

6.1.2.64 FEEDBACK_EMF

```
#define FEEDBACK_EMF 0x04
```

Обратная связь по ЭДС.

6.1.2.65 FEEDBACK_ENC_ADAPTIVE_HOLDING

```
#define FEEDBACK_ENC_ADAPTIVE_HOLDING 0x02
```

Включает алгоритм адаптивного удержания.

6.1.2.66 FEEDBACK_ENC_FILTER_BITS

```
#define FEEDBACK_ENC_FILTER_BITS 0x30
```

Биты, отвечающие за настройку внутреннего фильтра энкодерного сигнала.

6.1.2.67 FEEDBACK_ENC_FILTER_MEDIUM

```
#define FEEDBACK_ENC_FILTER_MEDIUM 0x20
```

Средняя фильтрация шумов: максимальная частота сигнала энкодера 1 МГц.

6.1.2.68 FEEDBACK_ENC_FILTER_NONE

```
#define FEEDBACK_ENC_FILTER_NONE 0x00
```

Выключает внутренний фильтр сигнала энкодера.

6.1.2.69 FEEDBACK_ENC_FILTER_STRONG

```
#define FEEDBACK_ENC_FILTER_STRONG 0x30
```

Сильная фильтрация шумов: максимальная частота сигнала энкодера 300 кГц.

6.1.2.70 FEEDBACK_ENC_FILTER_WEAK

```
#define FEEDBACK_ENC_FILTER_WEAK 0x10
```

Слабая фильтрация шумов: максимальная частота сигнала энкодера 3 МГц.

6.1.2.71 FEEDBACK_ENC_REVERSE

```
#define FEEDBACK_ENC_REVERSE 0x01
```

Обратный счет у энкодера.

6.1.2.72 FEEDBACK_ENC_TYPE_AUTO

```
#define FEEDBACK_ENC_TYPE_AUTO 0x00
```

Определяет тип энкодера автоматически.

6.1.2.73 FEEDBACK_ENC_TYPE_BITS

```
#define FEEDBACK_ENC_TYPE_BITS 0xC0
```

Биты, отвечающие за тип энкодера.

6.1.2.74 FEEDBACK_ENC_TYPE_DIFFERENTIAL

```
#define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80
```

Дифференциальный энкодер.

6.1.2.75 FEEDBACK_ENC_TYPE_SINGLE_ENDED

```
#define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40
```

Недифференциальный энкодер.

6.1.2.76 FEEDBACK_ENCODER

```
#define FEEDBACK_ENCODER 0x01
```

Обратная связь с помощью энкодера.

6.1.2.77 FEEDBACK_ENCODER_MEDIATED

```
#define FEEDBACK_ENCODER_MEDIATED 0x06
```

Обратная связь по энкодеру, опосредованному относительно двигателя механической передачей (например, винтовой передачей).

6.1.2.78 FEEDBACK_NONE

```
#define FEEDBACK_NONE 0x05
```

Обратная связь отсутствует.

6.1.2.79 HOME_DIR_FIRST

```
#define HOME_DIR_FIRST 0x001
```

Определяет направление первоначального движения мотора после поступления команды HOME.

Если флаг установлен - вправо; иначе - влево.

6.1.2.80 HOME_DIR_SECOND

```
#define HOME_DIR_SECOND 0x002
```

Определяет направление второго движения мотора.

Если флаг установлен - вправо; иначе - влево.

6.1.2.81 HOME_HALF_MV

```
#define HOME_HALF_MV 0x008
```

Если флаг установлен, сигналы остановки игнорируются в первой половине второго движения.

6.1.2.82 HOME_MV_SEC_EN

```
#define HOME_MV_SEC_EN 0x004
```

Если флаг установлен, реализуется второй этап доводки в домашнюю позицию; иначе - этап пропускается.

6.1.2.83 HOME_STOP_FIRST_BITS

```
#define HOME_STOP_FIRST_BITS 0x030
```

Биты, отвечающие за выбор сигнала завершения первого движения.

6.1.2.84 HOME_STOP_FIRST_LIM

```
#define HOME_STOP_FIRST_LIM 0x030
```

Первое движение завершается по сигналу с концевого переключателя.

6.1.2.85 HOME_STOP_FIRST_REV

```
#define HOME_STOP_FIRST_REV 0x010
```

Первое движение завершается по сигналу с Revolution sensor.

6.1.2.86 HOME_STOP_FIRST_SYN

```
#define HOME_STOP_FIRST_SYN 0x020
```

Первое движение завершается по сигналу со входа синхронизации.

6.1.2.87 HOME_STOP_SECOND_BITS

```
#define HOME_STOP_SECOND_BITS 0x0C0
```

Биты, отвечающие за выбор сигнала завершения второго движения.

6.1.2.88 HOME_STOP_SECOND_LIM

```
#define HOME_STOP_SECOND_LIM 0x0C0
```

Второе движение завершается по сигналу с концевого переключателя.

6.1.2.89 HOME_STOP_SECOND_REV

```
#define HOME_STOP_SECOND_REV 0x040
```

Второе движение завершается по сигналу с Revolution sensor.

6.1.2.90 HOME_STOP_SECOND_SYN

```
#define HOME_STOP_SECOND_SYN 0x080
```

Второе движение завершается по сигналу со входа синхронизации.

6.1.2.91 HOME_USE_FAST

```
#define HOME_USE_FAST 0x100
```

Если флаг установлен, используется быстрый поиск домашней позиции; иначе - традиционный.

6.1.2.92 JOY_REVERSE

```
#define JOY_REVERSE 0x01
```

Реверс воздействия джойстика.

Отклонение джойстика к большим значениям приводит к отрицательной скорости и наоборот.

6.1.2.93 LOW_UPWR_PROTECTION

```
#define LOW_UPWR_PROTECTION 0x02
```

Если флаг установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.

6.1.2.94 LS_SHORTED

```
#define LS_SHORTED 0x10
```

Если флаг установлен, то концевые переключатели замкнуты.

6.1.2.95 MICROSTEP_MODE_FRAC_128

```
#define MICROSTEP_MODE_FRAC_128 0x08
```

Деление шага 1/128.

6.1.2.96 MICROSTEP_MODE_FRAC_16

```
#define MICROSTEP_MODE_FRAC_16 0x05
```

Деление шага 1/16.

6.1.2.97 MICROSTEP_MODE_FRAC_2

```
#define MICROSTEP_MODE_FRAC_2 0x02
```

Деление шага 1/2.

6.1.2.98 MICROSTEP_MODE_FRAC_256

```
#define MICROSTEP_MODE_FRAC_256 0x09
```

Деление шага 1/256.

6.1.2.99 MICROSTEP_MODE_FRAC_32

```
#define MICROSTEP_MODE_FRAC_32 0x06
```

Деление шага 1/32.

6.1.2.100 MICROSTEP_MODE_FRAC_4

```
#define MICROSTEP_MODE_FRAC_4 0x03
```

Деление шага 1/4.

6.1.2.101 MICROSTEP_MODE_FRAC_64

```
#define MICROSTEP_MODE_FRAC_64 0x07
```

Деление шага 1/64.

6.1.2.102 MICROSTEP_MODE_FRAC_8

```
#define MICROSTEP_MODE_FRAC_8 0x04
```

Деление шага 1/8.

6.1.2.103 MICROSTEP_MODE_FULL

```
#define MICROSTEP_MODE_FULL 0x01
```

Полношаговый режим.

6.1.2.104 MOVE_STATE_ANTIPLAY

```
#define MOVE_STATE_ANTIPLAY 0x04
```

Выполняется компенсация люфта, если флаг установлен.

6.1.2.105 MOVE_STATE_MOVING

```
#define MOVE_STATE_MOVING 0x01
```

Если флаг установлен, то контроллер пытается вращать двигателем.

Не используйте этот флаг для ожидания завершения команды движения. Вместо него используйте MVCMD_RUNNING из поля MvCmdSts.

6.1.2.106 MOVE_STATE_TARGET_SPEED

```
#define MOVE_STATE_TARGET_SPEED 0x02
```

Флаг устанавливается при достижении заданной скорости.

6.1.2.107 MVCMD_ERROR

```
#define MVCMD_ERROR 0x40
```

Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно).

Имеет смысл если MVCMD_RUNNING указывает на завершение движения.

6.1.2.108 MVCMD_HOME

```
#define MVCMD_HOME 0x06
```

Команда home.

6.1.2.109 MVCMD_LEFT

```
#define MVCMD_LEFT 0x03
```

Команда left.

6.1.2.110 MVCMD_LOFT

```
#define MVCMD_LOFT 0x07
```

Команда loft.

6.1.2.111 MVCMD_MOVE

```
#define MVCMD_MOVE 0x01
```

Команда move.

6.1.2.112 MVCMD_MOVR

```
#define MVCMD_MOVR 0x02
```

Команда movr.

6.1.2.113 MVCMD_NAME_BITS

```
#define MVCMD_NAME_BITS 0x3F
```

Битовая маска активной команды.

6.1.2.114 MVCMD_RIGHT

```
#define MVCMD_RIGHT 0x04
```

Команда rigt.

6.1.2.115 MVCMD_RUNNING

```
#define MVCMD_RUNNING 0x80
```

Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).

6.1.2.116 MVCMD_SSTP

```
#define MVCMD_SSTP 0x08
```

Команда плавной остановки(SSTP).

6.1.2.117 MVCMD_STOP

```
#define MVCMD_STOP 0x05
```

Команда stop.

6.1.2.118 MVCMD_UKNWN

```
#define MVCMD_UKNWN 0x00
```

Неизвестная команда.

6.1.2.119 POWER_OFF_ENABLED

```
#define POWER_OFF_ENABLED 0x02
```

Если флаг установлен, снять напряжение с обмоток по прошествии PowerOffDelay.

Иначе - не снимать.

6.1.2.120 POWER_REDUCT_ENABLED

```
#define POWER_REDUCT_ENABLED 0x01
```

Если флаг установлен, уменьшить ток по прошествии CurrReductDelay.

Иначе - не уменьшать.

6.1.2.121 POWER_SMOOTH_CURRENT

```
#define POWER_SMOOTH_CURRENT 0x04
```

Если установлен, то запитывание обмоток, снятие питания или снижение/повышение тока происходят плавно со скоростью CurrentSetTime, а только потом выполняется та задача, которая вызвала это плавное изменение.

6.1.2.122 PWR_STATE_MAX

```
#define PWR_STATE_MAX 0x05
```

Обмотки двигателя питаются от максимального тока, который драйвер может обеспечить при этом напряжении.

6.1.2.123 PWR_STATE_NORM

```
#define PWR_STATE_NORM 0x03
```

Обмотки запитаны номинальным током.

6.1.2.124 PWR_STATE_OFF

```
#define PWR_STATE_OFF 0x01
```

Обмотки мотора разомкнуты и не управляются драйвером.

6.1.2.125 PWR_STATE_REDUCT

```
#define PWR_STATE_REDUCT 0x04
```

Обмотки намеренно запитаны уменьшенным током от рабочего для снижения потребляемой мощности.

6.1.2.126 PWR_STATE_UNKNOWN

```
#define PWR_STATE_UNKNOWN 0x00
```

Неизвестное состояние, которое не должно никогда реализовываться.

6.1.2.127 REV_SENS_INV

```
#define REV_SENS_INV 0x08
```

Сенсор считается активным, когда на нём 0, инвертирование делает активным уровень 1.

То есть если не инвертировать, то действует обычная логика - 0 это срабатывание/активация/активное состояние.

6.1.2.128 RPM_DIV_1000

```
#define RPM_DIV_1000 0x01
```

Флаг указывает на то что рабочая скорость указанная в команде задана в милли rpm.

Применим только для режима обратной связи ENCODER и только для BLDC-моторов.

6.1.2.129 SETPOS_IGNORE_ENCODER

```
#define SETPOS_IGNORE_ENCODER 0x02
```

Если установлен, то счётчик энкодера не обновляется.

6.1.2.130 SETPOS_IGNORE_POSITION

```
#define SETPOS_IGNORE_POSITION 0x01
```

Если установлен, то позиция в шагах и микрошагах не обновляется.

6.1.2.131 STATE_ALARM

```
#define STATE_ALARM 0x0000040
```

Контроллер находится в состоянии ALARM, показывая, что случилась какая-то опасная ситуация.

В состоянии ALARM все команды игнорируются пока не будет послана команда STOP и состояние ALARM деактивируется.

6.1.2.132 STATE_BORDERS_SWAP_MISSET

```
#define STATE_BORDERS_SWAP_MISSET 0x0008000
```

Достижение неверной границы.

6.1.2.133 STATE_BRAKE

```
#define STATE_BRAKE 0x0200
```

Состояние вывода управления тормозом.

Флаг "1" - если тормоз не запитан (зажат), "0" - если на тормоз подаётся питание (разжат).

6.1.2.134 STATE_BUTTON_LEFT

```
#define STATE_BUTTON_LEFT 0x0008
```

Состояние кнопки "влево" (1, если нажата).

6.1.2.135 STATE_BUTTON_RIGHT

```
#define STATE_BUTTON_RIGHT 0x0004
```

Состояние кнопки "вправо" (1, если нажата).

6.1.2.136 STATE_CONTR

```
#define STATE_CONTR 0x000003F
```

Флаги состояния контроллера.

6.1.2.137 STATE_CONTROLLER_OVERHEAT

```
#define STATE_CONTROLLER_OVERHEAT 0x0000200
```

Перегрелась микросхема контроллера.

6.1.2.138 STATE_CTP_ERROR

```
#define STATE_CTP_ERROR 0x0000080
```

Контроль позиции нарушен (используется только с шаговым двигателем).

Флаг устанавливается, когда положение энкодера и положение шага слишком далеки друг от друга.

6.1.2.139 STATE_DIG_SIGNAL

```
#define STATE_DIG_SIGNAL 0xFFFF
```

Флаги цифровых сигналов.

6.1.2.140 STATE_EEPROM_CONNECTED

```
#define STATE_EEPROM_CONNECTED 0x0000010
```

Подключена память EEPROM с настройками.

Встроенный профиль подвижки загружается из микросхемы памяти EEPROM, что позволяет подключать различные подвижки к контроллеру с автоматической настройкой.

6.1.2.141 STATE_ENC_A

```
#define STATE_ENC_A 0x2000
```

Состояние ножки A энкодера (флаг "1", если энкодер активен).

6.1.2.142 STATE_ENC_B

```
#define STATE_ENC_B 0x4000
```

Состояние ножки B энкодера (флаг "1", если энкодер активен).

6.1.2.143 STATE_ENGINE_RESPONSE_ERROR

```
#define STATE_ENGINE_RESPONSE_ERROR 0x0800000
```

Ошибка реакции двигателя на управляющее воздействие.

Отказ алгоритма управления двигателем означает, что он не может определять правильные решения с помощью полученных данных обратной связи. Единичный отказ может быть вызван механической проблемой. Повторяющийся сбой может быть вызван неправильной настройкой двигателя.

6.1.2.144 STATE_ERRC

```
#define STATE_ERRC 0x0000001
```

Недопустимая команда.

Полученная команда отсутствует в списке известных команд контроллера. Наиболее вероятной причиной является устаревшая прошивка.

6.1.2.145 STATE_ERRD

```
#define STATE_ERRD 0x0000002
```

Обнаружена ошибка целостности данных.

Данные внутри команды и ее CRC-код не соответствуют, поэтому данные не могут считаться действительными. Эта ошибка может быть вызвана электромагнитными помехами в интерфейсе UART/RS-232.

6.1.2.146 STATE_ERRV

```
#define STATE_ERRV 0x0000004
```

Недопустимое значение данных.

Обнаружена ошибка в значении. Значения в команде не могут быть применены без коррекции, поскольку они выходят за допустимый диапазон. Вместо исходных значений были использованы исправленные значения.

6.1.2.147 STATE_EXTIO_ALARM

```
#define STATE_EXTIO_ALARM 0x1000000
```

Ошибка вызвана внешним входным сигналом EXTIO.

6.1.2.148 STATE_GPIO_LEVEL

```
#define STATE_GPIO_LEVEL 0x0020
```

Состояние ввода/вывода общего назначения.

6.1.2.149 STATE_GPIO_PINOUT

```
#define STATE_GPIO_PINOUT 0x0010
```

Если флаг установлен, ввод/вывод общего назначения работает как выход; если флаг сброшен, ввод/вывод работает как вход.

6.1.2.150 STATE_IS_HOMED

```
#define STATE_IS_HOMED 0x0000020
```

Калибровка выполнена.

Это означает, что шкала относительного положения откалибрована с помощью аппаратного датчика абсолютного положения, такого как концевой переключатель.

6.1.2.151 STATE_LEFT_EDGE

```
#define STATE_LEFT_EDGE 0x0002
```

Достижение левой границы.

6.1.2.152 STATE_LOW_USB_VOLTAGE

```
#define STATE_LOW_USB_VOLTAGE 0x0002000
```

Устарело.

Слишком низкое напряжение на USB. Это поле оставлено для совместимости. Программное обеспечение не должно полагаться на значение этого поля.

6.1.2.153 STATE_OVERLOAD_POWER_CURRENT

```
#define STATE_OVERLOAD_POWER_CURRENT 0x0000800
```

Превышен максимальный ток потребления силовой части.

6.1.2.154 STATE_OVERLOAD_POWER_VOLTAGE

```
#define STATE_OVERLOAD_POWER_VOLTAGE 0x0000400
```

Превышено напряжение на силовой части.

6.1.2.155 STATE_OVERLOAD_USB_CURRENT

```
#define STATE_OVERLOAD_USB_CURRENT 0x0004000
```

Устарело.

Превышен максимальный ток потребления USB. Это поле оставлено для совместимости. Программное обеспечение не должно полагаться на значение этого поля.

6.1.2.156 STATE_OVERLOAD_USB_VOLTAGE

```
#define STATE_OVERLOAD_USB_VOLTAGE 0x0001000
```

Устарело.

Превышено напряжение на USB. Это поле оставлено для совместимости. Программное обеспечение не должно полагаться на значение этого поля.

6.1.2.157 STATE_POWER_OVERHEAT

```
#define STATE_POWER_OVERHEAT 0x0000100
```

Перегрев силового драйвера.

Управление двигателем отключено до восстановления рабочей температуры драйвера. Этого не должно происходить в коробочных версиях контроллера. Это может произойти в версии контроллера с «голой» платой и с пользовательским радиатором. Решение: используйте другой радиатор.

6.1.2.158 STATE_REV_SENSOR

```
#define STATE_REV_SENSOR 0x0400
```

Состояние вывода датчика оборотов (флаг "1", если датчик активен).

6.1.2.159 STATE_RIGHT_EDGE

```
#define STATE_RIGHT_EDGE 0x0001
```

Достижение правой границы.

6.1.2.160 STATE_SECUR

```
#define STATE_SECUR 0x1B3FFC0
```

Флаги безопасности.

6.1.2.161 STATE_SYNC_INPUT

```
#define STATE_SYNC_INPUT 0x0800
```

Состояние входа синхронизации (1, если вход синхронизации активен).

6.1.2.162 STATE_SYNC_OUTPUT

```
#define STATE_SYNC_OUTPUT 0x1000
```

Состояние выхода синхронизации (1, если выход синхронизации активен).

6.1.2.163 STATE_WINDING_RES_MISMATCH

```
#define STATE_WINDING_RES_MISMATCH 0x0100000
```

Сопротивления обмоток слишком сильно отличаются друг от друга.

Обычно это происходит с поврежденным шаговым двигателем, у которого полностью или частично закорочены обмотки.

6.1.2.164 SYNCIN_ENABLED

```
#define SYNCIN_ENABLED 0x01
```

Включение необходимости импульса синхронизации для начала движения.

6.1.2.165 SYNCIN_INVERT

```
#define SYNCIN_INVERT 0x02
```

Если установлен - срабатывает на спадающем фронте из 1 в 0.

Иначе - на нарастающем фронте из 0 в 1.

6.1.2.166 SYNCOUT_ENABLED

```
#define SYNCOUT_ENABLED 0x01
```

Синхронизация выхода работает согласно настройкам, если флаг установлен.

В ином случае значение выхода фиксировано и подчиняется SYNCOUT_STATE.

6.1.2.167 SYNCOUT_IN_STEPS

```
#define SYNCOUT_IN_STEPS 0x08
```

Если флаг установлен использовать шаги/импульсы энкодера для выходных импульсов синхронизации вместо миллисекунд.

6.1.2.168 SYNCOUT_INVERT

```
#define SYNCOUT_INVERT 0x04
```

Нулевой логический уровень является активным, если флаг установлен, а единичный - если флаг сброшен.

6.1.2.169 SYNCOUT_ONPERIOD

```
#define SYNCOUT_ONPERIOD 0x40
```

Выдает импульс синхронизации после прохождения SyncOutPeriod отсчётов.

6.1.2.170 SYNCOUT_ONSTART

```
#define SYNCOUT_ONSTART 0x10
```

Генерация синхронизирующего импульса при начале движения.

6.1.2.171 SYNCOUT_ONSTOP

```
#define SYNCOUT_ONSTOP 0x20
```

Генерация синхронизирующего импульса при остановке.

6.1.2.172 SYNCOUT_STATE

```
#define SYNCOUT_STATE 0x02
```

Когда значение выхода управляется напрямую (см.

флаг SYNCOUT_ENABLED), значение на выходе соответствует значению этого флага.

6.1.2.173 TS_TYPE_BITS

```
#define TS_TYPE_BITS 0x07
```

Биты, отвечающие за тип температурного датчика.

6.1.2.174 UART_PARITY_BITS

```
#define UART_PARITY_BITS 0x03
```

Биты, отвечающие за выбор четности.

6.1.2.175 WIND_A_STATE_ABSENT

```
#define WIND_A_STATE_ABSENT 0x00
```

Обмотка A не подключена.

6.1.2.176 WIND_A_STATE_MALFUNC

```
#define WIND_A_STATE_MALFUNC 0x02
```

Короткое замыкание на обмотке A.

6.1.2.177 WIND_A_STATE_OK

```
#define WIND_A_STATE_OK 0x03
```

Обмотка A работает адекватно.

6.1.2.178 WIND_A_STATE_UNKNOWN

```
#define WIND_A_STATE_UNKNOWN 0x01
```

Состояние обмотки A неизвестно.

6.1.2.179 WIND_B_STATE_ABSENT

```
#define WIND_B_STATE_ABSENT 0x00
```

Обмотка B не подключена.

6.1.2.180 WIND_B_STATE_MALFUNC

```
#define WIND_B_STATE_MALFUNC 0x20
```

Короткое замыкание на обмотке B.

6.1.2.181 WIND_B_STATE_OK

```
#define WIND_B_STATE_OK 0x30
```

Обмотка B работает адекватно.

6.1.2.182 WIND_B_STATE_UNKNOWN

```
#define WIND_B_STATE_UNKNOWN 0x10
```

Состояние обмотки B неизвестно.

6.1.2.183 XIMC_API

```
#define XIMC_API
```

Макрос импорта библиотеки.

Макросы позволяют автоматически импортировать функцию из общей библиотеки. Он автоматически расширяется до dllimport на msvc при включении файла заголовка.

6.1.3 Типы

6.1.3.1 calibration_t

```
typedef struct calibration_t calibration_t
```

Структура калибровок

Где найти все значения для расчета?

- XILab (не забудьте загрузить профиль для вашего позиционера. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg):
  - В настройках XILab перейдите во вкладку user units. Разделите второе число на первое — это и будет коэффициент A
  - В настройках XILab, перейдите во вкладку DC motor/BLDC motor/Stepper motor (зависит от используемого типа двигателя). Подставьте значение из поля Encoder counts per turn в формулу подсчета B коэффициента
- Профиль (откройте профиль любым текстовым редактором. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg):
  - Найдите в файле профиля поля Step_multiplier= и Unit_multiplier=. Разделите второе число на первое — это и будет коэффициент A
  - Найдите в файле профиля поле Encoder_CPT=. Подставьте значение из этого поля в формулу подсчета B коэффициента вместо ENCODER_COUNTS_PER_TURN

Как посчитать Speed, Accel, Decel и AntiplaySpeed в пользовательских единицах при использовании шагового двигателя с энкодером или DC/BLDC двигателей?

1. Используя XILab, загрузите профиль для вашего позиционера. Профиль должен соответствовать полному названию вашего позиционера. Например: 8MT173-25-MEn1.cfg
2. Включите режим Feedback encoder, если он не был включен ранее
3. Вводите скорость в пользовательских единицах в поле Working speed
4. Во вкладке User units выключите флаг User units. Это позволит видеть значение в поле Working speed в RPM
5. Умножьте значение из поля Working speed (в RPM) на коэффициент B. Например: $480 * 0.0000009375 = 0.00045$. Значение 0.00045 для позиционера 8MT173-25-MEn1 в режиме Encoder будет равно скорости 2 мм/сек

Ускорение, замедление и скорость в режиме антилюфта считаются аналогично

6.1.3.2 logging_callback_t

```
typedef void(XIMC_CALLCONV * logging_callback_t) (int loglevel, const wchar_t *message, void *user_data)
```

Прототип функции обратного вызова для логирования

Аргументы

loglevel	уровень логирования
message	сообщение

6.1.4 Функции

6.1.4.1 close_device()

```
result_t XIMC_API close_device (
              device_t * id )
```

Закрывает устройство

Аргументы

id	- идентификатор устройства

Заметки

Параметр id в данной функции является Си указателем, в отличие от большинства функций библиотеки использующих данный параметр

6.1.4.2 command_clear_fram()

```
result_t XIMC_API command_clear_fram (
              device_t id )
```

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

id	идентификатор устройства

6.1.4.3 command_eeread_settings()

```
result_t XIMC_API command_eeread_settings (
              device_t id )
```

Чтение настроек контроллера из EEPROM памяти позиционера.

Эта операция также автоматически выполняется при подключении позиционера с EEPROM памятью. Функция должна использоваться только производителем.

Аргументы

id	идентификатор устройства

6.1.4.4 command_eesave_settings()

```
result_t XIMC_API command_eesave_settings (
              device_t id )
```

Запись настроек контроллера в EEPROM память позиционера.

Функция должна использоваться только производителем.

Аргументы

id	идентификатор устройства

6.1.4.5 command_home()

```
result_t XIMC_API command_home (
              device_t id )
```

Движение в домашнюю позицию.

Алгоритм движения:

1) Двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME_DIR_FAST до достижения концевого выключателя, если флаг HOME_STOP_ENDS установлен. Или двигает до достижения сигнала с входа синхронизации, если установлен флаг HOME_STOP_SYNC. Или до поступления сигнала с датчика оборотов, если установлен флаг HOME_STOP_REV_SN

2) далее двигает согласно скоростям SlowHome, uSlowHome и флагу HOME_DIR_SLOW до достижения сигнала с входа синхронизации, если установлен флаг HOME_MV_SEC. Если флаг HOME↩_MV_SEC сброшен, пропускаем этот пункт.

3) далее двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME_DIR_SLOW на расстояние HomeDelta, uHomeDelta.

Описание флагов и переменных см. описание команд GHOM/SHOM

Аргументы

id	идентификатор устройства

См. также

home_settings_t
get_home_settings
set_home_settings

6.1.4.6 command_homezero()

```
result_t XIMC_API command_homezero (
              device_t id )
```

Запустить процедуру поиска домашней позиции, подождать её завершения и обнулить позицию в конце.

Это удобный путь для калибровки нулевой позиции.

Аргументы

	id	идентификатор устройства
out	*ret*	RESULT_OK, если контроллер завершил выполнение home и zero корректно или результат первого запроса к контроллеру со статусом отличным от RESULT_OK.

6.1.4.7 command_left()

```
result_t XIMC_API command_left (
              device_t id )
```

При получении команды "left" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.

Аргументы

id	идентификатор устройства

6.1.4.8 command_loft()

```
result_t XIMC_API command_loft (
              device_t id )
```

При получении команды "loft" двигатель смещается из текущей точки на расстояние Antiplay, заданное в настройках мотора (engine_settings), затем двигается в ту же точку.

Аргументы

id	идентификатор устройства

6.1.4.9 command_move()

```
result_t XIMC_API command_move (
              device_t id,
              int Position,
              int uPosition )
```

При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в полях Position, uPosition.

Для шагового мотора uPosition задает значение микрошага, для DC-мотора это поле не используется.

Аргументы

id	идентификатор устройства
Position	заданная позиция.
uPosition	часть позиции в микрошагах. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).

6.1.4.10 command_move_calb()

```
result_t XIMC_API command_move_calb (
              device_t id,
              float Position,
              const calibration_t * calibration )
```

Перемещение в позицию с использованием пользовательских единиц.

При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в поле Position.

Аргументы

id	идентификатор устройства
Position	позиция для перемещения
calibration	настройки пользовательских единиц

Заметки

Параметр Position корректируется таблицей коррекции.

6.1.4.11 command_movr()

```
result_t XIMC_API command_movr (
              device_t id,
              int DeltaPosition,
              int uDeltaPosition )
```

Перемещение на заданное смещение.

При получении команды "movr" двигатель начинает смещаться (если не используется режим "ТТЛ-СинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), влево или вправо (зависит от знака DeltaPosition) на количество импульсов указанное в полях DeltaPosition, uDeltaPosition. Для шагового мотора uDeltaPosition задает значение микрошага, для DC-мотора это поле не используется.

Аргументы

DeltaPosition	смещение.
uDeltaPosition	часть смещения в микрошагах. Величина микрошага и диапазон допустимых значений для данного поля зависят от выбранного режима деления шага (см. поле MicrostepMode в engine_settings).
id	идентификатор устройства

6.1.4.12 command_movr_calb()

```
result_t XIMC_API command_movr_calb (
              device_t id,
              float DeltaPosition,
              const calibration_t * calibration )
```

Перемещение на заданное смещение с использованием пользовательских единиц.

При получении команды "movr" двигатель начинает смещаться (если не используется режим "ТТЛ-СинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), влево или вправо (зависит от знака DeltaPosition) на расстояние указанное в поле DeltaPosition.

Аргументы

DeltaPosition	смещение.
id	идентификатор устройства
calibration	настройки пользовательских единиц

Заметки

Конечная координата вычисляемая с помощью DeltaPosition, корректируется таблицей коррекции. Однако корректировка не может быть применена в случае поступления команды movr во время движения. Команда movr устанавливает целевую позицию равной текущей целевой плюс дельта. Но точно определить текущую целевую координату во время движения библиотека не может. Поэтому она не может рассчитать конечную позицию и соответсвующую ей коррекцию.

6.1.4.13 command_power_off()

```
result_t XIMC_API command_power_off (
              device_t id )
```

Немедленное отключение питания двигателя вне зависимости от его состояния.

Команда предназначена для ручного управления питанием двигателя. Не следует использовать эту команду для отключения двигателя во время движения, так как питание может снова включиться для завершения движения. Для автоматического управления питанием двигателя и его отключения после остановки следует использовать систему управления электропитанием.

Аргументы

id	идентификатор устройства

См. также

get_power_settings
set_power_settings

6.1.4.14 command_read_robust_settings()

```
result_t XIMC_API command_read_robust_settings (
              device_t id )
```

Чтение важных настроек (калибровочные коэффициенты и т.

п.) контроллера из flash-памяти в оперативную, заменяя текущие настройки. Используется только производителем.

Аргументы

id	идентификатор устройства

6.1.4.15 command_read_settings()

```
result_t XIMC_API command_read_settings (
              device_t id )
```

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

Аргументы

id	идентификатор устройства

6.1.4.16 command_reset()

```
result_t XIMC_API command_reset (
              device_t id )
```

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

id	идентификатор устройства

6.1.4.17 command_right()

```
result_t XIMC_API command_right (
              device_t id )
```

При получении команды "rigt" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.

Аргументы

id	идентификатор устройства

6.1.4.18 command_save_robust_settings()

```
result_t XIMC_API command_save_robust_settings (
              device_t id )
```

При получении команды контроллер выполняет операцию сохранения важных настроек (калибровочные коэффициенты и т.

п.) во встроенную энергонезависимую память контроллера. Используется только производителем.

Аргументы

id	идентификатор устройства

6.1.4.19 command_save_settings()

```
result_t XIMC_API command_save_settings (
              device_t id )
```

При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.

Аргументы

id	идентификатор устройства

6.1.4.20 command_sstp()

```
result_t XIMC_API command_sstp (
              device_t id )
```

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

id	идентификатор устройства

6.1.4.21 command_start_measurements()

```
result_t XIMC_API command_start_measurements (
              device_t id )
```

Старт измерения и буферизации скорости, ошибки следования.

Аргументы

id	идентификатор устройства

6.1.4.22 command_stop()

```
result_t XIMC_API command_stop (
              device_t id )
```

Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим "удержания" дезактивируется для DC-двигателей, удержание тока в обмотках для шаговых двигателей (с учётом Power management настроек).

При вызове этой команды сбрасывается флаг ALARM.

Аргументы

id	идентификатор устройства

6.1.4.23 command_update_firmware()

```
result_t XIMC_API command_update_firmware (
              const char * uri,
              const uint8_t * data,
              uint32_t data_size )
```

Обновление прошивки.

Команда только для производителя.

Аргументы

uri	идентификатор устройства
data	указатель на массив байтов прошивки
data_size	размер массива в байтах

6.1.4.24 command_wait_for_stop()

```
result_t XIMC_API command_wait_for_stop (
              device_t id,
              uint32_t refresh_interval_ms )
```

Ожидание остановки контроллера

Аргументы

	id	идентификатор устройства
	refresh_interval_ms	Интервал обновления. Функция ждет столько миллисекунд между отправками контроллеру запроса get_status для проверки статуса остановки. Рекомендуемое значение интервала обновления - 10 мс. Используйте значения меньше 3 мс только если это необходимо - малые значения интервала обновления незначительно ускоряют обнаружение остановки, но создают существенно больший поток данных в канале связи контроллер-компьютер.
out	*ret*	RESULT_OK, если контроллер остановился, в противном случае первый результат выполнения команды get_status со статусом отличным от RESULT_OK.

6.1.4.25 command_zero()

```
result_t XIMC_API command_zero (
              device_t id )
```

Устанавливает текущую позицию равной 0.

Устанавливает позицию, в которую осуществляется движение по командам move и movr, равной нулю во всех случаях, кроме движения к позиции назначения. В последнем случае позиция назначения пересчитывается так, что в абсолютном положении точка назначения не меняется. То есть если мы находились в точке 400 и двигались к 500, то команда Zero делает текущую позицию 0, а позицию назначения - 100. Не изменяет режим движения: т.е. если движение осуществлялось, то оно продолжается; если мотор находился в режиме "удержания", то тип удержания сохраняется.

Аргументы

id	идентификатор устройства

6.1.4.26 enumerate_devices()

```
device_enumeration_t XIMC_API enumerate_devices (
              int enumerate_flags,
              const char * hints )
```

Поиск и составление списка доступных устройств.

По умолчанию формирует список устройств, подключенных к данному компьютеру и представленных в виде COM-портов. Дополнительно можно включить поиск сетевых устройств. Найденные в локальной сети устройства попадут в тот же список.
Для получения информации из собранного списка воспользуйтесь соответствующими функциями с префиксом `get_enumerate_` и полученным идентификатором `device_enumeration`.
После завершения работы со списком найденных устройств следует освободить память с помощью функции `free_enumerate_devices()`.

Аргументы

in	*enumerate_flags*	набор флагов, задающих режимы поиска. Флаги могут применяться совместно через побитовое "ИЛИ". • ENUMERATE_NETWORK - включает поиск сетевых устройств. Если флаг установлен, сетевые устройства будут добавлены в общий список. Если флаг не установлен, в списке будут только устройства, подключенные к данному компьютеру. • ENUMERATE_ALL_COM - при включенной опции опрашивает все устройства типа COM-порт в системе. При отключенной опции опрашивает только устройства, имена которых соответствуют маске устройств XIMC ("XIMC Motor Controller" в Windows, /dev/ximc/ и /dev/ttyACM/ на Linux/Mac). • ENUMERATE_PROBE - включает проверку устройств и сбор дополнительной информации (серийный номер, версию, модель, имя...). Если данный флаг установлен, в список будут добавлены только устройства, которые гарантированно можно открыть, но могут не попадать устройства, подключенные через RS232-преобразователи. Если флаг не установлен, то в списке будет больше устройств (в частности, попадут устройства, явным образом перечисленные в hints), но доступность и совместимость с данной библиотекой не гарантируется.
in	*hints*	дополнительная информация для повышения эффективности поиска. Имеет смысл использовать в случае сложной сетевой конфигурации, когда автоматический поиск может находить не всё. Формат - строка "ключ1=значение1\nключ2=значение2". Неизвестные ключи игнорируются. Один ключ может иметь несколько значений, которые перечисляются через запятую: ключ=значение1,значение2,значение3. Допустимые ключи: • addr - список URLов сетевых контроллеров или серверов с подключенными контроллерами. Поле применяется совместно с флагом ENUMERATE_NETWORK. Протоколы и форматы адресов: – xi-tcp://<ip-адрес> - сетевые контроллеры и контроллеры, подключенные через Ethernet-RS232 преобразователи. Если флаг ENUMERATE_PROBE не установлен, все перечисленные устройства попадут в список. – xi-net://<ip-адрес> - сетевые многоосевые системы, xi-net сервера. Запрос информации о наличии контроллеров по этим адресам будет сделан независимо от результатов автоматической процедуры сетевого поиска. • adapter_addr - список IP-адресов локальных сетевых адаптеров, через который должен осуществляться поиск. Если ключ отсутствует или ни одного адаптера не указано, то поиск производится на всех адаптерах.

Возвращает

device_enumeration - идентификатор списка устройств. Используется для получения информации об устройствах с помощью функций с префиксами get_enumerate_.

Примеры использования:

```
// Поиск локальных устройств без проверки
device_enumeration_t device_enumeration = enumerate_devices(0, "");
// Поиск локальных устройств с проверкой
device_enumeration_t device_enumeration = enumerate_devices(ENUMERATE_PROBE, "");
// Полностью автоматический поиск локальных и сетевых устройств
device_enumeration_t device_enumeration = enumerate_devices(ENUMERATE_NETWORK, "");
// Поиск локальных и сетевых устройств
// с использованием адаптера локального компьютера с адресом 192.168.0.100
// и явным обращениям к сетевому контроллеру с адресом 192.168.0.11
// и xi-net серверу с адресом  192.168.0.10
device_enumeration_t device_enumeration = enumerate_devices(
ENUMERATE_NETWORK,
"addr=192.168.0.10,xi-tcp://192.168.0.11\nadapter_addr=192.168.0.100"
);
```

6.1.4.27 free_enumerate_devices()

```
result_t XIMC_API free_enumerate_devices (
              device_enumeration_t device_enumeration )
```

Освобождает память, выделенную *enumerate_devices*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах

6.1.4.28 get_accessories_settings()

```
result_t XIMC_API get_accessories_settings (
              device_t id,
              accessories_settings_t * accessories_settings )
```

Устарело.

Чтение информации о дополнительных аксессуарах из EEPROM.

Аргументы

	id	идентификатор устройства
out	*accessories_settings*	структура, содержащая информацию о дополнительных аксессуарах

6.1.4.29 get_analog_data()

```
result_t XIMC_API get_analog_data (
                device_t id,
                analog_data_t * analog_data )
```

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

Эта функция используется для тестирования и калибровки устройства.

Аргументы

	id	идентификатор устройства
out	*analog_data*	аналоговые данные

6.1.4.30 get_bootloader_version()

```
result_t XIMC_API get_bootloader_version (
                device_t id,
                unsigned int * Major,
                unsigned int * Minor,
                unsigned int * Release )
```

Чтение номера версии загрузчика контроллера.

Аргументы

	id	идентификатор устройства
out	*Major*	номер основной версии
out	*Minor*	номер дополнительной версии
out	*Release*	номер релиза

6.1.4.31 get_brake_settings()

```
result_t XIMC_API get_brake_settings (
                device_t id,
                brake_settings_t * brake_settings )
```

Чтение настроек управления тормозом.

Аргументы

	id	идентификатор устройства
out	*brake_settings*	структура, содержащая настройки управления тормозом

6.1.4.32 get_calibration_settings()

```
result_t XIMC_API get_calibration_settings (
              device_t id,
              calibration_settings_t * calibration_settings )
```

Команда чтения калибровочных коэффициентов.

Команда используется только производителем. Эта функция заполняет структуру калибровочных коэффициентов. Эти коэффициенты используются для пересчёта кодов АЦП в токи обмоток и полный ток потребления. Коэффициенты сгруппированы в пары, XXX_A и XXX_B; пары представляют собой коэффициенты линейного уравнения. Первый коэффициент - тангенс угла наклона, второй - постоянное смещение. Таким образом, XXX_Current[mA] = XXX_A[mA/ADC]*XXX_ADC_CODE[ADC] + XXX_B[mA].

См. также

calibration_settings_t

Аргументы

	id	идентификатор устройства
out	*calibration_settings*	калибровочные коэффициенты

6.1.4.33 get_chart_data()

```
result_t XIMC_API get_chart_data (
              device_t id,
              chart_data_t * chart_data )
```

Команда чтения состояния обмоток и других нечасто используемых данных.

Предназначена в первую очередь для получения данных для построения графиков в паре с командой GETS.

См. также

chart_data_t

Аргументы

	id	идентификатор устройства
out	*chart_data*	структура chart_data.

6.1.4.34 get_control_settings()

```
result_t XIMC_API get_control_settings (
              device_t id,
              control_settings_t * control_settings )
```

Чтение настроек управления мотором.

При выборе CTL_MODE= 1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE= 2 включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed [0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+ 1]. При переходе от MaxSpeed [i] на MaxSpeed [i+ 1] действует ускорение, как обычно.

Аргументы

	id	идентификатор устройства
out	*control_settings*	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.

6.1.4.35 get_control_settings_calb()

```
result_t XIMC_API get_control_settings_calb (
              device_t id,
              control_settings_calb_t * control_settings_calb,
              const calibration_t * calibration )
```

Чтение настроек управления мотором с использованием пользовательских единиц.

При выборе CTL_MODE= 1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE= 2 включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed [0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+ 1]. При переходе от MaxSpeed [i] на MaxSpeed [i+ 1] действует ускорение, как обычно.

Аргументы

	id	идентификатор устройства
out	*control_settings_calb*	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.
	calibration	настройки пользовательских единиц

6.1.4.36 get_controller_name()

```
result_t XIMC_API get_controller_name (
              device_t id,
              controller_name_t * controller_name )
```

Чтение пользовательского имени контроллера и настроек из FRAM.

Аргументы

	id	идентификатор устройства
out	*controller_name*	структура, содержащая установленное пользовательское имя контроллера и флаги настроек

6.1.4.37 get_ctp_settings()

```
result_t XIMC_API get_ctp_settings (
              device_t id,
              ctp_settings_t * ctp_settings )
```

Чтение настроек контроля позиции(для шагового двигателя).

При управлении ШД с энкодером (CTP_BASE 0) появляется возможность обнаруживать потерю шагов. Контроллер знает количество шагов на оборот (GENG::StepsPerRev) и разрешение энкодера (GFBS::IPT). При включении контроля (флаг CTP_ENABLED), контроллер запоминает текущую позицию в шагах ШД и текущую позицию энкодера. Далее на каждом шаге позиция энкодера преобразовывается в шаги и, если разница оказывается больше CTPMinError, устанавливается флаг STATE_CTP_ERROR. При управлении ШД с датчиком оборотов (CTP_BASE 1), позиция контролируется по нему. По активному фронту на входе синхронизации контроллер запоминает текущее значение шагов. Далее, при каждом обороте проверяет, на сколько шагов сместились. При рассогласовании более CTPMinError устанавливается флаг STATE_CTP_ERROR.

Аргументы

	id	идентификатор устройства
out	*ctp_settings*	структура, содержащая настройки контроля позиции

6.1.4.38 get_debug_read()

```
result_t XIMC_API get_debug_read (
              device_t id,
              debug_read_t * debug_read )
```

Чтение данных из прошивки для отладки и поиска неисправностей.

Команда используется только производителем. Получаемые данные зависят от версии прошивки, истории и контекста использования.

Аргументы

	id	идентификатор устройства
out	*debug_read*	Данные для отладки.

6.1.4.39 get_device_count()

```
int XIMC_API get_device_count (
              device_enumeration_t device_enumeration )
```

Возвращает количество подключенных устройств.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах

6.1.4.40 get_device_information()

```
result_t XIMC_API get_device_information (
              device_t id,
              device_information_t * device_information )
```

Возвращает информацию об устройстве.

Все входные параметры должны быть указателями на выделенные области памяти длиной не менее 10 байт. Команда доступна как из инициализированного состояния, так и из исходного.

Аргументы

	id	идентификатор устройства.
out	*device_information*	информация об устройстве Информация об устройстве.

См. также

get_device_information

6.1.4.41 get_device_name()

```
pchar XIMC_API get_device_name (
              device_enumeration_t device_enumeration,
              int device_index )
```

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером *device_index*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах
in	*device_index*	номер устройства

6.1.4.42 get_edges_settings()

```
result_t XIMC_API get_edges_settings (
              device_t id,
              edges_settings_t * edges_settings )
```

Чтение настроек границ и концевых выключателей.

См. также

set_edges_settings

Аргументы

	id	идентификатор устройства
out	*edges_settings*	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей

6.1.4.43 get_edges_settings_calb()

```
result_t XIMC_API get_edges_settings_calb (
              device_t id,
              edges_settings_calb_t * edges_settings_calb,
              const calibration_t * calibration )
```

Чтение настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

set_edges_settings_calb

Аргументы

	id	идентификатор устройства
out	*edges_settings_calb*	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей
	calibration	настройки пользовательских единиц

Заметки

Внимание! Некоторые параметры структуры edges_settings_calb корректируются таблицей коррекции координат.

6.1.4.44 get_emf_settings()

```
result_t XIMC_API get_emf_settings (
              device_t id,
              emf_settings_t * emf_settings )
```

Чтение электромеханических настроек шагового двигателя.

Настройки различны для разных двигателей.

См. также

set_emf_settings

Аргументы

	id	идентификатор устройства
out	*emf_settings*	настройки EMF

6.1.4.45 get_encoder_information()

```
result_t XIMC_API get_encoder_information (
              device_t id,
              encoder_information_t * encoder_information )
```

Устарело.

Чтение информации об энкодере из EEPROM.

Аргументы

	id	идентификатор устройства
out	*encoder_information*	структура, содержащая информацию об энкодере

6.1.4.46 get_encoder_settings()

```
result_t XIMC_API get_encoder_settings (
              device_t id,
              encoder_settings_t * encoder_settings )
```

Устарело.

Чтение настроек энкодера из EEPROM.

Аргументы

	id	идентификатор устройства
out	*encoder_settings*	структура, содержащая настройки энкодера

6.1.4.47 get_engine_advanced_setup()

```
result_t XIMC_API get_engine_advanced_setup (
              device_t id,
              engine_advanced_setup_t * engine_advanced_setup )
```

Чтение расширенных настроек.

Используется только производителем.

См. также

set_engine_advanced_setup

Аргументы

	id	идентификатор устройства
out	*engine_advanced_setup*	настройки EAS

6.1.4.48 get_engine_settings()

```
result_t XIMC_API get_engine_settings (
              device_t id,
              engine_settings_t * engine_settings )
```

Чтение настроек 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

Аргументы

	id	идентификатор устройства
out	*engine_settings*	структура с настройками мотора

6.1.4.49 get_engine_settings_calb()

```
result_t XIMC_API get_engine_settings_calb (
              device_t id,
              engine_settings_calb_t * engine_settings_calb,
              const calibration_t * calibration )
```

Чтение настроек мотора с использованием пользовательских единиц.

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

Аргументы

	id	идентификатор устройства
out	*engine_settings_calb*	структура с настройками мотора
	calibration	настройки пользовательских единиц

6.1.4.50 get_entype_settings()

```
result_t XIMC_API get_entype_settings (
              device_t id,
              entype_settings_t * entype_settings )
```

Возвращает информацию о типе мотора и силового драйвера.

Аргументы

	id	идентификатор устройства
out	*entype_settings*	структура, содержащая настройки типа мотора и типа силового драйвера

6.1.4.51 get_enumerate_device_controller_name()

```
result_t XIMC_API get_enumerate_device_controller_name (
              device_enumeration_t device_enumeration,
              int device_index,
              controller_name_t * controller_name )
```

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером *device_index*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах
in	*device_index*	номер устройства
out	*controller*	name имя устройства

6.1.4.52 get_enumerate_device_information()

```
result_t XIMC_API get_enumerate_device_information (
              device_enumeration_t device_enumeration,
              int device_index,
              device_information_t * device_information )
```

Возвращает информацию о подключенном устройстве из перечисления устройств.

Возвращает информацию о устройстве с номером *device_index*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах
in	*device_index*	номер устройства
out	*device_information*	информация об устройстве

6.1.4.53 get_enumerate_device_network_information()

```
result_t XIMC_API get_enumerate_device_network_information (
                device_enumeration_t device_enumeration,
                int device_index,
                device_network_information_t * device_network_information )
```

Возвращает сетевую информацию о подключенном устройстве из перечисления устройств.

Возвращает сетевую информацию о устройстве с номером *device_index*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах
in	*device_index*	номер устройства
out	*device_network_information*	сетевая информация об устройстве

6.1.4.54 get_enumerate_device_serial()

```
result_t XIMC_API get_enumerate_device_serial (
                device_enumeration_t device_enumeration,
                int device_index,
                uint32_t * serial )
```

Возвращает серийный номер подключенного устройства из перечисления устройств.

Возвращает серийный номер устройства с номером *device_index*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах
in	*device_index*	номер устройства
in	*serial*	серийный номер устройства

6.1.4.55 get_enumerate_device_stage_name()

```
result_t XIMC_API get_enumerate_device_stage_name (
                device_enumeration_t device_enumeration,
                int device_index,
                stage_name_t * stage_name )
```

Возвращает имя подвижки для подключенного устройства из перечисления устройств.

Возвращает имя подвижки устройства с номером *device_index*.

Аргументы

in	*device_enumeration*	закрытый указатель на данные о перечисленных устойствах
in	*device_index*	номер устройства
out	*stage*	name имя подвижки

6.1.4.56 get_extended_settings()

```
result_t XIMC_API get_extended_settings (
                device_t id,
                extended_settings_t * extended_settings )
```

Чтение расширенных настроек.

В настоящее время не используется.

См. также

set_extended_settings

Аргументы

	id	идентификатор устройства
out	*extended_settings*	настройки EST

6.1.4.57 get_extio_settings()

```
result_t XIMC_API get_extio_settings (
                device_t id,
                extio_settings_t * extio_settings )
```

Команда чтения параметров настройки режимов внешнего ввода/вывода.

См. также

set_extio_settings

Аргументы

	id	идентификатор устройства
out	*extio_settings*	настройки EXTIO

6.1.4.58 get_feedback_settings()

```
result_t XIMC_API get_feedback_settings (
              device_t id,
              feedback_settings_t * feedback_settings )
```

Чтение настроек обратной связи

Аргументы

	id	идентификатор устройства
out	*IPS*	количество отсчётов энкодера на оборот вала. Диапазон: 1..65535. Поле устарело, рекомендуется записывать 0 в IPS и использовать расширенное поле CountsPerTurn. Может потребоваться обновление микропрограммы контроллера до последней версии.
out	*FeedbackType*	тип обратной связи
out	*FeedbackFlags*	флаги обратной связи
out	*CountsPerTurn*	количество отсчётов энкодера на оборот вала. Диапазон: 1..4294967295. Для использования поля CountsPerTurn нужно записать 0 в поле IPS, иначе будет использоваться значение из поля IPS.

6.1.4.59 get_firmware_version()

```
result_t XIMC_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

6.1.4.60 get_gear_information()

```
result_t XIMC_API get_gear_information (
              device_t id,
              gear_information_t * gear_information )
```

Устарело.

Чтение информации о редукторе из EEPROM.

Аргументы

	id	идентификатор устройства
out	*gear_information*	структура, содержащая информацию о редукторе

6.1.4.61 get_gear_settings()

```
result_t XIMC_API get_gear_settings (
              device_t id,
              gear_settings_t * gear_settings )
```

Устарело.

Чтение настроек редуктора из EEPROM.

Аргументы

	id	идентификатор устройства
out	*gear_settings*	структура, содержащая настройки редуктора

6.1.4.62 get_globally_unique_identifier()

```
result_t XIMC_API get_globally_unique_identifier (
              device_t id,
              globally_unique_identifier_t * globally_unique_identifier )
```

Считывает уникальный идентификатор каждого чипа, это значение не является случайным.

Используется только производителем. Уникальный идентификатор может быть использован в качестве инициализационного вектора для операций шифрования бутлоадера или в качестве серийного номера для USB и других применений.

Аргументы

	id	идентификатор устройства
out	*globally_unique_identifier*	результат полей 0-3 определяет уникальный 128-битный идентификатор.

6.1.4.63 get_hallsensor_information()

```
result_t XIMC_API get_hallsensor_information (
              device_t id,
              hallsensor_information_t * hallsensor_information )
```

Устарело.

Чтение информации о датчиках Холла из EEPROM.

Аргументы

	id	идентификатор устройства
out	*hallsensor_information*	структура, содержащая информацию о датчиках Холла

6.1.4.64 get_hallsensor_settings()

```
result_t XIMC_API get_hallsensor_settings (
              device_t id,
              hallsensor_settings_t * hallsensor_settings )
```

Устарело.

Чтение настроек датчиков Холла из EEPROM.

Аргументы

	id	идентификатор устройства
out	*hallsensor_settings*	структура, содержащая настройки датчиков Холла

6.1.4.65 get_home_settings()

```
result_t XIMC_API get_home_settings (
              device_t id,
              home_settings_t * home_settings )
```

Команда чтения настроек для подхода в home position.

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

См. также

home_settings_t

Аргументы

	id	идентификатор устройства
out	*home_settings*	настройки калибровки позиции

6.1.4.66 get_home_settings_calb()

```
result_t XIMC_API get_home_settings_calb (
              device_t id,
              home_settings_calb_t * home_settings_calb,
              const calibration_t * calibration )
```

Команда чтения настроек для подхода в home position с использованием пользовательских единиц.

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

См. также

home_settings_calb_t

Аргументы

	id	идентификатор устройства
out	*home_settings_calb*	настройки калибровки позиции
	calibration	настройки пользовательских единиц

6.1.4.67 get_init_random()

```
result_t XIMC_API get_init_random (
              device_t id,
              init_random_t * init_random )
```

Чтение случайного числа из контроллера.

Используется только производителем.

Аргументы

	id	идентификатор устройства
out	*init_random*	случайная последовательность, сгенерированная контроллером

6.1.4.68 get_joystick_settings()

```
result_t XIMC_API get_joystick_settings (
              device_t id,
              joystick_settings_t * joystick_settings )
```

Чтение настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость (при этом постоянно выполняется команда "soft stop"), а 100% отклонения соответствует MaxSpeed i, где i=0, если предыдущим использованием этого режима не было выбрано другое i. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. Первая скорость в списке не должна быть нулевой. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Подробнее см. раздел "Управление с помощью джойстика".

Аргументы

	id	идентификатор устройства
out	*joystick_settings*	структура, содержащая настройки джойстика

6.1.4.69 get_measurements()

```
result_t XIMC_API get_measurements (
              device_t id,
              measurements_t * measurements )
```

Команда чтения буфера данных для построения графиков скорости и ошибки следования.

Заполнение буфера начинается по команде "start_measurements". Буфер вмещает 25 точек, точки снимаются с периодом 1 мс. Для создания устойчивой системы следует считывать данные каждые 20 мс, если буфер полностью заполнен, то рекомендуется повторять считывания каждые 5 мс до момента пока буфер вновь не станет заполнен 20-ю точками.

См. также

measurements_t

Аргументы

	id	идентификатор устройства
out	*measurements*	структура с буфером и его длинной.

6.1.4.70 get_motor_information()

```
result_t XIMC_API get_motor_information (
              device_t id,
              motor_information_t * motor_information )
```

Устарело.

Чтение информации о двигателе из EEPROM.

Аргументы

	id	идентификатор устройства
out	*motor_information*	структура, содержащая информацию о двигателе

6.1.4.71 get_motor_settings()

```
result_t XIMC_API get_motor_settings (
              device_t id,
              motor_settings_t * motor_settings )
```

Устарело.

Чтение настроек двигателя из EEPROM.

Аргументы

	id	идентификатор устройства
out	*motor_settings*	структура, содержащая настройки двигателя

6.1.4.72 get_move_settings()

```
result_t XIMC_API get_move_settings (
              device_t id,
              move_settings_t * move_settings )
```

Команда чтения настроек перемещения (скорость, ускорение, порог и т.

д.).

Аргументы

	id	идентификатор устройства
out	*move_settings*	структура, содержащая настройки движения: скорость, ускорение, и т.д.

6.1.4.73 get_move_settings_calb()

```
result_t XIMC_API get_move_settings_calb (
              device_t id,
              move_settings_calb_t * move_settings_calb,
              const calibration_t * calibration )
```

Команда чтения настроек перемещения с использованием пользовательских единиц (скорость, ускорение, порог и т.

д.).

Аргументы

	id	идентификатор устройства
out	*move_settings_calb*	структура, содержащая настройки движения: скорость, ускорение, и т.д.
	calibration	настройки пользовательских единиц

6.1.4.74 get_network_settings()

```
result_t XIMC_API get_network_settings (
              device_t id,
              network_settings_t * network_settings )
```

Команда чтения сетевых настроек.

Используется только производителем. Эта функция возвращает текущие сетевые настройки.

Аргументы

DHCPEnabled	DHCP включен (1) или нет (0)
IPv4Address[4]	Массив[4] с IP-адресом
SubnetMask[4]	Массив[4] с маской подсети
DefaultGateway[4]	Массив[4] со шлюзом сети

6.1.4.75 get_nonvolatile_memory()

```
result_t XIMC_API get_nonvolatile_memory (
              device_t id,
              nonvolatile_memory_t * nonvolatile_memory )
```

Чтение пользовательских данных из FRAM.

Аргументы

	id	идентификатор устройства
out	*nonvolatile_memory*	структура, содержащая установленные пользовательские данные

6.1.4.76 get_password_settings()

```
result_t XIMC_API get_password_settings (
              device_t id,
              password_settings_t * password_settings )
```

Команда чтения пароля к веб-странице.

Используется только производителем. Эта функция позволяет прочитать пользовательский пароль к веб-странице из памяти контроллера.

Аргументы

UserPassword[20]	Строчка-пароль для доступа к веб-странице

6.1.4.77 get_pid_settings()

```
result_t XIMC_API get_pid_settings (
              device_t id,
              pid_settings_t * pid_settings )
```

Чтение ПИД-коэффициентов.

Эти коэффициенты определяют поведение позиционера. Коэффициенты различны для разных позиционеров.

См. также

set_pid_settings

Аргументы

	id	идентификатор устройства
out	*pid_settings*	настройки ПИД

6.1.4.78 get_position()

```
result_t XIMC_API get_position (
              device_t id,
              get_position_t * the_get_position )
```

Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.

Аргументы

	id	идентификатор устройства
out	*the_get_position*	структура, содержащая позицию мотора.

6.1.4.79 get_position_calb()

```
result_t XIMC_API get_position_calb (
              device_t id,
              get_position_calb_t * the_get_position_calb,
              const calibration_t * calibration )
```

Считывает значение положения в пользовательских единицах для шагового двигателя и в шагах энкодера всех двигателей.

Аргументы

	id	идентификатор устройства
out	*the_get_position_calb*	структура, содержащая позицию мотора.
	calibration	настройки пользовательских единиц

Заметки

Внимание! Некоторые параметры структуры get_position_calb корректируются таблицей коррекции координат.

6.1.4.80 get_power_settings()

```
result_t XIMC_API get_power_settings (
              device_t id,
              power_settings_t * power_settings )
```

Команда чтения параметров питания мотора.

Используется только с шаговым двигателем.

Аргументы

	id	идентификатор устройства
out	*power_settings*	структура, содержащая настройки питания шагового мотора

6.1.4.81 get_secure_settings()

```
result_t XIMC_API get_secure_settings (
              device_t id,
              secure_settings_t * secure_settings )
```

Команда записи установок защит.

Аргументы

	id	идентификатор устройства
out	*secure_settings*	настройки, определяющие максимально допустимые параметры, для защиты оборудования

См. также

status_t::flags

6.1.4.82 get_serial_number()

```
result_t XIMC_API get_serial_number (
              device_t id,
              unsigned int * SerialNumber )
```

Чтение серийного номера контроллера.

Аргументы

	id	идентификатор устройства
out	*SerialNumber*	серийный номер контроллера

6.1.4.83 get_stage_information()

```
result_t XIMC_API get_stage_information (
              device_t id,
              stage_information_t * stage_information )
```

Устарело.

Чтение информации о позиционере из EEPROM.

Аргументы

	id	идентификатор устройства
out	*stage_information*	структура, содержащая информацию о позиционере

6.1.4.84 get_stage_name()

```
result_t XIMC_API get_stage_name (
              device_t id,
              stage_name_t * stage_name )
```

Чтение пользовательского имени подвижки из EEPROM.

Аргументы

	id	идентификатор устройства
out	*stage_name*	структура, содержащая установленное пользовательское имя позиционера

6.1.4.85 get_stage_settings()

```
result_t XIMC_API get_stage_settings (
              device_t id,
              stage_settings_t * stage_settings )
```

Устарело.

Чтение настроек позиционера из EEPROM.

Аргументы

	id	идентификатор устройства
out	*stage_settings*	структура, содержащая настройки позиционера

6.1.4.86 get_status()

```
result_t XIMC_API get_status (
              device_t id,
              status_t * status )
```

Возвращает информацию о текущем состоянии устройства.

Аргументы

	id	идентификатор устройства
out	*status*	структура с информацией о текущем состоянии устройства Состояние устройства. Эта структура содержит основные параметры текущего состояния контроллера, такие как скорость, позиция и флаги состояния.

См. также

get_status

6.1.4.87 get_status_calb()

```
result_t XIMC_API get_status_calb (
              device_t id,
              status_calb_t * status,
              const calibration_t * calibration )
```

Возвращает информацию о текущем состоянии устройства.

Аргументы

	id	идентификатор устройства
out	*status*	структура с информацией о текущем состоянии устройства
	calibration	настройки пользовательских единиц Состояние устройства в калиброванных единицах. Эта структура содержит основные параметры текущего состояния контроллера, такие как скорость, позиция и флаги состояния, размерные величины выводятся в калиброванных единицах.

См. также

get_status

6.1.4.88 get_sync_in_settings()

```
result_t XIMC_API get_sync_in_settings (
              device_t id,
              sync_in_settings_t * sync_in_settings )
```

Чтение настроек для входного импульса синхронизации.

Эта функция считывает структуру с настройками синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

set_sync_in_settings

Аргументы

	id	идентификатор устройства
out	*sync_in_settings*	настройки синхронизации

6.1.4.89 get_sync_in_settings_calb()

```
result_t XIMC_API get_sync_in_settings_calb (
              device_t id,
              sync_in_settings_calb_t * sync_in_settings_calb,
              const calibration_t * calibration )
```

Чтение настроек для входного импульса синхронизации с использованием пользовательских единиц.

Эта функция считывает структуру с настройками синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

set_sync_in_settings_calb

Аргументы

	id	идентификатор устройства
out	*sync_in_settings_calb*	настройки синхронизации
	calibration	настройки пользовательских единиц

6.1.4.90 get_sync_out_settings()

```
result_t XIMC_API get_sync_out_settings (
              device_t id,
              sync_out_settings_t * sync_out_settings )
```

Чтение настроек для выходного импульса синхронизации.

Эта функция считывает структуру с настройками синхронизации, определяющими поведение выхода синхронизации, в память контроллера.

6.1.4.91 get_sync_out_settings_calb()

```
result_t XIMC_API get_sync_out_settings_calb (
              device_t id,
```

```
                sync_out_settings_calb_t * sync_out_settings_calb,
                const calibration_t * calibration )
```

Чтение настроек для выходного импульса синхронизации с использованием пользовательских единиц.

Эта функция считывает структуру с настройками синхронизации, определяющими поведение выхода синхронизации, в память контроллера.

См. также

set_sync_in_settings_calb

Аргументы

	id	идентификатор устройства
in	*sync_out_settings_calb*	настройки синхронизации
	calibration	настройки пользовательских единиц

6.1.4.92 get_uart_settings()

```
result_t XIMC_API get_uart_settings (
                device_t id,
                uart_settings_t * uart_settings )
```

Команда чтения настроек UART.

Эта функция заполняет структуру настроек UART.

См. также

uart_settings_t

Аргументы

	Speed	Скорость UART
out	*uart_settings*	настройки UART

6.1.4.93 goto_firmware()

```
result_t XIMC_API goto_firmware (
                device_t id,
                uint8_t * ret )
```

Перезагрузка в прошивку в контроллере

Аргументы

	id	идентификатор устройства
out	*ret*	RESULT_OK, если переход из загрузчика в прошивку возможен. После ответа на эту команду выполняется переход. RESULT_NO_FIRMWARE, если прошивка не найдена. RESULT_ALREADY_IN_FIRMWARE, если эта команда была вызвана из прошивки.

6.1.4.94 has_firmware()

```
result_t XIMC_API has_firmware (
              const char * uri,
              uint8_t * ret )
```

Проверка наличия прошивки в контроллере

Аргументы

	uri	уникальный идентификатор ресурса устройства
out	*ret*	ноль, если прошивка присутствует

6.1.4.95 logging_callback_stderr_narrow()

```
void XIMC_API logging_callback_stderr_narrow (
              int loglevel,
              const wchar_t * message,
              void * user_data )
```

Простая функция логирования на stderr в узких (однобайтных) символах

Аргументы

loglevel	уровень логирования
message	сообщение

6.1.4.96 logging_callback_stderr_wide()

```
void XIMC_API logging_callback_stderr_wide (
              int loglevel,
              const wchar_t * message,
              void * user_data )
```

Простая функция логирования на stderr в широких символах

Аргументы

loglevel	уровень логирования
message	сообщение

6.1.4.97 msec_sleep()

```
void XIMC_API msec_sleep (
              unsigned int msec )
```

Приостанавливает работу на указанное время

Аргументы

msec	время в миллисекундах

6.1.4.98 open_device()

```
device_t XIMC_API open_device (
              const char * uri )
```

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

Аргументы

in	*uri*	- уникальный идентификатор устройства. URI устройства имеет вид "xi-com:port" или "xi-net://host/serial" или "xi-emu:///abs_path_to_file". На POSIX системах допускается пропуск "рутовского" слэша; например, "xi-emu:///home/user/virt_controller.bin". Для USB-COM устройства "port" это URI устройства в ОС. Например, "xi-com:\\\\\.\\COM3" в Windows (с учётом экранирования двойные обратные слэши преобразуются в одинарные) или "xi-com:///dev/ttyACM0" в Linux/Mac. Для сетевого устройства "host" это IPv4 адрес или полностью определённое имя домена, "serial" это серийный номер устройства в шестнадцатеричной системе. Например, "xi-net://192.168.0.1/00001234" или "xi-net://hostname.com/89ABCDEF". Для работы по TCP протоколу используйте "xi-tcp://<ip/host>:<port>. Например, "xi-tcp://192.168.0.1:1818". Для виртуального устройства "abs_file_to_file" это путь к файлу с сохраненным состоянием устройства. Если файл не существует, он будет создан и инициализирован значениями по умолчанию. Например, "xi-emu:///C:/dir/file.bin" в Windows или "xi-emu:///home/user/file.bin"в Linux/Mac.

6.1.4.99 probe_device()

```
result_t XIMC_API probe_device (
              const char * uri )
```

Проверяет, является ли устройство с уникальным идентификатором *uri* XIMC-совместимым.

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

Аргументы

in	*uri*	- уникальный идентификатор устройства

6.1.4.100 service_command_updf()

```
result_t XIMC_API service_command_updf (
              device_t id )
```

Команда переводит контроллер в режим обновления прошивки.

Используется только производителем. Получив такую команду, прошивка платы устанавливает флаг (для загрузчика), отправляет эхо-ответ и перезагружает контроллер.

6.1.4.101 set_accessories_settings()

```
result_t XIMC_API set_accessories_settings (
              device_t id,
              const accessories_settings_t * accessories_settings )
```

Устарело.

Запись информации о дополнительных аксессуарах в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*accessories_settings*	структура, содержащая информацию о дополнительных аксессуарах

6.1.4.102 set_brake_settings()

```
result_t XIMC_API set_brake_settings (
              device_t id,
              const brake_settings_t * brake_settings )
```

Запись настроек управления тормозом.

Аргументы

	id	идентификатор устройства
in	*brake_settings*	структура, содержащая настройки управления тормозом

6.1.4.103 set_calibration_settings()

```
result_t XIMC_API set_calibration_settings (
              device_t id,
              const calibration_settings_t * calibration_settings )
```

Команда записи калибровочных коэффициентов.

Команда используется только производителем. Эта функция записывает структуру калибровочных коэффициентов в память контроллера. Эти коэффициенты используются для пересчёта кодов АЦП в токи обмоток и полный ток потребления. Коэффициенты сгруппированы в пары, XXX_A и XXX_B; пары представляют собой коэффициенты линейного уравнения. Первый коэффициент - тангенс угла наклона, второй - постоянное смещение. Таким образом, XXX_Current[mA] = XXX_A[mA/ADC]*↩ XXX_ADC_CODE[ADC] + XXX_B[mA].

См. также

calibration_settings_t

Аргументы

	id	идентификатор устройства
in	*calibration_settings*	калибровочные коэффициенты

6.1.4.104 set_control_settings()

```
result_t XIMC_API set_control_settings (
              device_t id,
              const control_settings_t * control_settings )
```

Запись настроек управления мотором.

При выборе CTL_MODE= 1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE= 2 включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed[0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+ 1]. При переходе от MaxSpeed [i] на MaxSpeed [i+ 1] действует ускорение, как обычно.

Аргументы

	id	идентификатор устройства
in	*control_settings*	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.

6.1.4.105 set_control_settings_calb()

```
result_t XIMC_API set_control_settings_calb (
              device_t id,
              const control_settings_calb_t * control_settings_calb,
              const calibration_t * calibration )
```

Запись настроек управления мотором с использованием пользовательских единиц.

При выборе CTL_MODE= 1 включается управление мотором с помощью джойстика. В этом режиме при отклонении джойстика на максимум двигатель стремится двигаться со скоростью MaxSpeed [i], где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Кнопки переключают номер скорости i. При выборе CTL_MODE= 2 включается управление мотором с помощью кнопок left/right. При нажатии на кнопки двигатель начинает двигаться в соответствующую сторону со скоростью MaxSpeed [0], по истечении времени Timeout[i] мотор двигается со скоростью MaxSpeed [i+ 1]. При переходе от MaxSpeed [i] на MaxSpeed [i+ 1] действует ускорение, как обычно.

Аргументы

	id	идентификатор устройства
in	*control_settings_calb*	структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо.
	calibration	настройки пользовательских единиц

6.1.4.106 set_controller_name()

```
result_t XIMC_API set_controller_name (
              device_t id,
              const controller_name_t * controller_name )
```

Запись пользовательского имени контроллера и настроек в FRAM.

Аргументы

	id	идентификатор устройства
in	*controller_information*	структура, содержащая информацию о контроллере

6.1.4.107 set_correction_table()

```
result_t XIMC_API set_correction_table (
              device_t id,
              const char * namefile )
```

Команда загрузки корректирующей таблицы из текстового файла.

Таблица используется для коррекции положения в случае механических неточностей. Работает для некоторых параметров в _calb командах.

Аргументы

	id	- идентификатор устройства
in	*namefile*	- путь до файла должен быть полным или относительным. Если параметр равен NULL, таблица коррекции будет очищена. Формат файла: два столбца, разделенных табуляцией. Заголовки столбцов строковые. Данные действительные, разделитель точка. Первый столбец - координата. Второй - отклонение, вызванное ошибкой механики. Максимальная длина таблицы 100 строк. Координаты должны быть отсортированы по возрастанию.

См. также

command_move
command_movr
get_position_calb
get_position_calb_t
get_status_calb
status_calb_t
get_edges_settings_calb
set_edges_settings_calb
edges_settings_calb_t

6.1.4.108 set_ctp_settings()

```
result_t XIMC_API set_ctp_settings (
              device_t id,
              const ctp_settings_t * ctp_settings )
```

Запись настроек контроля позиции(для шагового двигателя).

При управлении ШД с энкодером (CTP_BASE 0) появляется возможность обнаруживать потерю шагов. Контроллер знает количество шагов на оборот (GENG::StepsPerRev) и разрешение энкодера (GFBS::IPT). При включении контроля (флаг CTP_ENABLED), контроллер запоминает текущую позицию в шагах ШД и текущую позицию энкодера. Далее, на каждом шаге позиция энкодера преобразовывается в шаги и если разница оказывается больше CTPMinError, устанавливается флаг S↩TATE_CTP_ERROR.

При управлении ШД с датчиком оборотов (CTP_BASE 1), позиция контролируется по нему. По активному фронту на входе синхронизации контроллер запоминает текущее значение шагов. Далее, при каждом обороте проверяет, на сколько шагов сместились. При рассогласовании более CTPMin↩Error устанавливается флаг STATE_CTP_ERROR.

Аргументы

	id	идентификатор устройства
in	*ctp_settings*	структура, содержащая настройки контроля позиции

6.1.4.109 set_debug_write()

```
result_t XIMC_API set_debug_write (
              device_t id,
              const debug_write_t * debug_write )
```

Запись данных в прошивку для отладки и поиска неисправностей.

Команда используется только производителем.

Аргументы

	id	идентификатор устройства
in	*debug_write*	Данные для отладки.

6.1.4.110 set_edges_settings()

```
result_t XIMC_API set_edges_settings (
              device_t id,
              const edges_settings_t * edges_settings )
```

Запись настроек границ и концевых выключателей.

См. также

get_edges_settings

Аргументы

	id	идентификатор устройства
in	*edges_settings*	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей

6.1.4.111 set_edges_settings_calb()

```
result_t XIMC_API set_edges_settings_calb (
```

```
                device_t id,
                const edges_settings_calb_t * edges_settings_calb,
                const calibration_t * calibration )
```

Запись настроек границ и концевых выключателей с использованием пользовательских единиц.

См. также

get_edges_settings_calb

Аргументы

	id	идентификатор устройства
in	*edges_settings_calb*	настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей
	calibration	настройки пользовательских единиц

Заметки

Внимание! Некоторые параметры структуры edges_settings_calb корректируются таблицей коррекции координат.

6.1.4.112 set_emf_settings()

```
result_t XIMC_API set_emf_settings (
                device_t id,
                const emf_settings_t * emf_settings )
```

Запись электромеханических настроек шагового двигателя.

Настройки различны для разных двигателей. Пожалуйста, загружайте новые настройки, когда вы меняете мотор.

См. также

get_emf_settings

Аргументы

	id	идентификатор устройства
in	*emf_settings*	настройки EMF

6.1.4.113 set_encoder_information()

```
result_t XIMC_API set_encoder_information (
```

```
            device_t id,
            const encoder_information_t * encoder_information )
```

Устарело.

Запись информации об энкодере в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*encoder_information*	структура, содержащая информацию об энкодере

6.1.4.114 set_encoder_settings()

```
result_t XIMC_API set_encoder_settings (
            device_t id,
            const encoder_settings_t * encoder_settings )
```

Устарело.

Запись настроек энкодера в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*encoder_settings*	структура, содержащая настройки энкодера

6.1.4.115 set_engine_advanced_setup()

```
result_t XIMC_API set_engine_advanced_setup (
            device_t id,
            const engine_advanced_setup_t * engine_advanced_setup )
```

Запись расширенных настроек.

Используется только производителем.

См. также

get_engine_advanced_setup

Аргументы

	id	идентификатор устройства
in	*engine_advanced_setup*	настройки EAS

6.1.4.116 set_engine_settings()

```
result_t XIMC_API set_engine_settings (
              device_t id,
              const engine_settings_t * engine_settings )
```

Запись настроек 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

get_engine_settings

Аргументы

	id	идентификатор устройства
in	*engine_settings*	структура с настройками мотора

6.1.4.117 set_engine_settings_calb()

```
result_t XIMC_API set_engine_settings_calb (
              device_t id,
              const engine_settings_calb_t * engine_settings_calb,
              const calibration_t * calibration )
```

Запись настроек мотора с использованием пользовательских единиц.

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

get_engine_settings

Аргументы

	id	идентификатор устройства
in	*engine_settings_calb*	структура с настройками мотора
	calibration	настройки пользовательских единиц

6.1.4.118 set_entype_settings()

```
result_t XIMC_API set_entype_settings (
              device_t id,
              const entype_settings_t * entype_settings )
```

Запись информации о типе мотора и типе силового драйвера.

Аргументы

	id	идентификатор устройства
in	*entype_settings*	структура, содержащая настройки типа мотора и типа силового драйвера

6.1.4.119 set_extended_settings()

```
result_t XIMC_API set_extended_settings (
              device_t id,
              const extended_settings_t * extended_settings )
```

Запись расширенных настроек.

В настоящее время не используется.

См. также

get_extended_settings

Аргументы

	id	идентификатор устройства
in	*extended_settings*	настройки EST

6.1.4.120 set_extio_settings()

```
result_t XIMC_API set_extio_settings (
              device_t id,
              const extio_settings_t * extio_settings )
```

Команда записи параметров настройки режимов внешнего ввода/вывода.

Входные события обрабатываются по фронту. Выходные состояния сигнализируются логическим состоянием. По умолчанию нарастающий фронт считается моментом подачи входного сигнала, а единичное состояние считается активным выходом.

См. также

get_extio_settings

Аргументы

	id	идентификатор устройства
in	*extio_settings*	настройки EXTIO

6.1.4.121 set_feedback_settings()

```
result_t XIMC_API set_feedback_settings (
              device_t id,
              const feedback_settings_t * feedback_settings )
```

Запись настроек обратной связи.

Аргументы

	id	идентификатор устройства
in	*IPS*	количество отсчётов энкодера на оборот вала. Диапазон: 1..65535. Поле устарело, рекомендуется записывать 0 в IPS и использовать расширенное поле CountsPerTurn. Может потребоваться обновление микропрограммы контроллера до последней версии.
in	*FeedbackType*	тип обратной связи
in	*FeedbackFlags*	флаги обратной связи
in	*CountsPerTurn*	количество отсчётов энкодера на оборот вала. Диапазон: 1..4294967295. Для использования поля CountsPerTurn нужно записать 0 в поле IPS, иначе будет использоваться значение из поля IPS.

6.1.4.122 set_gear_information()

```
result_t XIMC_API set_gear_information (
              device_t id,
              const gear_information_t * gear_information )
```

Устарело.

Запись информации о редукторе в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*gear_information*	структура, содержащая информацию о редукторе

6.1.4.123 set_gear_settings()

```
result_t XIMC_API set_gear_settings (
              device_t id,
              const gear_settings_t * gear_settings )
```

Устарело.

Запись настроек редуктора в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*gear_settings*	структура, содержащая настройки редуктора

6.1.4.124 set_hallsensor_information()

```
result_t XIMC_API set_hallsensor_information (
              device_t id,
              const hallsensor_information_t * hallsensor_information )
```

Устарело.

Запись информации о датчиках Холла в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*hallsensor_information*	структура, содержащая информацию о датчиках Холла

6.1.4.125 set_hallsensor_settings()

```
result_t XIMC_API set_hallsensor_settings (
              device_t id,
              const hallsensor_settings_t * hallsensor_settings )
```

Устарело.

Запись настроек датчиков Холла в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*hallsensor_settings*	структура, содержащая настройки датчиков Холла

6.1.4.126 set_home_settings()

```
result_t XIMC_API set_home_settings (
              device_t id,
              const home_settings_t * home_settings )
```

Команда записи настроек для подхода в home position.

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

6.1.4.127 set_home_settings_calb()

```
result_t XIMC_API set_home_settings_calb (
              device_t id,
              const home_settings_calb_t * home_settings_calb,
              const calibration_t * calibration )
```

Команда записи настроек для подхода в home position с использованием пользовательских единиц.

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

См. также

home_settings_calb_t

Аргументы

	id	идентификатор устройства
in	*home_settings_calb*	настройки калибровки позиции
	calibration	настройки пользовательских единиц

6.1.4.128 set_joystick_settings()

```
result_t XIMC_API set_joystick_settings (
              device_t id,
              const joystick_settings_t * joystick_settings )
```

Запись настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость (при этом постоянно выполняется команда "soft stop"), а 100% отклонения соответствует MaxSpeed i, где i= 0, если предыдущим использованием этого режима не было выбрано другое i. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. Первая скорость в списке не должна быть нулевой. DeadZone вычисляется в десятых долях процента отклонения от центра (Joy↩ Center) до правого или левого максимума. Подробнее см. раздел "Управление с помощью джойстика" на сайте https://doc.xisupport.com/ru/8smc5-usb/8SMCn-USB/Technical_specification/↩ Additional_features/Joystick_control.html.

Аргументы

	id	идентификатор устройства
in	*joystick_settings*	структура, содержащая настройки джойстика

6.1.4.129 set_logging_callback()

```
void XIMC_API set_logging_callback (
              logging_callback_t logging_callback,
              void * user_data )
```

Устанавливает функцию обратного вызова для логирования.

Вызов назначает стандартный логгер (stderr, syslog), если передан NULL

Аргументы

logging_callback	указатель на функцию обратного вызова

6.1.4.130 set_motor_information()

```
result_t XIMC_API set_motor_information (
              device_t id,
              const motor_information_t * motor_information )
```

Устарело.

Запись информации о двигателе в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*motor_information*	структура, содержащая информацию о двигателе

6.1.4.131 set_motor_settings()

```
result_t XIMC_API set_motor_settings (
              device_t id,
              const motor_settings_t * motor_settings )
```

Устарело.

Запись настроек двигателя в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*motor_settings*	структура, содержащая настройки двигателя

6.1.4.132 set_move_settings()

```
result_t XIMC_API set_move_settings (
              device_t id,
              const move_settings_t * move_settings )
```

Команда записи настроек перемещения (скорость, ускорение, порог и т.

д.).

Аргументы

	id	идентификатор устройства
in	*move_settings*	структура, содержащая настройки движения: скорость, ускорение, и т.д.

6.1.4.133 set_move_settings_calb()

```
result_t XIMC_API set_move_settings_calb (
                device_t id,
                const move_settings_calb_t * move_settings_calb,
                const calibration_t * calibration )
```

Команда записи настроек перемещения, с использованием пользовательских единиц (скорость, ускорение, порог и т.

д.).

Аргументы

	id	идентификатор устройства
in	*move_settings_calb*	структура, содержащая настройки движения: скорость, ускорение, и т.д.
	calibration	настройки пользовательских единиц

6.1.4.134 set_network_settings()

```
result_t XIMC_API set_network_settings (
                device_t id,
                const network_settings_t * network_settings )
```

Команда записи сетевых настроек.

Используется только производителем. Эта функция меняет сетевые настройки на заданные.

Аргументы

DHCPEnabled	DHCP включен (1) или нет (0)
IPv4Address[4]	Массив[4] с IP-адресом
SubnetMask[4]	Массив[4] с маской подсети
DefaultGateway[4]	Массив[4] со шлюзом сети

6.1.4.135 set_nonvolatile_memory()

```
result_t XIMC_API set_nonvolatile_memory (
                device_t id,
                const nonvolatile_memory_t * nonvolatile_memory )
```

Запись пользовательских данных во FRAM.

Аргументы

	id	идентификатор устройства
in	*nonvolatile_memory*	структура, содержащая установленные пользовательские данные

6.1.4.136 set_password_settings()

```
result_t XIMC_API set_password_settings (
              device_t id,
              const password_settings_t * password_settings )
```

Команда записи пароля к веб-странице.

Используется только производителем. Эта функция меняет пользовательский пароль к веб-странице.

Аргументы

UserPassword[20]	Строчка-пароль для доступа к веб-странице

6.1.4.137 set_pid_settings()

```
result_t XIMC_API set_pid_settings (
              device_t id,
              const pid_settings_t * pid_settings )
```

Запись ПИД-коэффициентов.

Эти коэффициенты определяют поведение позиционера. Коэффициенты различны для разных позиционеров. Пожалуйста, загружайте новые настройки, когда вы меняете мотор или позиционер.

См. также

get_pid_settings

Аргументы

	id	идентификатор устройства
in	*pid_settings*	настройки ПИД

6.1.4.138 set_position()

```
result_t XIMC_API set_position (
              device_t id,
              const set_position_t * the_set_position )
```

Устанавливает произвольное значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера для всех двигателей.

Аргументы

	id	идентификатор устройства
out	*the_set_position*	структура, содержащая позицию мотора.

6.1.4.139 set_position_calb()

```
result_t XIMC_API set_position_calb (
              device_t id,
              const set_position_calb_t * the_set_position_calb,
              const calibration_t * calibration )
```

Устанавливает произвольное значение положения и значение энкодера всех двигателей с использованием пользовательских единиц.

Аргументы

	id	идентификатор устройства
out	*the_set_position_calb*	структура, содержащая позицию мотора.
	calibration	настройки пользовательских единиц

6.1.4.140 set_power_settings()

```
result_t XIMC_API set_power_settings (
              device_t id,
              const power_settings_t * power_settings )
```

Команда записи параметров питания мотора.

Используется только с шаговым двигателем.

Аргументы

	id	идентификатор устройства
in	*power_settings*	структура, содержащая настройки питания шагового мотора

6.1.4.141 set_secure_settings()

```
result_t XIMC_API set_secure_settings (
              device_t id,
              const secure_settings_t * secure_settings )
```

Команда записи установок защит.

Аргументы

id	идентификатор устройства
secure_settings	структура с настройками критических значений

См. также

status_t::flags

6.1.4.142 set_serial_number()

```
result_t XIMC_API set_serial_number (
              device_t id,
              const serial_number_t * serial_number )
```

Запись серийного номера и версии железа во flash память контроллера.

Вместе с новым серийным номером и версией железа передаётся "Ключ", только при совпадении которого происходит изменение и сохранение. Функция используется только производителем. Используется только загрузчиком.

Аргументы

	id	идентификатор устройства
in	*serial_number*	структура, содержащая серийный номер, версию железа и ключ.

6.1.4.143 set_stage_information()

```
result_t XIMC_API set_stage_information (
              device_t id,
              const stage_information_t * stage_information )
```

Устарело.

Запись информации о позиционере в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*stage_information*	структура, содержащая информацию о позиционере

6.1.4.144 set_stage_name()

```
result_t XIMC_API set_stage_name (
              device_t id,
              const stage_name_t * stage_name )
```

Запись пользовательского имени подвижки в EEPROM.

Аргументы

	id	идентификатор устройства
in	*stage_name*	структура, содержащая установленное пользовательское имя позиционера

6.1.4.145 set_stage_settings()

```
result_t XIMC_API set_stage_settings (
              device_t id,
              const stage_settings_t * stage_settings )
```

Устарело.

Запись настроек позиционера в EEPROM. Функция должна использоваться только производителем.

Аргументы

	id	идентификатор устройства
in	*stage_settings*	структура, содержащая настройки позиционера

6.1.4.146 set_sync_in_settings()

```
result_t XIMC_API set_sync_in_settings (
              device_t id,
              const sync_in_settings_t * sync_in_settings )
```

Запись настроек для входного импульса синхронизации.

Эта функция записывает структуру с настройками входного импульса синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

get_sync_in_settings

Аргументы

	id	идентификатор устройства
in	*sync_in_settings*	настройки синхронизации

6.1.4.147 set_sync_in_settings_calb()

```
result_t XIMC_API set_sync_in_settings_calb (
              device_t id,
              const sync_in_settings_calb_t * sync_in_settings_calb,
              const calibration_t * calibration )
```

Запись настроек для входного импульса синхронизации с использованием пользовательских единиц.

Эта функция записывает структуру с настройками входного импульса синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

get_sync_in_settings_calb

Аргументы

	id	идентификатор устройства
in	*sync_in_settings_calb*	настройки синхронизации
	calibration	настройки пользовательских единиц

6.1.4.148 set_sync_out_settings()

```
result_t XIMC_API set_sync_out_settings (
              device_t id,
              const sync_out_settings_t * sync_out_settings )
```

Запись настроек для выходного импульса синхронизации.

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

См. также

get_sync_in_settings

Аргументы

	id	идентификатор устройства
in	*sync_out_settings*	настройки синхронизации

6.1.4.149 set_sync_out_settings_calb()

```
result_t XIMC_API set_sync_out_settings_calb (
              device_t id,
              const sync_out_settings_calb_t * sync_out_settings_calb,
              const calibration_t * calibration )
```

Запись настроек для выходного импульса синхронизации с использованием пользовательских единиц.

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

См. также

get_sync_in_settings_calb

Аргументы

	id	идентификатор устройства
in	*sync_out_settings_calb*	настройки синхронизации
	calibration	настройки пользовательских единиц

6.1.4.150 set_uart_settings()

```
result_t XIMC_API set_uart_settings (
              device_t id,
              const uart_settings_t * uart_settings )
```

Команда записи настроек UART.

Эта функция записывает структуру настроек UART в память контроллера.

См. также

uart_settings_t

Аргументы

	Speed	Скорость UART
in	*uart_settings*	настройки UART

6.1.4.151 write_key()

```
result_t XIMC_API write_key (
              const char * uri,
              uint8_t * key )
```

Запись ключа защиты Функция используется только производителем.

Аргументы

	uri	идентификатор устройства
in	*key*	ключ защиты. Диапазон: 0..4294967295

6.1.4.152 ximc_version()

```
void XIMC_API ximc_version (
              char * version )
```

Возвращает версию библиотеки

Аргументы

version	буфер для строки с версией, 32 байт достаточно

Предметный указатель