libximc

2.10.5

Generated by Doxygen 1.8.2

Fri Jun 8 2018 13:14:15

Contents

Chapter 1

libximc library

Documentation for libximc library.

Libximc is cross-platform library for working with ximc 8SMC4 and 8SMC5 controllers.

Full documentation about ximc controllers is `there`

Full documentation about libximc API is available on the page ximc.h.

1.1 About ximc

We offer an inexpensive and ultra-compact servo-drive with USB interface for stepper motors with external power supply. Forget about cumbersome and expensive servo-drives! All you need is a stepper motor, a controller, a USB cable and any stabilized external power supply. That is all! Forget about active coolers as well. Controller's board is about the same size as a notepad or a cellphone, therefore, you may just put it down on the worktable without any assembly procedures. The controller works with any type of compact stepper motors with the rated winding current of up to 3A. Controller works with stepper motors with no feedback as well as with ones equipped with encoders in feedback loop, including linear encoders on the stages. The motor connector on the controller board is the same as one used by Standa company and it fits to all the Standa stages. USB connector provides easy communication and work with computer. Several controllers can be connected to one computer either via USB ports or through a special backplane supplied with multiaxis systems. The controller is fully compatible with the majority of operating systems, e.g., Windows, Mac OS X, Linux, etc.

1.2 About libximc

Congratulations on choosing XIMC multi-platform programming library! This document contains all information about XIMC library. It utilizes well known virtual COM-port interface, so you can use it on Windows 7, Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Linux, Mac OS X. XIMC multi-platform programing library supports plug/unplug on the fly. Each device can be controlled only by one program at once. Multiple processes (programs) that control one device simultaneously are not allowed.

Please read the Introduction to start work with library.

To use libximc in your project please consult with How to use with...

Chapter 2

Introduction

2.1 About

Congratulations on choosing XIMC multi-platform programming library! This document contains all information about XIMC library. It utilizes well known virtual COM-port interface, so you can use it on Windows 7, Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Linux, Mac OS X. XIMC multi-platform programing library supports plug/unplug on the fly. Each device can be controlled only by one program at once. Multiple processes (programs) that control one device simultaneously are not allowed.

2.2 System requirements

2.2.1 For rebuilding library

On Windows:

- Windows 2000 or later, 64-bit system (if compiling both arhitectures) or 32-bit system.
- Microsoft Visual C++ 2013 or later
- cygwin with tar, bison, flex, curl installed
- 7z

On Linux:

- 64-bit or/and 32-bit system system
- gcc 4 or later
- common autotools: autoconf, autoheader, aclocal, automake, autoreconf, libtool
- gmake
- doxygen - for building docs
- LaTeX distribution (teTeX or texlive) - for building docs
- flex 2.5.30+
- bison
- mercurial (for building developer version from hg)

On Mac OS X:

- XCode 4
- doxygen
- mactex
- autotools
- mercurial (for building developer version from hg)

If mercurial is used, please enable 'purge' extension by adding to ∼/.hgrc following lines:

```
[extensions]
  hgext.purge=
```

2.2.2 For using library

Supported operating systems (32 or 64 bit) and environment requirements:

- Mac OS X 10.6
- Windows 2000 or later
- Autotools-compatible unix. Package is installed from sources.
- Linux debian-based 32 and 64 bit. DEB package is built against Debian Squeeze 7
- Linux debian-based ARM. DEB package is built on Ubuntu 14.04
- Linux rpm-based. RPM is built against OpenSUSE 12
- Java 7 64-bit or 32-bit
- .NET 2.0 (32-bit only)
- Delphi (32-bit only)

Build requirements:

- Windows: Microsoft Visual C++ 2013 or mingw (currently not supported)
- UNIX: gcc 4, gmake
- Mac OS X: XCode 4
- JDK 7

Chapter 3

How to rebuild library

3.1 Building on generic UNIX

Generic version could be built with standard autotools.

```
./build.sh lib
```

Built files (library, headers, documentation) are installed to ./dist/local directory. It is a generic developer build. Sometimes you need to specify additional parameters to command line for your machine. Please look to following OS sections.

3.2 Building on debian-based linux systems

Requirement: 64-bit and 32-bit debian system, ubuntu Typical set of packages: gcc, autotools, autoconf, libtool, dpkg-dev, flex, bison, doxygen, texlive, mercurial Full set of packages: apt-get install ruby1.9.1 debhelper vim sudo g++ mercurial git curl make cmake autotools-dev automake autoconf libtool default-jre-headless default-jdk openjdk-6-jdk dpkg-dev lintian texlive texlive-latex-extra texlive-lang-cyrillic dh-autoreconf hardening-wrapper bison flex doxygen lsb-release pkg-config check For ARM cross-compiling install gcc-arm-linux-gnueabihf from your ARM toolchain.

It's required to match library and host architecture: 64-bit library can be built only at 64-bit host, 32-bit library - only at 32-bit host. ARM library is built with armhf cross-compiler gcc-arm-linux-gnueabihf.

To build library and package invoke a script:

```
$ ./build.sh libdeb
```

For ARM library replace 'libdeb' with 'libdebarm'.

Grab packages from ./ximc/deb and locally installed binaries from ./dist/local.

3.3 Building on redhat-based linux systems

Requirement: 64-bit redhat-based system (Fedora, Red Hat, SUSE) Typical set of packages: gcc, autotools, autoconf, libtool, flex, bison, doxygen, texlive, mercurial Full set of packages: autoconf automake bison doxygen flex gcc gcc-32bit gcc-c++ gcc-c++-32bit java-1_7_0-openjdk java-1_7_0-openjdk-devel libtool lsb-release make mercurial rpm-build rpm-devel rpmlint texlive texlive-fonts-extra texlive-latex

It's possible to build both 32- and 64-bit libraries on 64-bit host system. 64-bit library can't be built on 32-bit system.

To build library and package invoke a script:

```
$ ./build.sh librpm
```

Grab packages from ./ximc/rpm and locally installed binaries from ./dist/local.

3.4 Buliding on Mac OS X

To build and package a script invoke a script:

```
$ ./build.sh libosx
```

Built library (classical and framework), examples (classical and .app), documentation are located at ./ximc/macosx, locally installed binaries from ./dist/local.

3.5 Buliding on Windows

Requirements: 64-bit windows (build script builds both architectures), cygwin (must be installed to a default path), mercurial.

Invoke a script:

```
$ ./build.bat
```

Grab packages from ./deb/win32 and ./deb/win64

To build debug version of the library set environment variable ”DEBUG” to ”true” before running the build script.

3.6 Source code access

XIMC source codes are given under special request.

Chapter 4

How to use with...

Library usage can be examinated from test application testapp. Non-C languages are supported because library supports stdcall calling convention and so can be used with a variety of languages.

C test project is located at 'examples/testapp' directory, C# test project - at 'examples/testcs', VB.NET - 'examples/testvbnet', Delphi 6 - 'examples/testdelphi', sample bindings for MATLAB - 'examples/testmatlab', for Java - 'examples/testjava', for Python - 'examples/testpython'. Development kit also contains precompiled examples: testapp and testappeasy as 32 and 64-bit applications for Windows and 64-bit application for osx, testcs, testvbnet, testdelphi - 32-bit only, testjava is architecture-independent, testmatlab and testpython are runtime-interpreted.

NOTE: SDK requires Microsoft Visual C++ Redistributable Package (provided with SDK - vcredist_x86 or vcredist_x64)

4.1 Usage with C

4.1.1 Visual C++

Testapp can be built using testapp.sln. Library must be compiled with MS Visual C++ too, mingw-library isn't supported. Make sure that Microsoft Visual C++ Redistributable Package is installed.

Open solution examples/testapp/testapp.sln, build and run from the IDE.

4.1.2 CodeBlocks

Testapp can be built using testcodeblocks.cbp. Library must be compiled with MS Visual C++ too, mingw-library isn't supported. Make sure that Microsoft Visual C++ Redistributable Package is installed. $*$

Open solution examples/testcodeblocks/testcodeblocks.cbp, build and run from the IDE.

4.1.3 MinGW

MinGW is a port of GCC to win32 platform. It's required to install MinGW package. Currently not supported

MinGW-compiled testapp can be built with MS Visual C++ or mingw library.

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

Then copy library libximc.dll to current directory and launch testapp.exe.

4.1.4 C++ Builder

First of all you should create C++ Builder-style import library. Visual C++ library is not compatible with BCB. Invoke:

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

Then compile test application:

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

4.1.5 XCode

Test app should be built with XCode project testapp.xcodeproj. Library is a Mac OS X framework, and at example application it's bundled inside testapp.app

Then launch application testapp.app and check activity output in Console.app.

4.1.6 GCC

Make sure that libximc (rpm, deb, freebsd package or tarball) is installed at your system. Installation of package should be performed with a package manager of operating system. On OS X a framework is provided.

Note that user should belong to system group which allows access to a serial port (dip or serial, for example).

Copy file /usr/share/libximc/keyfile.sqlite project directory:

```
$ cp /usr/share/libximc/keyfile.sqlite .
```

Test application can be built with the installed library with the following script:

```
$ make
```

In case of cross-compilation (target architecture differs from the current system architecture) feed -m64 or -m32 flag to compiler. On OS X it's needed to use -arch flag instead to build an universal binary. Please consult a compiler documentation.

Then launch the application as:

```
$ make run
```

Note: make run on OS X copies a library to the current directory. If you want to use library from the custom directory please be sure to specify LD_LIBRARY_PATH or DYLD_LIBRARY_PATH to the directory with the library.

4.2 .NET

Wrapper assembly for libximc.dll is wrappers/csharp/ximcnet.dll. It is provided with two different architectures and depends on .NET 2.0.

Test .NET applications for Visual Studio 2013 is located at testcs (for C#) and testvbnet (for VB.NET) respectively. Open solutions and build.

4.3 Delphi

Wrapper for libximc.dll is a unit wrappers/delphi/ximc.pas

Console test application for is located at testdelphi. Tested with Delphi 6 and only 32-bit version.

Just compile, place DLL near the executable and run program.

4.4 Java

How to run example on Linux. Navigate to ximc-2.x.x./examples/testjava/compiled/ and run:

```
$ cp /usr/share/libximc/keyfile.sqlite .
$ java -cp /usr/share/java/libjximc.jar:testjava.jar ru.ximc.TestJava
```

How to run example on Windows or Mac. Navigate to ximc-2.x.x./examples/testjava/compiled/. Copy contents of ximc-2.x.x/ximc/win64 or ximc-2.x.x/ximc/macosx accordingly to the current directory. Then run:

```
$ java -classpath libjximc.jar -classpath testjava.jar ru.ximc.TestJava
```

How to modify and recompile an example. Navigate to examples/testjava/compiled. Sources are embedded in a testjava.jar. Extract them:

```
$ jar xvf testjava.jar ru META-INF
```

Then rebuild sources:

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

or for windows or mac

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

Then build a jar:

```
$ jar cmf META-INF/MANIFEST.MF testjava.jar ru
```

4.5 Python

Change current directory to the examples/testpython.

Before launch:

On OS X: copy library ximc/macosx/libximc.framework to the current directory.

On Linux: you may need to set LD_LIBRARY_PATH so Python can locate libraries with RPATH. For example, you may need:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:`pwd`
```

On Windows before the start nothing needs to be done

Launch Python 2 or Python 3:

```
python testpython.py
```

4.6 MATLAB

Sample MATLAB program testximc.m is provided at the directory examples/testmatlab. On windows copy ximc.h, libximc.dll, bindy.dll, xiwrapper.dll and contents of ximc/(win32,win64)/wrappers/matlab/ directory to the current directory.

Before launch:

On OS X: copy ximc/macosx/libximc.framework, ximc/macosx/wrappers/ximcm.h, ximc/ximc.h $*$ to the directory examples/matlab. Install XCode compatible with Matlab.

On Linux: install libximc∗deb and libximc-dev∗dev of target architecture. Then copy ximc/macosx/wrappers/ximcm.h to the directory examples/matlab. Install gcc compatible with Matlab.

For XCode and gcc version compability check document `https://www.mathworks.com/content/dam/mathworks/mat` `SystemRequirements-Release2014a_SupportedCompilers.pdf` or similar.

On Windows before the start nothing needs to be done

Change current directory in the MATLAB to the examples/matlab. Then launch in MATLAB prompt:

```
testximc
```

4.7 Generic logging facility

If you want to turn on file logging, you should run the program that uses libximc library with the ”XILOG” environment variable set to desired file name. This file will be opened for writing on the first log event and will be closed when the program which uses libximc terminates. Data which is sent to/received from the controller is logged along with port open and close events.

4.8 Required permissions

libximc generally does not require special permissions to work, it only needs read/write access to USB-serial ports on the system. An exception to this rule is a Windows-only ”fix_usbser_sys()” function - it needs elevation and will produce null result if run as a regular user.

4.9 C-profiles

C-profiles are header files distributed with the libximc library. They enable one to set all controller settings for any of the supported stages with a single function call in a C/C++ program. You may see how to use C-profiles in ”testcprofile” example directory.

Chapter 5

Data Structure Documentation

5.1 accessories_settings_t Struct Reference

Additional accessories information.

Data Fields

- char MagneticBrakeInfo [25]

  The manufacturer and the part number of magnetic brake, the maximum string length is 24 characters.
- float MBRatedVoltage

  Rated voltage for controlling the magnetic brake (B).
- float MBRatedCurrent

  Rated current for controlling the magnetic brake (A).
- float MBTorque

  Retention moment (mN m).
- unsigned int MBSettings

  Magnetic brake settings flags.
- char TemperatureSensorInfo [25]

  The manufacturer and the part number of the temperature sensor, the maximum string length: 24 characters.
- float TSMin

  The minimum measured temperature (degrees Celsius) Data type: float.
- float TSMax

  The maximum measured temperature (degrees Celsius) Data type: float.
- float TSGrad

  The temperature gradient (V/degrees Celsius).
- unsigned int TSSettings

  Temperature sensor settings flags.
- unsigned int LimitSwitchesSettings

  Temperature sensor settings flags.

5.1.1 Detailed Description

Additional accessories information.

See Also

set_accessories_settings
get_accessories_settings
get_accessories_settings, set_accessories_settings

5.1.2 Field Documentation

5.1.2.1 unsigned int LimitSwitchesSettings

Temperature sensor settings flags.

5.1.2.2 char MagneticBrakeInfo[25]

The manufacturer and the part number of magnetic brake, the maximum string length is 24 characters.

5.1.2.3 float MBRatedCurrent

Rated current for controlling the magnetic brake (A).

Data type: float.

5.1.2.4 float MBRatedVoltage

Rated voltage for controlling the magnetic brake (B).

Data type: float.

5.1.2.5 unsigned int MBSettings

Magnetic brake settings flags.

5.1.2.6 float MBTorque

Retention moment (mN m).

Data type: float.

5.1.2.7 char TemperatureSensorInfo[25]

The manufacturer and the part number of the temperature sensor, the maximum string length: 24 characters.

5.1.2.8 float TSGrad

The temperature gradient (V/degrees Celsius).

Data type: float.

5.1.2.9 float TSMax

The maximum measured temperature (degrees Celsius) Data type: float.

5.1.2.10 float TSMin

The minimum measured temperature (degrees Celsius) Data type: float.

5.1.2.11 unsigned int TSSettings

Temperature sensor settings flags.

5.2 analog_data_t Struct Reference

Analog data.

Data Fields

- unsigned int A1Voltage_ADC

  "Voltage on pin 1 winding A" raw data from ADC.
- unsigned int A2Voltage_ADC

  "Voltage on pin 2 winding A" raw data from ADC.
- unsigned int B1Voltage_ADC

  "Voltage on pin 1 winding B" raw data from ADC.
- unsigned int B2Voltage_ADC

  "Voltage on pin 2 winding B" raw data from ADC.
- unsigned int SupVoltage_ADC

  "Voltage on the top of MOSFET full bridge" raw data from ADC.
- unsigned int ACurrent_ADC

  "Winding A current" raw data from ADC.
- unsigned int BCurrent_ADC

  "Winding B current" raw data from ADC.
- unsigned int FullCurrent_ADC

  "Full current" raw data from ADC.
- unsigned int Temp_ADC

  Voltage from temperature sensor, raw data from ADC.
- unsigned int Joy_ADC

  Joystick raw data from ADC.
- unsigned int Pot_ADC

  Voltage on analog input, raw data from ADC.
- unsigned int L5_ADC

  USB supply voltage after the current sense resistor, from ADC.
- unsigned int H5_ADC

  Power supply USB from ADC.
- int A1Voltage

  "Voltage on pin 1 winding A" calibrated data.
- int A2Voltage

  "Voltage on pin 2 winding A" calibrated data.
- int B1Voltage

  "Voltage on pin 1 winding B" calibrated data.
- int B2Voltage

  "Voltage on pin 2 winding B" calibrated data.
- int SupVoltage

  "Voltage on the top of MOSFET full bridge" calibrated data.
- int ACurrent

  "Winding A current" calibrated data.
- int BCurrent

  "Winding B current" calibrated data.
- int FullCurrent

  "Full current" calibrated data.
- int Temp

  Temperature, calibrated data.

- int Joy

  Joystick, calibrated data.
- int Pot

  Analog input, calibrated data.
- int L5

  USB supply voltage after the current sense resistor.
- int H5

  Power supply USB.
- unsigned int **deprecated**
- int R

  Motor winding resistance in mOhms(is only used with stepper motor).
- int L

  Motor winding pseudo inductance in uHn(is only used with stepper motor).

5.2.1 Detailed Description

Analog data.

This structure contains raw analog data from ADC embedded on board. These data used for device testing and deep recalibraton by manufacturer only.

See Also

get_analog_data
get_analog_data

5.2.2 Field Documentation

5.2.2.1 int A1Voltage

”Voltage on pin 1 winding A” calibrated data.

5.2.2.2 unsigned int A1Voltage_ADC

”Voltage on pin 1 winding A” raw data from ADC.

5.2.2.3 int A2Voltage

”Voltage on pin 2 winding A” calibrated data.

5.2.2.4 unsigned int A2Voltage_ADC

”Voltage on pin 2 winding A” raw data from ADC.

5.2.2.5 int ACurrent

”Winding A current” calibrated data.

5.2.2.6 unsigned int ACurrent_ADC

”Winding A current” raw data from ADC.

5.2.2.7 int B1Voltage

”Voltage on pin 1 winding B” calibrated data.

5.2.2.8 unsigned int B1Voltage_ADC

”Voltage on pin 1 winding B” raw data from ADC.

5.2.2.9 int B2Voltage

”Voltage on pin 2 winding B” calibrated data.

5.2.2.10 unsigned int B2Voltage_ADC

”Voltage on pin 2 winding B” raw data from ADC.

5.2.2.11 int BCurrent

”Winding B current” calibrated data.

5.2.2.12 unsigned int BCurrent_ADC

”Winding B current” raw data from ADC.

5.2.2.13 int FullCurrent

”Full current” calibrated data.

5.2.2.14 unsigned int FullCurrent_ADC

”Full current” raw data from ADC.

5.2.2.15 int Joy

Joystick, calibrated data.

Range: 0..10000

5.2.2.16 unsigned int Joy_ADC

Joystick raw data from ADC.

5.2.2.17 int L

Motor winding pseudo inductance in uHn(is only used with stepper motor).

5.2.2.18 int L5

USB supply voltage after the current sense resistor.

5.2.2.19 unsigned int L5_ADC

USB supply voltage after the current sense resistor, from ADC.

5.2.2.20 int Pot

Analog input, calibrated data.

Range: 0..10000

5.2.2.21 int R

Motor winding resistance in mOhms(is only used with stepper motor).

5.2.2.22 int SupVoltage

"Voltage on the top of MOSFET full bridge" calibrated data.

5.2.2.23 unsigned int SupVoltage_ADC

"Voltage on the top of MOSFET full bridge" raw data from ADC.

5.2.2.24 int Temp

Temperature, calibrated data.

5.2.2.25 unsigned int Temp_ADC

Voltage from temperature sensor, raw data from ADC.

5.3 brake_settings_t Struct Reference

Brake settings.

Data Fields

- unsigned int t1

  Time in ms between turn on motor power and turn off brake.
- unsigned int t2

  Time in ms between turn off brake and moving readiness.
- unsigned int t3

  Time in ms between motor stop and turn on brake.
- unsigned int t4

  Time in ms between turn on brake and turn off motor power.
- unsigned int BrakeFlags

  Brake settings flags.

5.3.1 Detailed Description

Brake settings.

This structure contains parameters of brake control.

See Also

set_brake_settings
get_brake_settings
get_brake_settings, set_brake_settings

5.3.2 Field Documentation

5.3.2.1 unsigned int BrakeFlags

Brake settings flags.

5.3.2.2 unsigned int t1

Time in ms between turn on motor power and turn off brake.

5.3.2.3 unsigned int t2

Time in ms between turn off brake and moving readiness.

All moving commands will execute after this interval.

5.3.2.4 unsigned int t3

Time in ms between motor stop and turn on brake.

5.3.2.5 unsigned int t4

Time in ms between turn on brake and turn off motor power.

5.4 calibration_settings_t Struct Reference

Calibration settings.

Data Fields

- float CSS1_A

  Scaling factor for the analogue measurements of the winding A current.
- float CSS1_B

  Shift factor for the analogue measurements of the winding A current.
- float CSS2_A

  Scaling factor for the analogue measurements of the winding B current.
- float CSS2_B

  Shift factor for the analogue measurements of the winding B current.
- float FullCurrent_A

  Scaling factor for the analogue measurements of the full current.

- float FullCurrent_B

    Shift factor for the analogue measurements of the full current.

5.4.1 Detailed Description

Calibration settings.

This structure contains calibration settings.

See Also

get_calibration_settings
set_calibration_settings
get_calibration_settings, set_calibration_settings

5.4.2 Field Documentation

5.4.2.1 float CSS1_A

Scaling factor for the analogue measurements of the winding A current.

5.4.2.2 float CSS1_B

Shift factor for the analogue measurements of the winding A current.

5.4.2.3 float CSS2_A

Scaling factor for the analogue measurements of the winding B current.

5.4.2.4 float CSS2_B

Shift factor for the analogue measurements of the winding B current.

5.4.2.5 float FullCurrent_A

Scaling factor for the analogue measurements of the full current.

5.4.2.6 float FullCurrent_B

Shift factor for the analogue measurements of the full current.

5.5 calibration_t Struct Reference

Calibration companion structure.

Data Fields

- double A

    Mulitiplier.
- unsigned int MicrostepMode

    Microstep mode.

5.5.1 Detailed Description

Calibration companion structure.

5.6 chart_data_t Struct Reference

Additional device state.

Data Fields

- int WindingVoltageA

  In the case step motor, the voltage across the winding A; in the case of a brushless, the voltage on the first coil, in the case of the only DC.
- int WindingVoltageB

  In the case step motor, the voltage across the winding B; in case of a brushless, the voltage on the second winding, and in the case of DC is not used.
- int WindingVoltageC

  In the case of a brushless, the voltage on the third winding, in the case step motor and DC is not used.
- int WindingCurrentA

  In the case step motor, the current in the coil A; brushless if the current in the first coil, and in the case of a single DC.
- int WindingCurrentB

  In the case step motor, the current in the coil B; brushless if the current in the second coil, and in the case of DC is not used.
- int WindingCurrentC

  In the case of a brushless, the current in the third winding, in the case step motor and DC is not used.
- unsigned int Pot

  Analog input value in ten-thousandths.
- unsigned int Joy

  The joystick position in the ten-thousandths.
- int DutyCycle

  Duty cycle of PWM.

5.6.1 Detailed Description

Additional device state.

This structure contains additional values such as winding's voltages, currents and temperature.

See Also

get_chart_data
get_chart_data

5.6.2 Field Documentation

5.6.2.1 int DutyCycle

Duty cycle of PWM.

5.6.2.2 unsigned int Joy

The joystick position in the ten-thousandths.

Range: 0..10000

5.6.2.3 unsigned int Pot

Analog input value in ten-thousandths.

Range: 0..10000

5.6.2.4 int WindingCurrentA

In the case step motor, the current in the coil A; brushless if the current in the first coil, and in the case of a single DC.

5.6.2.5 int WindingCurrentB

In the case step motor, the current in the coil B; brushless if the current in the second coil, and in the case of DC is not used.

5.6.2.6 int WindingCurrentC

In the case of a brushless, the current in the third winding, in the case step motor and DC is not used.

5.6.2.7 int WindingVoltageA

In the case step motor, the voltage across the winding A; in the case of a brushless, the voltage on the first coil, in the case of the only DC.

5.6.2.8 int WindingVoltageB

In the case step motor, the voltage across the winding B; in case of a brushless, the voltage on the second winding, and in the case of DC is not used.

5.6.2.9 int WindingVoltageC

In the case of a brushless, the voltage on the third winding, in the case step motor and DC is not used.

5.7 command_add_sync_in_action_calb_t Struct Reference

Data Fields

- float Position

  Desired position or shift.
- unsigned int Time

  Time for which you want to achieve the desired position in microseconds.

5.7.1 Field Documentation

5.7.1.1 float Position

Desired position or shift.

5.7.1.2 unsigned int Time

Time for which you want to achieve the desired position in microseconds.

5.8 command_add_sync_in_action_t Struct Reference

This command adds one element of the FIFO commands.

Data Fields

- int Position

  Desired position or shift (whole steps)
- int uPosition

  The fractional part of a position or shift in microsteps.
- unsigned int Time

  Time for which you want to achieve the desired position in microseconds.

5.8.1 Detailed Description

This command adds one element of the FIFO commands.

See Also

command_add_sync_in_action

5.8.2 Field Documentation

5.8.2.1 unsigned int Time

Time for which you want to achieve the desired position in microseconds.

5.8.2.2 int uPosition

The fractional part of a position or shift in microsteps.

Is only used with stepper motor. Range: -255..255.

5.9 command_change_motor_t Struct Reference

Change motor - command for switching output relay.

Data Fields

- unsigned int Motor

  Motor number which it should be switch relay on [0..1].

5.9.1 Detailed Description

Change motor - command for switching output relay.

See Also

command_change_motor

5.10 control_settings_calb_t Struct Reference

Data Fields

- float MaxSpeed [10]

  Array of speeds using with joystick and button control.
- unsigned int Timeout [9]

  timeout[i] is time in ms, after that max_speed[i+1] is applying.
- unsigned int MaxClickTime

  Maximum click time.
- unsigned int Flags

  Control flags.
- float DeltaPosition

  Shift (delta) of position.

5.10.1 Field Documentation

5.10.1.1 unsigned int Flags

Control flags.

5.10.1.2 unsigned int MaxClickTime

Maximum click time.

Prior to the expiration of this time the first speed isn't enabled.

5.10.1.3 float MaxSpeed[10]

Array of speeds using with joystick and button control.

5.10.1.4 unsigned int Timeout[9]

timeout[i] is time in ms, after that max_speed[i+1] is applying.

It is using with buttons control only.

5.11 control_settings_t Struct Reference

Control settings.

Data Fields

- unsigned int MaxSpeed [10]

  Array of speeds (full step) using with joystick and button control.
- unsigned int uMaxSpeed [10]

  Array of speeds (1/256 microstep) using with joystick and button control.
- unsigned int Timeout [9]

  timeout[i] is time in ms, after that max_speed[i+1] is applying.
- unsigned int MaxClickTime

  Maximum click time.
- unsigned int Flags

  Control flags.
- int DeltaPosition

  Shift (delta) of position.
- int uDeltaPosition

  Fractional part of the shift in micro steps.

5.11.1 Detailed Description

Control settings.

This structure contains control parameters. When choosing CTL_MODE=1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i=0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL_MODE=2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout[i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i+1] to acceleration, as usual. The figure above shows the sensitivity of the joystick feature on its position.

See Also

set_control_settings
get_control_settings
get_control_settings, set_control_settings

5.11.2 Field Documentation

5.11.2.1 unsigned int Flags

Control flags.

5.11.2.2 unsigned int MaxClickTime

Maximum click time.

Prior to the expiration of this time the first speed isn't enabled.

5.11.2.3 unsigned int MaxSpeed[10]

Array of speeds (full step) using with joystick and button control.

Range: 0..100000.

5.11.2.4 unsigned int Timeout[9]

timeout[i] is time in ms, after that max_speed[i+1] is applying.

It is using with buttons control only.

5.11.2.5 int uDeltaPosition

Fractional part of the shift in micro steps.

Is only used with stepper motor. Range: -255..255.

5.11.2.6 unsigned int uMaxSpeed[10]

Array of speeds (1/256 microstep) using with joystick and button control.

5.12 controller_name_t Struct Reference

Controller user name and flags of setting.

Data Fields

- char ControllerName [17]

  User conroller name.
- unsigned int CtrlFlags

  Flags of internal controller settings.

5.12.1 Detailed Description

Controller user name and flags of setting.

See Also

get_controller_name, set_controller_name

5.12.2 Field Documentation

5.12.2.1 char ControllerName[17]

User conroller name.

Can be set by user for his/her convinience. Max string length: 16 chars.

5.12.2.2 unsigned int CtrlFlags

Flags of internal controller settings.

5.13 ctp_settings_t Struct Reference

Control position settings(is only used with stepper motor).

Data Fields

- unsigned int CTPMinError

  Minimum contrast steps from step motor encoder position, wich set STATE_CTP_ERROR flag.
- unsigned int CTPFlags

  Position control flags.

5.13.1 Detailed Description

Control position settings(is only used with stepper motor).

When controlling the step motor with encoder (CTP_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE_CTP_ERROR and set ALARM state. When controlling the step motor with speed sensor (CTP_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE_CTP_ERROR and set ALARM state.

See Also

set_ctp_settings
get_ctp_settings
get_ctp_settings, set_ctp_settings

5.13.2 Field Documentation

5.13.2.1 unsigned int CTPFlags

Position control flags.

5.13.2.2 unsigned int CTPMinError

Minimum contrast steps from step motor encoder position, wich set STATE_CTP_ERROR flag.

Measured in steps step motor.

5.14 debug_read_t Struct Reference

Debug data.

Data Fields

- uint8_t DebugData [128]

  Arbitrary debug data.

5.14.1 Detailed Description

Debug data.

These data are used for device debugging by manufacturer only.

See Also

get_debug_read

5.14.2 Field Documentation

5.14.2.1 uint8_t DebugData[128]

Arbitrary debug data.

5.15 debug_write_t Struct Reference

Debug data.

Data Fields

- uint8_t DebugData [128]

  Arbitrary debug data.

5.15.1 Detailed Description

Debug data.

These data are used for device debugging by manufacturer only.

See Also

set_debug_write

5.15.2 Field Documentation

5.15.2.1 uint8_t DebugData[128]

Arbitrary debug data.

5.16 device_information_t Struct Reference

Read command controller information.

Data Fields

- char Manufacturer [5]

  Manufacturer.
- char ManufacturerId [3]

  Manufacturer id.
- char ProductDescription [9]

  Product description.
- unsigned int Major

  The major number of the hardware version.
- unsigned int Minor

Minor number of the hardware version.

- unsigned int Release

  Number of edits this release of hardware.

5.16.1 Detailed Description

Read command controller information.

The controller responds to this command in any state. Manufacturer field for all XI** devices should contain the string "XIMC" (validation is performed on it) The remaining fields contain information about the device.

See Also

get_device_information
get_device_information_impl

5.16.2 Field Documentation

5.16.2.1 unsigned int Major

The major number of the hardware version.

5.16.2.2 unsigned int Minor

Minor number of the hardware version.

5.16.2.3 unsigned int Release

Number of edits this release of hardware.

5.17 device_network_information_t Struct Reference

Device network information structure.

Data Fields

- uint32_t ipv4

  IPv4 address, passed in network byte order (big-endian byte order)
- char nodename [16]

  Name of the Bindy node which hosts the device.
- uint32_t axis_state

  Flags representing device state.
- char locker_username [16]

  Name of the user who locked the device (if any)
- char locker_nodename [16]

  Bindy node name, which was used to lock the device (if any)
- time_t locked_time

  Time the lock was acquired at (UTC, microseconds since the epoch)

5.17.1 Detailed Description

Device network information structure.

5.18 edges_settings_calb_t Struct Reference

Data Fields

- unsigned int BorderFlags

  Border flags.
- unsigned int EnderFlags

  Limit switches flags.
- float LeftBorder

  Left border position, used if BORDER_IS_ENCODER flag is set.
- float RightBorder

  Right border position, used if BORDER_IS_ENCODER flag is set.

5.18.1 Field Documentation

5.18.1.1 unsigned int BorderFlags

Border flags.

5.18.1.2 unsigned int EnderFlags

Limit switches flags.

5.18.1.3 float LeftBorder

Left border position, used if BORDER_IS_ENCODER flag is set.

5.18.1.4 float RightBorder

Right border position, used if BORDER_IS_ENCODER flag is set.

5.19 edges_settings_t Struct Reference

Edges settings.

Data Fields

- unsigned int BorderFlags

  Border flags.
- unsigned int EnderFlags

  Limit switches flags.
- int LeftBorder

  Left border position, used if BORDER_IS_ENCODER flag is set.
- int uLeftBorder

Left border position in 1/256 microsteps(used with stepper motor only).

- int RightBorder

  Right border position, used if BORDER_IS_ENCODER flag is set.
- int uRightBorder

  Right border position in 1/256 microsteps.

5.19.1 Detailed Description

Edges settings.

This structure contains border and limit switches settings. Please load new engine settings when you change positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

See Also

set_edges_settings
get_edges_settings
get_edges_settings, set_edges_settings

5.19.2 Field Documentation

5.19.2.1 unsigned int BorderFlags

Border flags.

5.19.2.2 unsigned int EnderFlags

Limit switches flags.

5.19.2.3 int LeftBorder

Left border position, used if BORDER_IS_ENCODER flag is set.

5.19.2.4 int RightBorder

Right border position, used if BORDER_IS_ENCODER flag is set.

5.19.2.5 int uLeftBorder

Left border position in 1/256 microsteps(used with stepper motor only).

Range: -255..255.

5.19.2.6 int uRightBorder

Right border position in 1/256 microsteps.

Used with stepper motor only. Range: -255..255.

5.20 encoder_information_t Struct Reference

Encoder information.

Data Fields

- char Manufacturer [17]

  Manufacturer.
- char PartNumber [25]

  Series and PartNumber.

5.20.1 Detailed Description

Encoder information.

See Also

set_encoder_information
get_encoder_information
get_encoder_information, set_encoder_information

5.20.2 Field Documentation

5.20.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

5.20.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

5.21 encoder_settings_t Struct Reference

Encoder settings.

Data Fields

- float MaxOperatingFrequency

  Max operation frequency (kHz).
- float SupplyVoltageMin

  Minimum supply voltage (V).
- float SupplyVoltageMax

  Maximum supply voltage (V).
- float MaxCurrentConsumption

  Max current consumption (mA).
- unsigned int PPR

  The number of counts per revolution.
- unsigned int EncoderSettings

  Encoder settings flags.

5.21.1 Detailed Description

Encoder settings.

See Also

set_encoder_settings
get_encoder_settings
get_encoder_settings, set_encoder_settings

5.21.2 Field Documentation

5.21.2.1 unsigned int EncoderSettings

Encoder settings flags.

5.21.2.2 float MaxCurrentConsumption

Max current consumption (mA).

Data type: float.

5.21.2.3 float MaxOperatingFrequency

Max operation frequency (kHz).

Data type: float.

5.21.2.4 float SupplyVoltageMax

Maximum supply voltage (V).

Data type: float.

5.21.2.5 float SupplyVoltageMin

Minimum supply voltage (V).

Data type: float.

5.22 engine_settings_calb_t Struct Reference

Data Fields

- unsigned int NomVoltage

  Rated voltage in tens of mV.
- unsigned int NomCurrent

  Rated current.
- float NomSpeed

  Nominal speed.
- unsigned int EngineFlags

  Flags of engine settings.
- float Antiplay

Number of pulses or steps for backlash (play) compensation procedure.
- unsigned int MicrostepMode

  Flags of microstep mode.
- unsigned int StepsPerRev

  Number of full steps per revolution(Used with stepper motor only).

5.22.1 Field Documentation

5.22.1.1 float Antiplay

Number of pulses or steps for backlash (play) compensation procedure.

Used if ENGINE_ANTIPLAY flag is set.

5.22.1.2 unsigned int EngineFlags

Flags of engine settings.

5.22.1.3 unsigned int MicrostepMode

Flags of microstep mode.

5.22.1.4 unsigned int NomCurrent

Rated current.

Controller will keep current consumed by motor below this value if ENGINE_LIMIT_CURR flag is set. Range: 15..8000

5.22.1.5 float NomSpeed

Nominal speed.

Controller will keep motor speed below this value if ENGINE_LIMIT_RPM flag is set.

5.22.1.6 unsigned int NomVoltage

Rated voltage in tens of mV.

Controller will keep the voltage drop on motor below this value if ENGINE_LIMIT_VOLT flag is set (used with DC only).

5.22.1.7 unsigned int StepsPerRev

Number of full steps per revolution(Used with stepper motor only).

Range: 1..65535.

5.23 engine_settings_t Struct Reference

Movement limitations and settings, related to the motor.

Data Fields

- unsigned int NomVoltage

    Rated voltage in tens of mV.
- unsigned int NomCurrent

    Rated current.
- unsigned int NomSpeed

    Nominal (maximum) speed (in whole steps/s or rpm for DC and stepper motor as a master encoder).
- unsigned int uNomSpeed

    The fractional part of a nominal speed in microsteps (is only used with stepper motor).
- unsigned int EngineFlags

    Flags of engine settings.
- int Antiplay

    Number of pulses or steps for backlash (play) compensation procedure.
- unsigned int MicrostepMode

    Flags of microstep mode.
- unsigned int StepsPerRev

    Number of full steps per revolution(Used with stepper motor only).

5.23.1 Detailed Description

Movement limitations and settings, related to the motor.

This structure contains useful motor settings. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics. All boards are supplied with standard set of engine setting on controller's flash memory. Please load new engine settings when you change motor, encoder, positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

See Also

set_engine_settings
get_engine_settings
get_engine_settings, set_engine_settings

5.23.2 Field Documentation

5.23.2.1 int Antiplay

Number of pulses or steps for backlash (play) compensation procedure.

Used if ENGINE_ANTIPLAY flag is set.

5.23.2.2 unsigned int EngineFlags

Flags of engine settings.

5.23.2.3 unsigned int MicrostepMode

Flags of microstep mode.

5.23.2.4 unsigned int NomCurrent

Rated current.

Controller will keep current consumed by motor below this value if ENGINE_LIMIT_CURR flag is set. Range: 15..8000

5.23.2.5 unsigned int NomSpeed

Nominal (maximum) speed (in whole steps/s or rpm for DC and stepper motor as a master encoder).

Controller will keep motor shaft RPM below this value if ENGINE_LIMIT_RPM flag is set. Range: 1..100000.

5.23.2.6 unsigned int NomVoltage

Rated voltage in tens of mV.

Controller will keep the voltage drop on motor below this value if ENGINE_LIMIT_VOLT flag is set (used with DC only).

5.23.2.7 unsigned int StepsPerRev

Number of full steps per revolution(Used with stepper motor only).

Range: 1..65535.

5.23.2.8 unsigned int uNomSpeed

The fractional part of a nominal speed in microsteps (is only used with stepper motor).

5.24 entype_settings_t Struct Reference

Engine type and driver type settings.

Data Fields

- unsigned int EngineType

  Flags of engine type.
- unsigned int DriverType

  Flags of driver type.

5.24.1 Detailed Description

Engine type and driver type settings.

Parameters

id	an identifier of device
EngineType	engine type
DriverType	driver type

See Also

get_entype_settings, set_entype_settings

5.24.2 Field Documentation

5.24.2.1 unsigned int DriverType

Flags of driver type.

5.24.2.2 unsigned int EngineType

Flags of engine type.

5.25 extio_settings_t Struct Reference

EXTIO settings.

Data Fields

- unsigned int EXTIOSetupFlags

  External IO setup flags.
- unsigned int EXTIOModeFlags

  External IO mode flags.

5.25.1 Detailed Description

EXTIO settings.

This structure contains all EXTIO settings. By default input event are signalled through rising front and output states are signalled by high logic state.

See Also

get_extio_settings
set_extio_settings
get_extio_settings, set_extio_settings

5.25.2 Field Documentation

5.25.2.1 unsigned int EXTIOModeFlags

External IO mode flags.

5.25.2.2 unsigned int EXTIOSetupFlags

External IO setup flags.

5.26 feedback_settings_t Struct Reference

Feedback settings.

Data Fields

- unsigned int IPS

  The number of encoder counts per shaft revolution.
- unsigned int FeedbackType

  Feedback type.
- unsigned int FeedbackFlags

  Describes feedback flags.
- unsigned int CountsPerTurn

  The number of encoder counts per shaft revolution.

5.26.1 Detailed Description

Feedback settings.

This structure contains feedback settings.

See Also

get_feedback_settings, set_feedback_settings

5.26.2 Field Documentation

5.26.2.1 unsigned int CountsPerTurn

The number of encoder counts per shaft revolution.

Range: 1..4294967295. To use the CountsPerTurn field, write 0 in the IPS field, otherwise the value from the IPS field will be used.

5.26.2.2 unsigned int FeedbackFlags

Describes feedback flags.

5.26.2.3 unsigned int FeedbackType

Feedback type.

5.26.2.4 unsigned int IPS

The number of encoder counts per shaft revolution.

Range: 1..655535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version.

5.27 gear_information_t Struct Reference

Gear information.

Data Fields

- char Manufacturer [17]

  Manufacturer.
- char PartNumber [25]

  Series and PartNumber.

5.27.1 Detailed Description

Gear information.

See Also

set_gear_information
get_gear_information
get_gear_information, set_gear_information

5.27.2 Field Documentation

5.27.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

5.27.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

5.28 gear_settings_t Struct Reference

Gear setings.

Data Fields

- float ReductionIn

  Input reduction coefficient.
- float ReductionOut

  Output reduction coefficient.
- float RatedInputTorque

  Max continuous torque (N m).
- float RatedInputSpeed

  Max speed on the input shaft (rpm).
- float MaxOutputBacklash

  Output backlash of the reduction gear(degree).
- float InputInertia

  Equivalent input gear inertia (g cm2).
- float Efficiency

  Reduction gear efficiency (%).

5.28.1 Detailed Description

Gear setings.

See Also

set_gear_settings
get_gear_settings
get_gear_settings, set_gear_settings

5.28.2 Field Documentation

5.28.2.1 float Efficiency

Reduction gear efficiency (%).

Data type: float.

5.28.2.2 float InputInertia

Equivalent input gear inertia (g cm2).

Data type: float.

5.28.2.3 float MaxOutputBacklash

Output backlash of the reduction gear(degree).

Data type: float.

5.28.2.4 float RatedInputSpeed

Max speed on the input shaft (rpm).

Data type: float.

5.28.2.5 float RatedInputTorque

Max continuous torque (N m).

Data type: float.

5.28.2.6 float ReductionIn

Input reduction coefficient.

(Output = (ReductionOut / ReductionIn) $*$ Input) Data type: float.

5.28.2.7 float ReductionOut

Output reduction coefficient.

(Output = (ReductionOut / ReductionIn) $*$ Input) Data type: float.

5.29 get_position_calb_t Struct Reference

Data Fields

- float Position

  The position in the engine.
- long_t EncPosition

  Encoder position.

5.29.1 Field Documentation

5.29.1.1 long_t EncPosition

Encoder position.

5.29.1.2 float Position

The position in the engine.

5.30 get_position_t Struct Reference

Position information.

Data Fields

- int Position

  The position of the whole steps in the engine.
- int uPosition

  Microstep position is only used with stepper motors.
- long_t EncPosition

  Encoder position.

5.30.1 Detailed Description

Position information.

Useful structure that contains position value in steps and micro for stepper motor and encoder steps of all engines.

See Also

get_position

5.30.2 Field Documentation

5.30.2.1 long_t EncPosition

Encoder position.

5.31 globally_unique_identifier_t Struct Reference

Globally unique identifier.

Data Fields

- unsigned int UniqueID0

  Unique ID 0.
- unsigned int UniqueID1

  Unique ID 1.
- unsigned int UniqueID2

  Unique ID 2.
- unsigned int UniqueID3

  Unique ID 3.

5.31.1 Detailed Description

Globally unique identifier.

See Also

get_globally_unique_identifier

5.31.2 Field Documentation

5.31.2.1 unsigned int UniqueID0

Unique ID 0.

5.31.2.2 unsigned int UniqueID1

Unique ID 1.

5.31.2.3 unsigned int UniqueID2

Unique ID 2.

5.31.2.4 unsigned int UniqueID3

Unique ID 3.

5.32 hallsensor_information_t Struct Reference

Hall sensor information.

Data Fields

- char Manufacturer [17]

  Manufacturer.
- char PartNumber [25]

  Series and PartNumber.

5.32.1 Detailed Description

Hall sensor information.

See Also

set_hallsensor_information
get_hallsensor_information
get_hallsensor_information, set_hallsensor_information

5.32.2 Field Documentation

5.32.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

5.32.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

5.33 hallsensor_settings_t Struct Reference

Hall sensor settings.

Data Fields

- float MaxOperatingFrequency
  Max operation frequency (kHz).
- float SupplyVoltageMin
  Minimum supply voltage (V).
- float SupplyVoltageMax
  Maximum supply voltage (V).
- float MaxCurrentConsumption
  Max current consumption (mA).
- unsigned int PPR
  The number of counts per revolution.

5.33.1 Detailed Description

Hall sensor settings.

See Also

set_hallsensor_settings
get_hallsensor_settings
get_hallsensor_settings, set_hallsensor_settings

5.33.2 Field Documentation

5.33.2.1 float MaxCurrentConsumption

Max current consumption (mA).

Data type: float.

5.33.2.2 float MaxOperatingFrequency

Max operation frequency (kHz).

Data type: float.

5.33.2.3 float SupplyVoltageMax

Maximum supply voltage (V).

Data type: float.

5.33.2.4 float SupplyVoltageMin

Minimum supply voltage (V).

Data type: float.

5.34 home_settings_calb_t Struct Reference

Data Fields

- float FastHome

  Speed used for first motion.
- float SlowHome

  Speed used for second motion.
- float HomeDelta

  Distance from break point.
- unsigned int HomeFlags

  Home settings flags.

5.34.1 Field Documentation

5.34.1.1 float FastHome

Speed used for first motion.

5.34.1.2 float HomeDelta

Distance from break point.

5.34.1.3 unsigned int HomeFlags

Home settings flags.

5.34.1.4 float SlowHome

Speed used for second motion.

5.35 home_settings_t Struct Reference

Position calibration settings.

Data Fields

- unsigned int FastHome

  Speed used for first motion.
- unsigned int uFastHome

  Part of the speed for first motion, microsteps.
- unsigned int SlowHome

  Speed used for second motion.
- unsigned int uSlowHome

  Part of the speed for second motion, microsteps.
- int HomeDelta

  Distance from break point.
- int uHomeDelta

  Part of the delta distance, microsteps.
- unsigned int HomeFlags

  Home settings flags.

5.35.1 Detailed Description

Position calibration settings.

This structure contains settings used in position calibrating. It specify behaviour of calibrating position.

See Also

get_home_settings
set_home_settings
command_home
get_home_settings, set_home_settings

5.35.2 Field Documentation

5.35.2.1 unsigned int FastHome

Speed used for first motion.

Range: 0..100000.

5.35.2.2 int HomeDelta

Distance from break point.

5.35.2.3 unsigned int HomeFlags

Home settings flags.

5.35.2.4 unsigned int SlowHome

Speed used for second motion.

Range: 0..100000.

5.35.2.5 unsigned int uFastHome

Part of the speed for first motion, microsteps.

5.35.2.6 int uHomeDelta

Part of the delta distance, microsteps.

Range: -255..255.

5.35.2.7 unsigned int uSlowHome

Part of the speed for second motion, microsteps.

5.36 init_random_t Struct Reference

Random key.

Data Fields

- uint8_t key [16]

  Random key.

5.36.1 Detailed Description

Random key.

Structure that contains random key used in encryption of WKEY and SSER command contents.

See Also

get_init_random

5.36.2 Field Documentation

5.36.2.1 uint8_t key[16]

Random key.

5.37 joystick_settings_t Struct Reference

Joystick settings.

Data Fields

- unsigned int JoyLowEnd

  Joystick lower end position.
- unsigned int JoyCenter

  Joystick center position.
- unsigned int JoyHighEnd

  Joystick higher end position.
- unsigned int ExpFactor

  Exponential nonlinearity factor.
- unsigned int DeadZone

  Joystick dead zone.
- unsigned int JoyFlags

  Joystick flags.

5.37.1 Detailed Description

Joystick settings.

This structure contains joystick parameters. If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy.

See Also

set_joystick_settings
get_joystick_settings
get_joystick_settings, set_joystick_settings

5.37.2 Field Documentation

5.37.2.1 unsigned int DeadZone

Joystick dead zone.

5.37.2.2 unsigned int ExpFactor

Exponential nonlinearity factor.

5.37.2.3 unsigned int JoyCenter

Joystick center position.

Range: 0..10000.

5.37.2.4 unsigned int JoyFlags

Joystick flags.

5.37.2.5 unsigned int JoyHighEnd

Joystick higher end position.

Range: 0..10000.

5.37.2.6 unsigned int JoyLowEnd

Joystick lower end position.

Range: 0..10000.

5.38 measurements_t Struct Reference

The buffer holds no more than 25 points.

Data Fields

- int Speed [25]
  Current speed.
- int Error [25]
  Current error.
- unsigned int Length
  Length of actual data in buffer.

5.38.1 Detailed Description

The buffer holds no more than 25 points.

The exact length of the received buffer is reflected in the Length field.

See Also

measurements
get_measurements

5.38.2 Field Documentation

5.38.2.1 int Error[25]

Current error.

5.38.2.2 unsigned int Length

Length of actual data in buffer.

5.38.2.3 int Speed[25]

Current speed.

5.39 motor_information_t Struct Reference

motor information.

Data Fields

- char Manufacturer [17]

  Manufacturer.
- char PartNumber [25]

  Series and PartNumber.

5.39.1 Detailed Description

motor information.

See Also

set_motor_information
get_motor_information
get_motor_information, set_motor_information

5.39.2 Field Documentation

5.39.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

5.39.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

5.40 motor_settings_t Struct Reference

Physical characteristics and limitations of the motor.

Data Fields

- unsigned int MotorType

  Motor Type flags.
- unsigned int ReservedField

  Reserved.
- unsigned int Poles

  Number of pole pairs for DC or BLDC motors or number of steps per rotation for stepper motor.
- unsigned int Phases

  Number of phases for BLDC motors.
- float NominalVoltage

  Nominal voltage on winding (B).

- float NominalCurrent

    Maximum direct current in winding for DC and BLDC engines, nominal current in windings for stepper motor (A).
- float NominalSpeed

    Not used.
- float NominalTorque

    Nominal torque(mN m).
- float NominalPower

    Nominal power(W).
- float WindingResistance

    Resistance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(Ohm).
- float WindingInductance

    Inductance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(mH).
- float RotorInertia

    Rotor inertia(g cm2).
- float StallTorque

    Torque hold position for a stepper motor or torque at a motionless rotor for other types of engines (mN m).
- float DetentTorque

    Holding torque position with un-powered coils (mN m).
- float TorqueConstant

    Torque constant, which determines the aspect ratio of maximum moment of force from the rotor current flowing in the coil (mN m / A).
- float SpeedConstant

    Velocity constant, which determines the value or amplitude of the induced voltage on the motion of DC or BLDC motor (rpm / V) or stepper motor (steps/s / V).
- float SpeedTorqueGradient

    Speed torque gradient (rpm / mN m).
- float MechanicalTimeConstant

    Mechanical time constant (ms).
- float MaxSpeed

    The maximum speed for stepper motors (steps/s) or DC and BLDC motors (rmp).
- float MaxCurrent

    The maximum current in the winding (A).
- float MaxCurrentTime

    Safe duration of overcurrent in the winding (ms).
- float NoLoadCurrent

    The current consumption in idle mode (A).
- float NoLoadSpeed

    Idle speed (rpm).

5.40.1 Detailed Description

Physical characteristics and limitations of the motor.

See Also

set_motor_settings
get_motor_settings
get_motor_settings, set_motor_settings

5.40.2 Field Documentation

5.40.2.1 float DetentTorque

Holding torque position with un-powered coils (mN m).

Data type: float.

5.40.2.2 float MaxCurrent

The maximum current in the winding (A).

Data type: float.

5.40.2.3 float MaxCurrentTime

Safe duration of overcurrent in the winding (ms).

Data type: float.

5.40.2.4 float MaxSpeed

The maximum speed for stepper motors (steps/s) or DC and BLDC motors (rmp).

Data type: float.

5.40.2.5 float MechanicalTimeConstant

Mechanical time constant (ms).

Data type: float.

5.40.2.6 unsigned int MotorType

Motor Type flags.

5.40.2.7 float NoLoadCurrent

The current consumption in idle mode (A).

Used for DC and BLDC motors. Data type: float.

5.40.2.8 float NoLoadSpeed

Idle speed (rpm).

Used for DC and BLDC motors. Data type: float.

5.40.2.9 float NominalCurrent

Maximum direct current in winding for DC and BLDC engines, nominal current in windings for stepper motor (A).

Data type: float.

5.40.2.10 float NominalPower

Nominal power(W).

Used for DC and BLDC engine. Data type: float.

5.40.2.11 float NominalSpeed

Not used.

Nominal speed(rpm). Used for DC and BLDC engine. Data type: float.

5.40.2.12 float NominalTorque

Nominal torque(mN m).

Used for DC and BLDC engine. Data type: float.

5.40.2.13 float NominalVoltage

Nominal voltage on winding (B).

Data type: float

5.40.2.14 unsigned int Phases

Number of phases for BLDC motors.

5.40.2.15 unsigned int Poles

Number of pole pairs for DC or BLDC motors or number of steps per rotation for stepper motor.

5.40.2.16 float RotorInertia

Rotor inertia(g cm2).

Data type: float.

5.40.2.17 float SpeedConstant

Velocity constant, which determines the value or amplitude of the induced voltage on the motion of DC or BLDC motor (rpm / V) or stepper motor (steps/s / V).

Data type: float.

5.40.2.18 float SpeedTorqueGradient

Speed torque gradient (rpm / mN m).

Data type: float.

5.40.2.19 float StallTorque

Torque hold position for a stepper motor or torque at a motionless rotor for other types of engines (mN m).

Data type: float.

5.40.2.20 float TorqueConstant

Torque constant, which determines the aspect ratio of maximum moment of force from the rotor current flowing in the coil (mN m / A).

Used mainly for DC motors. Data type: float.

5.40.2.21 float WindingInductance

Inductance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(mH).

Data type: float.

5.40.2.22 float WindingResistance

Resistance of windings for DC engine, each of two windings for stepper motor or each of there windings for BLDC engine(Ohm).

Data type: float.

5.41 move_settings_calb_t Struct Reference

Data Fields

- float Speed

  Target speed.
- float Accel

  Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC).
- float Decel

  Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC).
- float AntiplaySpeed

  Speed in antiplay mode.

5.41.1 Field Documentation

5.41.1.1 float Accel

Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC).

5.41.1.2 float AntiplaySpeed

Speed in antiplay mode.

5.41.1.3 float Decel

Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC).

5.41.1.4 float Speed

Target speed.

5.42 move_settings_t Struct Reference

Move settings.

Data Fields

- unsigned int Speed

    Target speed (for stepper motor: steps/s, for DC: rpm).
- unsigned int uSpeed

    Target speed in microstep fractions/s.
- unsigned int Accel

    Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC).
- unsigned int Decel

    Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC).
- unsigned int AntiplaySpeed

    Speed in antiplay mode, full steps/s(stepper motor) or RPM(DC).
- unsigned int uAntiplaySpeed

    Speed in antiplay mode, 1/256 microsteps/s.

5.42.1 Detailed Description

Move settings.

See Also

set_move_settings
get_move_settings
get_move_settings, set_move_settings

5.42.2 Field Documentation

5.42.2.1 unsigned int Accel

Motor shaft acceleration, steps/s^2(stepper motor) or RPM/s(DC).

Range: 1..65535.

5.42.2.2 unsigned int AntiplaySpeed

Speed in antiplay mode, full steps/s(stepper motor) or RPM(DC).

Range: 0..100000.

5.42.2.3 unsigned int Decel

Motor shaft deceleration, steps/s^2(stepper motor) or RPM/s(DC).

Range: 1..65535.

5.42.2.4 unsigned int Speed

Target speed (for stepper motor: steps/s, for DC: rpm).

Range: 0..100000.

5.42.2.5 unsigned int uAntiplaySpeed

Speed in antiplay mode, 1/256 microsteps/s.

Used with stepper motor only.

5.42.2.6 unsigned int uSpeed

Target speed in microstep fractions/s.

Using with stepper motor only.

5.43 nonvolatile_memory_t Struct Reference

Userdata for save into FRAM.

Data Fields

- unsigned int UserData [7]

  User data.

5.43.1 Detailed Description

Userdata for save into FRAM.

See Also

get_nonvolatile_memory, set_nonvolatile_memory

5.43.2 Field Documentation

5.43.2.1 unsigned int UserData[7]

User data.

Can be set by user for his/her convinience. Each element of the array stores only 32 bits of user data. This is important on systems where an int type contains more than 4 bytes. For example that all amd64 systems.

5.44 pid_settings_t Struct Reference

PID settings.

Data Fields

- unsigned int KpU

  Proportional gain for voltage PID routine.
- unsigned int KiU

  Integral gain for voltage PID routine.
- unsigned int KdU

  Differential gain for voltage PID routine.
- float Kpf

Proportional gain for BLDC position PID routine.

- float Kif

  Integral gain for BLDC position PID routine.
- float Kdf

  Differential gain for BLDC position PID routine.

5.44.1 Detailed Description

PID settings.

This structure contains factors for PID routine. It specify behaviour of PID routine for voltage. These factors are slightly different for different positioners. All boards are supplied with standard set of PID setting on controller's flash memory. Please load new PID settings when you change positioner. Please note that wrong PID settings lead to device malfunction.

See Also

set_pid_settings
get_pid_settings
get_pid_settings, set_pid_settings

5.45 power_settings_t Struct Reference

Step motor power settings.

Data Fields

- unsigned int HoldCurrent

  Current in holding regime, percent of nominal.
- unsigned int CurrReductDelay

  Time in ms from going to STOP state to reducting current.
- unsigned int PowerOffDelay

  Time in s from going to STOP state to turning power off.
- unsigned int CurrentSetTime

  Time in ms to reach nominal current.
- unsigned int PowerFlags

  Flags of power settings of stepper motor.

5.45.1 Detailed Description

Step motor power settings.

See Also

set_move_settings
get_move_settings
get_power_settings, set_power_settings

5.45.2 Field Documentation

5.45.2.1 unsigned int CurrentSetTime

Time in ms to reach nominal current.

5.45.2.2 unsigned int CurrReductDelay

Time in ms from going to STOP state to reducting current.

5.45.2.3 unsigned int HoldCurrent

Current in holding regime, percent of nominal.

Range: 0..100.

5.45.2.4 unsigned int PowerFlags

Flags of power settings of stepper motor.

5.45.2.5 unsigned int PowerOffDelay

Time in s from going to STOP state to turning power off.

5.46 secure_settings_t Struct Reference

This structure contains raw analog data from ADC embedded on board.

Data Fields

- unsigned int LowUpwrOff
  Lower voltage limit to turn off the motor, tens of mV.
- unsigned int CriticalIpwr
  Maximum motor current which triggers ALARM state, in mA.
- unsigned int CriticalUpwr
  Maximum motor voltage which triggers ALARM state, tens of mV.
- unsigned int CriticalT
  Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius.
- unsigned int CriticalIusb
  Maximum USB current which triggers ALARM state, in mA.
- unsigned int CriticalUusb
  Maximum USB voltage which triggers ALARM state, tens of mV.
- unsigned int MinimumUusb
  Minimum USB voltage which triggers ALARM state, tens of mV.
- unsigned int Flags
  Flags of secure settings.

5.46.1 Detailed Description

This structure contains raw analog data from ADC embedded on board.

These data used for device testing and deep recalibraton by manufacturer only.

See Also

get_secure_settings
set_secure_settings
get_secure_settings, set_secure_settings

5.46.2 Field Documentation

5.46.2.1 unsigned int CriticalIpwr

Maximum motor current which triggers ALARM state, in mA.

5.46.2.2 unsigned int CriticalIusb

Maximum USB current which triggers ALARM state, in mA.

5.46.2.3 unsigned int CriticalT

Maximum temperature, which triggers ALARM state, in tenths of degrees Celcius.

5.46.2.4 unsigned int CriticalUpwr

Maximum motor voltage which triggers ALARM state, tens of mV.

5.46.2.5 unsigned int CriticalUusb

Maximum USB voltage which triggers ALARM state, tens of mV.

5.46.2.6 unsigned int Flags

Flags of secure settings.

5.46.2.7 unsigned int LowUpwrOff

Lower voltage limit to turn off the motor, tens of mV.

5.46.2.8 unsigned int MinimumUusb

Minimum USB voltage which triggers ALARM state, tens of mV.

5.47 serial_number_t Struct Reference

Serial number structure and hardware version.

Data Fields

- unsigned int SN

  New board serial number.
- uint8_t Key [32]

  Protection key (256 bit).
- unsigned int Major

  The major number of the hardware version.
- unsigned int Minor

  Minor number of the hardware version.
- unsigned int Release

  Number of edits this release of hardware.

5.47.1 Detailed Description

Serial number structure and hardware version.

The structure keep new serial number, hardware version and valid key. The SN and hardware version are changed and saved when transmitted key matches stored key. Can be used by manufacturer only.

See Also

set_serial_number

5.47.2 Field Documentation

5.47.2.1 uint8_t Key[32]

Protection key (256 bit).

5.47.2.2 unsigned int Major

The major number of the hardware version.

5.47.2.3 unsigned int Minor

Minor number of the hardware version.

5.47.2.4 unsigned int Release

Number of edits this release of hardware.

5.47.2.5 unsigned int SN

New board serial number.

5.48 set_position_calb_t Struct Reference

Data Fields

- float Position

  The position in the engine.
- long_t EncPosition

  Encoder position.
- unsigned int PosFlags

  Position setting flags.

5.48.1 Field Documentation

5.48.1.1 long_t EncPosition

Encoder position.

5.48.1.2 unsigned int PosFlags

Position setting flags.

5.48.1.3 float Position

The position in the engine.

5.49 set_position_t Struct Reference

Position information.

Data Fields

- int Position

  The position of the whole steps in the engine.
- int uPosition

  Microstep position is only used with stepper motors.
- long_t EncPosition

  Encoder position.
- unsigned int PosFlags

  Position setting flags.

5.49.1 Detailed Description

Position information.

Useful structure that contains position value in steps and micro for stepper motor and encoder steps of all engines.

See Also

set_position

5.49.2 Field Documentation

5.49.2.1 long_t EncPosition

Encoder position.

5.49.2.2 unsigned int PosFlags

Position setting flags.

5.50 stage_information_t Struct Reference

Stage information.

Data Fields

- char Manufacturer [17]

  Manufacturer.
- char PartNumber [25]

  Series and PartNumber.

5.50.1 Detailed Description

Stage information.

See Also

set_stage_information
get_stage_information
get_stage_information, set_stage_information

5.50.2 Field Documentation

5.50.2.1 char Manufacturer[17]

Manufacturer.

Max string length: 16 chars.

5.50.2.2 char PartNumber[25]

Series and PartNumber.

Max string length: 24 chars.

5.51 stage_name_t Struct Reference

Stage user name.

Data Fields

- char PositionerName [17]

  User positioner name.

5.51.1 Detailed Description

Stage user name.

See Also

get_stage_name, set_stage_name

5.51.2 Field Documentation

5.51.2.1 char PositionerName[17]

User positioner name.

Can be set by user for his/her convinience. Max string length: 16 chars.

5.52 stage_settings_t Struct Reference

Stage settings.

Data Fields

- float LeadScrewPitch

  Lead screw pitch (mm).
- char Units [9]

  Units for MaxSpeed and TravelRange fields of the structure (steps, degrees, mm, ...).
- float MaxSpeed

  Max speed (Units/c).
- float TravelRange

  Travel range (Units).
- float SupplyVoltageMin

  Supply voltage minimum (V).
- float SupplyVoltageMax

  Supply voltage maximum (V).
- float MaxCurrentConsumption

  Max current consumption (A).
- float HorizontalLoadCapacity

  Horizontal load capacity (kg).
- float VerticalLoadCapacity

  Vertical load capacity (kg).

5.52.1 Detailed Description

Stage settings.

See Also

set_stage_settings
get_stage_settings
get_stage_settings, set_stage_settings

5.52.2 Field Documentation

5.52.2.1 float HorizontalLoadCapacity

Horizontal load capacity (kg).

Data type: float.

5.52.2.2 float LeadScrewPitch

Lead screw pitch (mm).

Data type: float.

5.52.2.3 float MaxCurrentConsumption

Max current consumption (A).

Data type: float.

5.52.2.4 float MaxSpeed

Max speed (Units/c).

Data type: float.

5.52.2.5 float SupplyVoltageMax

Supply voltage maximum (V).

Data type: float.

5.52.2.6 float SupplyVoltageMin

Supply voltage minimum (V).

Data type: float.

5.52.2.7 float TravelRange

Travel range (Units).

Data type: float.

5.52.2.8 char Units[9]

Units for MaxSpeed and TravelRange fields of the structure (steps, degrees, mm, ...).

Max string length: 8 chars.

5.52.2.9 float VerticalLoadCapacity

Vertical load capacity (kg).

Data type: float.

5.53 status_calb_t Struct Reference

Data Fields

- unsigned int MoveSts

    Flags of move state.
- unsigned int MvCmdSts

Move command state.

- unsigned int PWRSts

  Flags of power state of stepper motor.
- unsigned int EncSts

  Encoder state.
- unsigned int WindSts

  Winding state.
- float CurPosition

  Current position.
- long_t EncPosition

  Current encoder position.
- float CurSpeed

  Motor shaft speed.
- int Ipwr

  Engine current.
- int Upwr

  Power supply voltage, tens of mV.
- int Iusb

  USB current consumption.
- int Uusb

  USB voltage, tens of mV.
- int CurT

  Temperature in tenths of degrees C.
- unsigned int Flags

  Status flags.
- unsigned int GPIOFlags

  Status flags.
- unsigned int CmdBufFreeSpace

  This field shows the amount of free cells buffer synchronization chain.

5.53.1 Field Documentation

5.53.1.1 unsigned int CmdBufFreeSpace

This field shows the amount of free cells buffer synchronization chain.

5.53.1.2 float CurPosition

Current position.

5.53.1.3 float CurSpeed

Motor shaft speed.

5.53.1.4 int CurT

Temperature in tenths of degrees C.

5.53.1.5 long_t EncPosition

Current encoder position.

5.53.1.6 unsigned int EncSts

Encoder state.

5.53.1.7 unsigned int Flags

Status flags.

5.53.1.8 unsigned int GPIOFlags

Status flags.

5.53.1.9 int Ipwr

Engine current.

5.53.1.10 int Iusb

USB current consumption.

5.53.1.11 unsigned int MoveSts

Flags of move state.

5.53.1.12 unsigned int MvCmdSts

Move command state.

5.53.1.13 unsigned int PWRSts

Flags of power state of stepper motor.

5.53.1.14 int Upwr

Power supply voltage, tens of mV.

5.53.1.15 int Uusb

USB voltage, tens of mV.

5.53.1.16 unsigned int WindSts

Winding state.

5.54 status_t Struct Reference

Device state.

Data Fields

- unsigned int MoveSts

  Flags of move state.
- unsigned int MvCmdSts

  Move command state.
- unsigned int PWRSts

  Flags of power state of stepper motor.
- unsigned int EncSts

  Encoder state.
- unsigned int WindSts

  Winding state.
- int CurPosition

  Current position.
- int uCurPosition

  Step motor shaft position in 1/256 microsteps.
- long_t EncPosition

  Current encoder position.
- int CurSpeed

  Motor shaft speed.
- int uCurSpeed

  Part of motor shaft speed in 1/256 microsteps.
- int Ipwr

  Engine current.
- int Upwr

  Power supply voltage, tens of mV.
- int Iusb

  USB current consumption.
- int Uusb

  USB voltage, tens of mV.
- int CurT

  Temperature in tenths of degrees C.
- unsigned int Flags

  Status flags.
- unsigned int GPIOFlags

  Status flags.
- unsigned int CmdBufFreeSpace

  This field shows the amount of free cells buffer synchronization chain.

5.54.1 Detailed Description

Device state.

Useful structure that contains current controller state, including speed, position and boolean flags.

See Also

get_status_impl

5.54.2 Field Documentation

5.54.2.1 unsigned int CmdBufFreeSpace

This field shows the amount of free cells buffer synchronization chain.

5.54.2.2 int CurPosition

Current position.

5.54.2.3 int CurSpeed

Motor shaft speed.

5.54.2.4 int CurT

Temperature in tenths of degrees C.

5.54.2.5 long_t EncPosition

Current encoder position.

5.54.2.6 unsigned int EncSts

Encoder state.

5.54.2.7 unsigned int Flags

Status flags.

5.54.2.8 unsigned int GPIOFlags

Status flags.

5.54.2.9 int Ipwr

Engine current.

5.54.2.10 int Iusb

USB current consumption.

5.54.2.11 unsigned int MoveSts

Flags of move state.

5.54.2.12 unsigned int MvCmdSts

Move command state.

5.54.2.13 unsigned int PWRSts

Flags of power state of stepper motor.

5.54.2.14 int uCurPosition

Step motor shaft position in 1/256 microsteps.

Used only with stepper motor.

5.54.2.15 int uCurSpeed

Part of motor shaft speed in 1/256 microsteps.

Used only with stepper motor.

5.54.2.16 int Upwr

Power supply voltage, tens of mV.

5.54.2.17 int Uusb

USB voltage, tens of mV.

5.54.2.18 unsigned int WindSts

Winding state.

5.55 sync_in_settings_calb_t Struct Reference

Data Fields

- unsigned int SyncInFlags

  Flags for synchronization input setup.
- unsigned int ClutterTime

  Input synchronization pulse dead time (mks).
- float Position

  Desired position or shift.
- float Speed

  Target speed.

5.55.1 Field Documentation

5.55.1.1 unsigned int ClutterTime

Input synchronization pulse dead time (mks).

5.55.1.2 float Position

Desired position or shift.

5.55.1.3 float Speed

Target speed.

5.55.1.4 unsigned int SyncInFlags

Flags for synchronization input setup.

5.56 sync_in_settings_t Struct Reference

Synchronization settings.

Data Fields

- unsigned int SyncInFlags

  Flags for synchronization input setup.
- unsigned int ClutterTime

  Input synchronization pulse dead time (mks).
- int Position

  Desired position or shift (whole steps)
- int uPosition

  The fractional part of a position or shift in microsteps.
- unsigned int Speed

  Target speed (for stepper motor: steps/s, for DC: rpm).
- unsigned int uSpeed

  Target speed in microsteps/s.

5.56.1 Detailed Description

Synchronization settings.

This structure contains all synchronization settings, modes, periods and flags. It specifes behaviour of input synchronization. All boards are supplied with standard set of these settings.

See Also

get_sync_in_settings
set_sync_in_settings
get_sync_in_settings, set_sync_in_settings

5.56.2 Field Documentation

5.56.2.1 unsigned int ClutterTime

Input synchronization pulse dead time (mks).

5.56.2.2 unsigned int Speed

Target speed (for stepper motor: steps/s, for DC: rpm).

Range: 0..100000.

5.56.2.3 unsigned int SyncInFlags

Flags for synchronization input setup.

5.56.2.4 int uPosition

The fractional part of a position or shift in microsteps.

Is used with stepper motor. Range: -255..255.

5.56.2.5 unsigned int uSpeed

Target speed in microsteps/s.

Using with stepper motor only.

5.57 sync_out_settings_calb_t Struct Reference

Data Fields

- unsigned int SyncOutFlags

  Flags of synchronization output.
- unsigned int SyncOutPulseSteps

  This value specifies duration of output pulse.
- unsigned int SyncOutPeriod

  This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set.
- float Accuracy

  This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.

5.57.1 Field Documentation

5.57.1.1 float Accuracy

This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.

5.57.1.2 unsigned int SyncOutFlags

Flags of synchronization output.

5.57.1.3 unsigned int SyncOutPeriod

This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set.

5.57.1.4 unsigned int SyncOutPulseSteps

This value specifies duration of output pulse.

It is measured microseconds when SYNCOUT_IN_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT_IN_STEPS is set.

5.58 sync_out_settings_t Struct Reference

Synchronization settings.

Data Fields

- unsigned int SyncOutFlags

  Flags of synchronization output.
- unsigned int SyncOutPulseSteps

  This value specifies duration of output pulse.
- unsigned int SyncOutPeriod

  This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set.
- unsigned int Accuracy

  This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.
- unsigned int uAccuracy

  This is the neighborhood around the target coordinates in micro steps (only used with stepper motor).

5.58.1 Detailed Description

Synchronization settings.

This structure contains all synchronization settings, modes, periods and flags. It specifes behaviour of output synchronization. All boards are supplied with standard set of these settings.

See Also

get_sync_out_settings
set_sync_out_settings
get_sync_out_settings, set_sync_out_settings

5.58.2 Field Documentation

5.58.2.1 unsigned int Accuracy

This is the neighborhood around the target coordinates, which is getting hit in the target position and the momentum generated by the stop.

5.58.2.2 unsigned int SyncOutFlags

Flags of synchronization output.

5.58.2.3 unsigned int SyncOutPeriod

This value specifies number of encoder pulses or steps between two output synchronization pulses when SYNCOUT_ONPERIOD is set.

5.58.2.4 unsigned int SyncOutPulseSteps

This value specifies duration of output pulse.

It is measured microseconds when SYNCOUT_IN_STEPS flag is cleared or in encoder pulses or motor steps when SYNCOUT_IN_STEPS is set.

5.58.2.5 unsigned int uAccuracy

This is the neighborhood around the target coordinates in micro steps (only used with stepper motor).

5.59 uart_settings_t Struct Reference

UART settings.

Data Fields

- unsigned int Speed

  UART speed.
- unsigned int UARTSetupFlags

  UART parity flags.

5.59.1 Detailed Description

UART settings.

This structure contains UART settings.

See Also

get_uart_settings
set_uart_settings
get_uart_settings, set_uart_settings

5.59.2 Field Documentation

5.59.2.1 unsigned int UARTSetupFlags

UART parity flags.

Chapter 6

File Documentation

6.1 ximc.h File Reference

Header file for libximc library.

Data Structures

- struct calibration_t

  Calibration companion structure.
- struct device_network_information_t

  Device network information structure.
- struct feedback_settings_t

  Feedback settings.
- struct home_settings_t

  Position calibration settings.
- struct home_settings_calb_t
- struct move_settings_t

  Move settings.
- struct move_settings_calb_t
- struct engine_settings_t

  Movement limitations and settings, related to the motor.
- struct engine_settings_calb_t
- struct entype_settings_t

  Engine type and driver type settings.
- struct power_settings_t

  Step motor power settings.
- struct secure_settings_t

  This structure contains raw analog data from ADC embedded on board.
- struct edges_settings_t

  Edges settings.
- struct edges_settings_calb_t
- struct pid_settings_t

  PID settings.
- struct sync_in_settings_t

  Synchronization settings.
- struct sync_in_settings_calb_t
- struct sync_out_settings_t

Synchronization settings.

- struct sync_out_settings_calb_t
- struct extio_settings_t

  EXTIO settings.
- struct brake_settings_t

  Brake settings.
- struct control_settings_t

  Control settings.
- struct control_settings_calb_t
- struct joystick_settings_t

  Joystick settings.
- struct ctp_settings_t

  Control position settings(is only used with stepper motor).
- struct uart_settings_t

  UART settings.
- struct calibration_settings_t

  Calibration settings.
- struct controller_name_t

  Controller user name and flags of setting.
- struct nonvolatile_memory_t

  Userdata for save into FRAM.
- struct command_add_sync_in_action_t

  This command adds one element of the FIFO commands.
- struct command_add_sync_in_action_calb_t
- struct get_position_t

  Position information.
- struct get_position_calb_t
- struct set_position_t

  Position information.
- struct set_position_calb_t
- struct status_t

  Device state.
- struct status_calb_t
- struct measurements_t

  The buffer holds no more than 25 points.
- struct chart_data_t

  Additional device state.
- struct device_information_t

  Read command controller information.
- struct serial_number_t

  Serial number structure and hardware version.
- struct analog_data_t

  Analog data.
- struct debug_read_t

  Debug data.
- struct debug_write_t

  Debug data.
- struct stage_name_t

  Stage user name.
- struct stage_information_t

  Stage information.

- struct stage_settings_t

  Stage settings.
- struct motor_information_t

  motor information.
- struct motor_settings_t

  Physical characteristics and limitations of the motor.
- struct encoder_information_t

  Encoder information.
- struct encoder_settings_t

  Encoder settings.
- struct hallsensor_information_t

  Hall sensor information.
- struct hallsensor_settings_t

  Hall sensor settings.
- struct gear_information_t

  Gear information.
- struct gear_settings_t

  Gear setings.
- struct accessories_settings_t

  Additional accessories information.
- struct init_random_t

  Random key.
- struct globally_unique_identifier_t

  Globally unique identifier.
- struct command_change_motor_t

  Change motor - command for switching output relay.

Macros

- #define XIMC_API

  Library import macro Macros allows to automatically import function from shared library.
- #define XIMC_CALLCONV

  Library calling convention macros.
- #define XIMC_RETTYPE void∗

  Thread return type.
- #define device_undefined -1

  Handle specified undefined device.

Result statuses

- #define result_ok 0

  success
- #define result_error -1

  generic error
- #define result_not_implemented -2

  function is not implemented
- #define result_value_error -3

  value error
- #define result_nodevice -4

  device is lost

Logging level

- #define LOGLEVEL_ERROR 0x01

  Logging level - error.
- #define LOGLEVEL_WARNING 0x02

  Logging level - warning.
- #define LOGLEVEL_INFO 0x03

  Logging level - info.
- #define LOGLEVEL_DEBUG 0x04

  Logging level - debug.

Enumerate devices flags

- #define ENUMERATE_PROBE 0x01

  Check if a device with OS name name is XIMC device.
- #define ENUMERATE_ALL_COM 0x02

  Check all COM devices.
- #define ENUMERATE_NETWORK 0x04

  Check network devices.

Flags of move state

Specify move states.

See Also

get_status
status_t::move_state
status_t::MoveSts, get_status_impl

- #define MOVE_STATE_MOVING 0x01

  This flag indicates that controller is trying to move the motor.
- #define MOVE_STATE_TARGET_SPEED 0x02

  Target speed is reached, if flag set.
- #define MOVE_STATE_ANTIPLAY 0x04

  Motor is playing compensation, if flag set.

Flags of internal controller settings

See Also

set_controller_name
get_controller_name
controller_name_t::CtrlFlags, get_controller_name, set_controller_name

- #define EEPROM_PRECEDENCE 0x01

  If the flag is set settings from external EEPROM override controller settings.

Flags of power state of stepper motor

Specify power states.

See Also

status_t::power_state
get_status
status_t::PWRSts, get_status_impl

- #define PWR_STATE_UNKNOWN 0x00

  Unknown state, should never happen.
- #define PWR_STATE_OFF 0x01

  Motor windings are disconnected from the driver.
- #define PWR_STATE_NORM 0x03

  Motor windings are powered by nominal current.

- #define PWR_STATE_REDUCT 0x04

  Motor windings are powered by reduced current to lower power consumption.
- #define PWR_STATE_MAX 0x05

  Motor windings are powered by maximum current driver can provide at this voltage.

Status flags

GPIO state flags returned by device query. Contains boolean part of controller state. May be combined with bitwise OR.

See Also

status_t::flags
get_status
status_t::GPIOFlags, get_status_impl

- #define STATE_CONTR 0x00003F

  Flags of controller states.
- #define STATE_ERRC 0x000001

  Command error encountered.
- #define STATE_ERRD 0x000002

  Data integrity error encountered.
- #define STATE_ERRV 0x000004

  Value error encountered.
- #define STATE_EEPROM_CONNECTED 0x000010

  EEPROM with settings is connected.
- #define STATE_IS_HOMED 0x000020

  Calibration performed.
- #define STATE_SECUR 0x73FFC0

  Flags of security.
- #define STATE_ALARM 0x000040

  Controller is in alarm state indicating that something dangerous had happened.
- #define STATE_CTP_ERROR 0x000080

  Control position error(is only used with stepper motor).
- #define STATE_POWER_OVERHEAT 0x000100

  Power driver overheat.
- #define STATE_CONTROLLER_OVERHEAT 0x000200

  Controller overheat.
- #define STATE_OVERLOAD_POWER_VOLTAGE 0x000400

  Power voltage exceeds safe limit.
- #define STATE_OVERLOAD_POWER_CURRENT 0x000800

  Power current exceeds safe limit.
- #define STATE_OVERLOAD_USB_VOLTAGE 0x001000

  USB voltage exceeds safe limit.
- #define STATE_LOW_USB_VOLTAGE 0x002000

  USB voltage is insufficient for normal operation.
- #define STATE_OVERLOAD_USB_CURRENT 0x004000

  USB current exceeds safe limit.
- #define STATE_BORDERS_SWAP_MISSET 0x008000

  Engine stuck at the wrong edge.
- #define STATE_LOW_POWER_VOLTAGE 0x010000

  Power voltage is lower than Low Voltage Protection limit.
- #define STATE_H_BRIDGE_FAULT 0x020000

  Signal from the driver that fault happened.
- #define STATE_CURRENT_MOTOR_BITS 0x0C0000

  Bits indicating the current operating motor on boards with multiple outputs for engine mounting.
- #define STATE_CURRENT_MOTOR0 0x000000

  Motor 0.
- #define STATE_CURRENT_MOTOR1 0x040000

  Motor 1.

- #define STATE_CURRENT_MOTOR2 0x080000

  Motor 2.
- #define STATE_CURRENT_MOTOR3 0x0C0000

  Motor 3.
- #define STATE_WINDING_RES_MISMATCH 0x100000

  The difference between winding resistances is too large.
- #define STATE_ENCODER_FAULT 0x200000

  Signal from the encoder that fault happened.
- #define STATE_MOTOR_CURRENT_LIMIT 0x400000

  Current limit exceeded.
- #define STATE_DIG_SIGNAL 0xFFFF

  Flags of digital signals.
- #define STATE_RIGHT_EDGE 0x0001

  Engine stuck at the right edge.
- #define STATE_LEFT_EDGE 0x0002

  Engine stuck at the left edge.
- #define STATE_BUTTON_RIGHT 0x0004

  Button "right" state (1 if pressed).
- #define STATE_BUTTON_LEFT 0x0008

  Button "left" state (1 if pressed).
- #define STATE_GPIO_PINOUT 0x0010

  External GPIO works as Out, if flag set; otherwise works as In.
- #define STATE_GPIO_LEVEL 0x0020

  State of external GPIO pin.
- #define STATE_BRAKE 0x0200

  State of Brake pin.
- #define STATE_REV_SENSOR 0x0400

  State of Revolution sensor pin.
- #define STATE_SYNC_INPUT 0x0800

  State of Sync input pin.
- #define STATE_SYNC_OUTPUT 0x1000

  State of Sync output pin.
- #define STATE_ENC_A 0x2000

  State of encoder A pin.
- #define STATE_ENC_B 0x4000

  State of encoder B pin.

Encoder state

Encoder state returned by device query.

See Also

status_t::encsts
get_status
status_t::EncSts, get_status_impl

- #define ENC_STATE_ABSENT 0x00

  Encoder is absent.
- #define ENC_STATE_UNKNOWN 0x01

  Encoder state is unknown.
- #define ENC_STATE_MALFUNC 0x02

  Encoder is connected and malfunctioning.
- #define ENC_STATE_REVERS 0x03

  Encoder is connected and operational but counts in other direction.
- #define ENC_STATE_OK 0x04

  Encoder is connected and working properly.

Winding state

Motor winding state returned by device query.

See Also

status_t::windsts
get_status
status_t::WindSts, get_status_impl

- #define WIND_A_STATE_ABSENT 0x00
  Winding A is disconnected.
- #define WIND_A_STATE_UNKNOWN 0x01
  Winding A state is unknown.
- #define WIND_A_STATE_MALFUNC 0x02
  Winding A is short-circuited.
- #define WIND_A_STATE_OK 0x03
  Winding A is connected and working properly.
- #define WIND_B_STATE_ABSENT 0x00
  Winding B is disconnected.
- #define WIND_B_STATE_UNKNOWN 0x10
  Winding B state is unknown.
- #define WIND_B_STATE_MALFUNC 0x20
  Winding B is short-circuited.
- #define WIND_B_STATE_OK 0x30
  Winding B is connected and working properly.

Move command state

Move command (command_move, command_movr, command_left, command_right, command_stop, command_home, command_loft, command_sstp) and its state (run, finished, error).

See Also

status_t::mvcmdsts
get_status
status_t::MvCmdSts, get_status_impl

- #define MVCMD_NAME_BITS 0x3F
  Move command bit mask.
- #define MVCMD_UKNWN 0x00
  Unknown command.
- #define MVCMD_MOVE 0x01
  Command move.
- #define MVCMD_MOVR 0x02
  Command movr.
- #define MVCMD_LEFT 0x03
  Command left.
- #define MVCMD_RIGHT 0x04
  Command rigt.
- #define MVCMD_STOP 0x05
  Command stop.
- #define MVCMD_HOME 0x06
  Command home.
- #define MVCMD_LOFT 0x07
  Command loft.
- #define MVCMD_SSTP 0x08
  Command soft stop.
- #define MVCMD_ERROR 0x40
  Finish state (1 - move command have finished with an error, 0 - move command have finished correctly).
- #define MVCMD_RUNNING 0x80
  Move command state (0 - move command have finished, 1 - move command is being executed).

Flags of engine settings

Specify motor shaft movement algorithm and list of limitations. Flags returned by query of engine settings. May be combined with bitwise OR.

See Also

engine_settings_t::flags
set_engine_settings
get_engine_settings
engine_settings_t::EngineFlags, get_engine_settings, set_engine_settings

- #define ENGINE_REVERSE 0x01
  Reverse flag.
- #define ENGINE_CURRENT_AS_RMS 0x02
  Engine current meaning flag.
- #define ENGINE_MAX_SPEED 0x04
  Max speed flag.
- #define ENGINE_ANTIPLAY 0x08
  Play compensation flag.
- #define ENGINE_ACCEL_ON 0x10
  Acceleration enable flag.
- #define ENGINE_LIMIT_VOLT 0x20
  Maximum motor voltage limit enable flag(is only used with DC motor).
- #define ENGINE_LIMIT_CURR 0x40
  Maximum motor current limit enable flag(is only used with DC motor).
- #define ENGINE_LIMIT_RPM 0x80
  Maximum motor speed limit enable flag.

Flags of microstep mode

Specify settings of microstep mode. Using with step motors. Flags returned by query of engine settings. May be combined with bitwise OR

See Also

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
  Full step mode.
- #define MICROSTEP_MODE_FRAC_2 0x02
  1/2 step mode.
- #define MICROSTEP_MODE_FRAC_4 0x03
  1/4 step mode.
- #define MICROSTEP_MODE_FRAC_8 0x04
  1/8 step mode.
- #define MICROSTEP_MODE_FRAC_16 0x05
  1/16 step mode.
- #define MICROSTEP_MODE_FRAC_32 0x06
  1/32 step mode.
- #define MICROSTEP_MODE_FRAC_64 0x07
  1/64 step mode.
- #define MICROSTEP_MODE_FRAC_128 0x08
  1/128 step mode.
- #define MICROSTEP_MODE_FRAC_256 0x09
  1/256 step mode.

Flags of engine type

Specify motor type. Flags returned by query of engine settings.

See Also

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

  A value that shouldn't be used.
- #define ENGINE_TYPE_DC 0x01

  DC motor.
- #define ENGINE_TYPE_2DC 0x02

  2 DC motors.
- #define ENGINE_TYPE_STEP 0x03

  Step motor.
- #define ENGINE_TYPE_TEST 0x04

  Duty cycle are fixed.
- #define ENGINE_TYPE_BRUSHLESS 0x05

  Brushless motor.

Flags of driver type

Specify driver type. Flags returned by query of engine settings.

See Also

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

- #define DRIVER_TYPE_DISCRETE_FET 0x01

  Driver with discrete FET keys.
- #define DRIVER_TYPE_INTEGRATE 0x02

  Driver with integrated IC.
- #define DRIVER_TYPE_EXTERNAL 0x03

  External driver.

Flags of power settings of stepper motor

Specify power settings. Flags returned by query of power settings.

See Also

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

- #define POWER_REDUCT_ENABLED 0x01

  Current reduction enabled after CurrReductDelay, if this flag is set.
- #define POWER_OFF_ENABLED 0x02

  Power off enabled after PowerOffDelay, if this flag is set.
- #define POWER_SMOOTH_CURRENT 0x04

  Current ramp-up/down is performed smoothly during current_set_time, if this flag is set.

Flags of secure settings

Specify secure settings. Flags returned by query of secure settings.

See Also

secure_settings_t::flags
get_secure_settings
set_secure_settings
secure_settings_t::Flags, get_secure_settings, set_secure_settings

- #define ALARM_ON_DRIVER_OVERHEATING 0x01

  If this flag is set enter Alarm state on driver overheat signal.
- #define LOW_UPWR_PROTECTION 0x02

  If this flag is set turn off motor when voltage is lower than LowUpwrOff.
- #define H_BRIDGE_ALERT 0x04

  If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.
- #define ALARM_ON_BORDERS_SWAP_MISSET 0x08

  If this flag is set enter Alarm state on borders swap misset.
- #define ALARM_FLAGS_STICKING 0x10

  If this flag is set only a STOP command can turn all alarms to 0.
- #define USB_BREAK_RECONNECT 0x20

  If this flag is set USB brake reconnect module will be enable.

Position setting flags

Flags used in setting of position.

See Also

get_position
set_position
set_position_t::PosFlags, set_position

- #define SETPOS_IGNORE_POSITION 0x01

  Will not reload position in steps/microsteps if this flag is set.
- #define SETPOS_IGNORE_ENCODER 0x02

  Will not reload encoder state if this flag is set.

Feedback type.

See Also

set_feedback_settings
get_feedback_settings
feedback_settings_t::FeedbackType, get_feedback_settings, set_feedback_settings

- #define FEEDBACK_ENCODER 0x01

  Feedback by encoder.
- #define FEEDBACK_EMF 0x04

  Feedback by EMF.
- #define FEEDBACK_NONE 0x05

  Feedback is absent.

Describes feedback flags.

See Also

set_feedback_settings
get_feedback_settings
feedback_settings_t::FeedbackFlags, get_feedback_settings, set_feedback_settings

- #define FEEDBACK_ENC_REVERSE 0x01

  Reverse count of encoder.
- #define FEEDBACK_ENC_TYPE_BITS 0xC0

  Bits of the encoder type.

- #define FEEDBACK_ENC_TYPE_AUTO 0x00
  Auto detect encoder type.
- #define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40
  Single ended encoder.
- #define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80
  Differential encoder.

Flags for synchronization input setup

See Also

sync_settings_t::syncin_flags
get_sync_settings
set_sync_settings
sync_in_settings_t::SyncInFlags, get_sync_in_settings, set_sync_in_settings

- #define SYNCIN_ENABLED 0x01
  Synchronization in mode is enabled, if this flag is set.
- #define SYNCIN_INVERT 0x02
  Trigger on falling edge if flag is set, on rising edge otherwise.
- #define SYNCIN_GOTOPOSITION 0x04
  The engine is go to position specified in Position and uPosition, if this flag is set.

Flags of synchronization output

See Also

sync_settings_t::syncout_flags
get_sync_settings
set_sync_settings
sync_out_settings_t::SyncOutFlags, get_sync_out_settings, set_sync_out_settings

- #define SYNCOUT_ENABLED 0x01
  Synchronization out pin follows the synchronization logic, if set.
- #define SYNCOUT_STATE 0x02
  When output state is fixed by negative SYNCOUT_ENABLED flag, the pin state is in accordance with this flag state.
- #define SYNCOUT_INVERT 0x04
  Low level is active, if set, and high level is active otherwise.
- #define SYNCOUT_IN_STEPS 0x08
  Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.
- #define SYNCOUT_ONSTART 0x10
  Generate synchronization pulse when movement starts.
- #define SYNCOUT_ONSTOP 0x20
  Generate synchronization pulse when movement stops.
- #define SYNCOUT_ONPERIOD 0x40
  Generate synchronization pulse every SyncOutPeriod encoder pulses.

External IO setup flags

See Also

extio_settings_t::setup_flags
get_extio_settings
set_extio_settings
extio_settings_t::EXTIOSetupFlags, get_extio_settings, set_extio_settings

- #define EXTIO_SETUP_OUTPUT 0x01
  EXTIO works as output if flag is set, works as input otherwise.
- #define EXTIO_SETUP_INVERT 0x02
  Interpret EXTIO states and fronts inverted if flag is set.

External IO mode flags

See Also

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

  Bits of the behaviour selector when the signal on input goes to the active state.
- #define EXTIO_SETUP_MODE_IN_NOP 0x00

  Do nothing.
- #define EXTIO_SETUP_MODE_IN_STOP 0x01

  Issue STOP command, ceasing the engine movement.
- #define EXTIO_SETUP_MODE_IN_PWOF 0x02

  Issue PWOF command, powering off all engine windings.
- #define EXTIO_SETUP_MODE_IN_MOVR 0x03

  Issue MOVR command with last used settings.
- #define EXTIO_SETUP_MODE_IN_HOME 0x04

  Issue HOME command.
- #define EXTIO_SETUP_MODE_IN_ALARM 0x05

  Set Alarm when the signal goes to the active state.
- #define EXTIO_SETUP_MODE_OUT_BITS 0xF0

  Bits of the output behaviour selection.
- #define EXTIO_SETUP_MODE_OUT_OFF 0x00

  EXTIO pin always set in inactive state.
- #define EXTIO_SETUP_MODE_OUT_ON 0x10

  EXTIO pin always set in active state.
- #define EXTIO_SETUP_MODE_OUT_MOVING 0x20

  EXTIO pin stays active during moving state.
- #define EXTIO_SETUP_MODE_OUT_ALARM 0x30

  EXTIO pin stays active during Alarm state.
- #define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40

  EXTIO pin stays active when windings are powered.
- #define EXTIO_SETUP_MODE_OUT_MOTOR_FOUND 0x50

  EXTIO pin stays active when motor is connected (first winding).

Border flags

Specify types of borders and motor behaviour on borders. May be combined with bitwise OR.

See Also

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

- #define BORDER_IS_ENCODER 0x01

  Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.
- #define BORDER_STOP_LEFT 0x02

  Motor should stop on left border.
- #define BORDER_STOP_RIGHT 0x04

  Motor should stop on right border.
- #define BORDERS_SWAP_MISSET_DETECTION 0x08

  Motor should stop on both borders.

Limit switches flags

Specify electrical behaviour of limit switches like order and pulled positions. May be combined with bitwise OR.

See Also

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

- #define ENDER_SWAP 0x01
  First limit switch on the right side, if set; otherwise on the left side.
- #define ENDER_SW1_ACTIVE_LOW 0x02
  1 - Limit switch connnected to pin SW1 is triggered by a low level on pin.
- #define ENDER_SW2_ACTIVE_LOW 0x04
  1 - Limit switch connnected to pin SW2 is triggered by a low level on pin.

Brake settings flags

Specify behaviour of brake. May be combined with bitwise OR.

See Also

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

- #define BRAKE_ENABLED 0x01
  Brake control is enabled, if this flag is set.
- #define BRAKE_ENG_PWROFF 0x02
  Brake turns off power of step motor, if this flag is set.

Control flags

Specify motor control settings by joystick or buttons. May be combined with bitwise OR.

See Also

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

- #define CONTROL_MODE_BITS 0x03
  Bits to control engine by joystick or buttons.
- #define CONTROL_MODE_OFF 0x00
  Control is disabled.
- #define CONTROL_MODE_JOY 0x01
  Control by joystick.
- #define CONTROL_MODE_LR 0x02
  Control by left/right buttons.
- #define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04
  Pushed left button corresponds to open contact, if this flag is set.
- #define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08
  Pushed right button corresponds to open contact, if this flag is set.

Joystick flags

Control joystick states.

See Also

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

- #define JOY_REVERSE 0x01
  Joystick action is reversed.

Position control flags

Specify settings of position control. May be combined with bitwise OR.

See Also

get_ctp_settings
set_ctp_settings
ctp_settings_t::CTPFlags, get_ctp_settings, set_ctp_settings

- #define CTP_ENABLED 0x01
  Position control is enabled, if flag set.
- #define CTP_BASE 0x02
  Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.
- #define CTP_ALARM_ON_ERROR 0x04
  Set ALARM on mismatch, if flag set.
- #define REV_SENS_INV 0x08
  Sensor is active when it 0 and invert makes active level 1.
- #define CTP_ERROR_CORRECTION 0x10
  Correct errors which appear when slippage if the flag is set.

Home settings flags

Specify behaviour for home command. May be combined with bitwise OR.

See Also

get_home_setting s
set_home_settings
command_home
home_settings_t::HomeFlags, get_home_settings, set_home_settings

- #define HOME_DIR_FIRST 0x001
  Flag defines direction of 1st motion after execution of home command.
- #define HOME_DIR_SECOND 0x002
  Flag defines direction of 2nd motion.
- #define HOME_MV_SEC_EN 0x004
  Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.
- #define HOME_HALF_MV 0x008
  If the flag is set, the stop signals are ignored in start of second movement the first half-turn.
- #define HOME_STOP_FIRST_BITS 0x030
  Bits of the first stop selector.
- #define HOME_STOP_FIRST_REV 0x010
  First motion stops by revolution sensor.
- #define HOME_STOP_FIRST_SYN 0x020
  First motion stops by synchronization input.
- #define HOME_STOP_FIRST_LIM 0x030
  First motion stops by limit switch.
- #define HOME_STOP_SECOND_BITS 0x0C0
  Bits of the second stop selector.
- #define HOME_STOP_SECOND_REV 0x040
  Second motion stops by revolution sensor.
- #define HOME_STOP_SECOND_SYN 0x080
  Second motion stops by synchronization input.
- #define HOME_STOP_SECOND_LIM 0x0C0
  Second motion stops by limit switch.
- #define HOME_USE_FAST 0x100
  Use the fast algorithm of calibration to the home position, if set; otherwise the traditional algorithm.

UART parity flags

See Also

uart_settings_t::UARTSetupFlags, get_uart_settings, set_uart_settings

- #define UART_PARITY_BITS 0x03
  Bits of the parity.
- #define UART_PARITY_BIT_EVEN 0x00
  Parity bit 1, if even.
- #define UART_PARITY_BIT_ODD 0x01
  Parity bit 1, if odd.
- #define UART_PARITY_BIT_SPACE 0x02
  Parity bit always 0.
- #define UART_PARITY_BIT_MARK 0x03
  Parity bit always 1.
- #define UART_PARITY_BIT_USE 0x04
  None parity.
- #define UART_STOP_BIT 0x08
  If set - one stop bit, else two stop bit.

Motor Type flags

See Also

motor_settings_t::MotorType, get_motor_settings, set_motor_settings

- #define MOTOR_TYPE_UNKNOWN 0x00
  Unknown type of engine.
- #define MOTOR_TYPE_STEP 0x01
  Step engine.
- #define MOTOR_TYPE_DC 0x02
  DC engine.
- #define MOTOR_TYPE_BLDC 0x03
  BLDC engine.

Encoder settings flags

See Also

encoder_settings_t::EncoderSettings, get_encoder_settings, set_encoder_settings

- #define ENCSET_DIFFERENTIAL_OUTPUT 0x001
  If flag is set the encoder has differential output, else single ended output.
- #define ENCSET_PUSHPULL_OUTPUT 0x004
  If flag is set the encoder has push-pull output, else open drain output.
- #define ENCSET_INDEXCHANNEL_PRESENT 0x010
  If flag is set the encoder has index channel, else encoder hasn‘t it.
- #define ENCSET_REVOLUTIONSENSOR_PRESENT 0x040
  If flag is set the encoder has revolution sensor, else encoder hasn‘t it.
- #define ENCSET_REVOLUTIONSENSOR_ACTIVE_HIGH 0x100
  If flag is set the revolution sensor active state is high logic state, else active state is low logic state.

Magnetic brake settings flags

See Also

accessories_settings_t::MBSettings, get_accessories_settings, set_accessories_settings

- #define MB_AVAILABLE 0x01
  If flag is set the magnetic brake is available.
- #define MB_POWERED_HOLD 0x02
  If this flag is set the magnetic brake is on when powered.

Temperature sensor settings flags

See Also

accessories_settings_t::LimitSwitchesSettings, get_accessories_settings, set_accessories_settings

- #define TS_TYPE_BITS 0x07

  Bits of the temperature sensor type.
- #define TS_TYPE_UNKNOWN 0x00

  Unknow type of sensor.
- #define TS_TYPE_THERMOCOUPLE 0x01

  Thermocouple.
- #define TS_TYPE_SEMICONDUCTOR 0x02

  The semiconductor temperature sensor.
- #define TS_AVAILABLE 0x08

  If flag is set the temperature sensor is available.
- #define LS_ON_SW1_AVAILABLE 0x01

  If flag is set the limit switch connnected to pin SW1 is available.
- #define LS_ON_SW2_AVAILABLE 0x02

  If flag is set the limit switch connnected to pin SW2 is available.
- #define LS_SW1_ACTIVE_LOW 0x04

  If flag is set the limit switch connnected to pin SW1 is triggered by a low level on pin.
- #define LS_SW2_ACTIVE_LOW 0x08

  If flag is set the limit switch connnected to pin SW2 is triggered by a low level on pin.
- #define LS_SHORTED 0x10

  If flag is set the Limit switches is shorted.

Typedefs

- typedef unsigned long long **ulong_t**
- typedef long long **long_t**
- typedef int device_t

  Type describes device identifier.
- typedef int result_t

  Type specifies result of any operation.
- typedef uint32_t device_enumeration_t

  Type describes device enumeration structure.
- typedef struct calibration_t calibration_t

  Calibration companion structure.
- typedef struct device_network_information_t device_network_information_t

  Device network information structure.

Functions

Controller settings setup

Functions for adjusting engine read/write almost all controller settings.

- result_t XIMC_API set_feedback_settings (device_t id, const feedback_settings_t ∗feedback_settings)

  Feedback settings.
- result_t XIMC_API get_feedback_settings (device_t id, feedback_settings_t ∗feedback_settings)

  Feedback settings.
- result_t XIMC_API set_home_settings (device_t id, const home_settings_t ∗home_settings)

  Set home settings.
- result_t XIMC_API **set_home_settings_calb** (device_t id, const home_settings_calb_t ∗home_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API get_home_settings (device_t id, home_settings_t ∗home_settings)

Read home settings.
- result_t XIMC_API **get_home_settings_calb** (device_t id, home_settings_calb_t ∗home_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API set_move_settings (device_t id, const move_settings_t ∗move_settings)

  Set command setup movement (speed, acceleration, threshold and etc).
- result_t XIMC_API **set_move_settings_calb** (device_t id, const move_settings_calb_t ∗move_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API get_move_settings (device_t id, move_settings_t ∗move_settings)

  Read command setup movement (speed, acceleration, threshold and etc).
- result_t XIMC_API **get_move_settings_calb** (device_t id, move_settings_calb_t ∗move_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API set_engine_settings (device_t id, const engine_settings_t ∗engine_settings)

  Set engine settings.
- result_t XIMC_API **set_engine_settings_calb** (device_t id, const engine_settings_calb_t ∗engine_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API get_engine_settings (device_t id, engine_settings_t ∗engine_settings)

  Read engine settings.
- result_t XIMC_API **get_engine_settings_calb** (device_t id, engine_settings_calb_t ∗engine_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t ∗entype_settings)

  Set engine type and driver type.
- result_t XIMC_API get_entype_settings (device_t id, entype_settings_t ∗entype_settings)

  Return engine type and driver type.
- result_t XIMC_API set_power_settings (device_t id, const power_settings_t ∗power_settings)

  Set settings of step motor power control.
- result_t XIMC_API get_power_settings (device_t id, power_settings_t ∗power_settings)

  Read settings of step motor power control.
- result_t XIMC_API set_secure_settings (device_t id, const secure_settings_t ∗secure_settings)

  Set protection settings.
- result_t XIMC_API get_secure_settings (device_t id, secure_settings_t ∗secure_settings)

  Read protection settings.
- result_t XIMC_API set_edges_settings (device_t id, const edges_settings_t ∗edges_settings)

  Set border and limit switches settings.
- result_t XIMC_API **set_edges_settings_calb** (device_t id, const edges_settings_calb_t ∗edges_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API get_edges_settings (device_t id, edges_settings_t ∗edges_settings)

  Read border and limit switches settings.
- result_t XIMC_API **get_edges_settings_calb** (device_t id, edges_settings_calb_t ∗edges_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API set_pid_settings (device_t id, const pid_settings_t ∗pid_settings)

  Set PID settings.
- result_t XIMC_API get_pid_settings (device_t id, pid_settings_t ∗pid_settings)

  Read PID settings.
- result_t XIMC_API set_sync_in_settings (device_t id, const sync_in_settings_t ∗sync_in_settings)

  Set input synchronization settings.
- 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)
- result_t XIMC_API get_sync_in_settings (device_t id, sync_in_settings_t ∗sync_in_settings)

  Read input synchronization settings.
- 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)
- result_t XIMC_API set_sync_out_settings (device_t id, const sync_out_settings_t ∗sync_out_settings)

  Set output synchronization settings.
- 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)
- result_t XIMC_API get_sync_out_settings (device_t id, sync_out_settings_t ∗sync_out_settings)

  Read output synchronization settings.
- 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)
- result_t XIMC_API set_extio_settings (device_t id, const extio_settings_t ∗extio_settings)

Set EXTIO settings.
- result_t XIMC_API get_extio_settings (device_t id, extio_settings_t ∗extio_settings)

  Read EXTIO settings.
- result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t ∗brake_settings)

  Set settings of brake control.
- result_t XIMC_API get_brake_settings (device_t id, brake_settings_t ∗brake_settings)

  Read settings of brake control.
- result_t XIMC_API set_control_settings (device_t id, const control_settings_t ∗control_settings)

  Set settings of motor control.
- result_t XIMC_API **set_control_settings_calb** (device_t id, const control_settings_calb_t ∗control_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API get_control_settings (device_t id, control_settings_t ∗control_settings)

  Read settings of motor control.
- result_t XIMC_API **get_control_settings_calb** (device_t id, control_settings_calb_t ∗control_settings_calb, const calibration_t ∗calibration)
- result_t XIMC_API set_joystick_settings (device_t id, const joystick_settings_t ∗joystick_settings)

  Set settings of joystick.
- result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t ∗joystick_settings)

  Read settings of joystick.
- result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t ∗ctp_settings)

  Set settings of control position(is only used with stepper motor).
- result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t ∗ctp_settings)

  Read settings of control position(is only used with stepper motor).
- result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t ∗uart_settings)

  Set UART settings.
- result_t XIMC_API get_uart_settings (device_t id, uart_settings_t ∗uart_settings)

  Read UART settings.
- result_t XIMC_API set_calibration_settings (device_t id, const calibration_settings_t ∗calibration_settings)

  Set calibration settings.
- result_t XIMC_API get_calibration_settings (device_t id, calibration_settings_t ∗calibration_settings)

  Read calibration settings.
- result_t XIMC_API set_controller_name (device_t id, const controller_name_t ∗controller_name)

  Write user controller name and flags of setting from FRAM.
- result_t XIMC_API get_controller_name (device_t id, controller_name_t ∗controller_name)

  Read user controller name and flags of setting from FRAM.
- result_t XIMC_API set_nonvolatile_memory (device_t id, const nonvolatile_memory_t ∗nonvolatile_memory)

  Write userdata into FRAM.
- result_t XIMC_API get_nonvolatile_memory (device_t id, nonvolatile_memory_t ∗nonvolatile_memory)

  Read userdata from FRAM.

Group of commands movement control

- result_t XIMC_API command_stop (device_t id)

  Immediately stop the engine, the transition to the STOP, mode key BREAK (winding short-circuited), the regime "retention" is deactivated for DC motors, keeping current in the windings for stepper motors (with Power management settings).
- result_t XIMC_API command_add_sync_in_action (device_t id, const command_add_sync_in_action_t ∗the_command_add_sync_in_action)

  This command adds one element of the FIFO commands that are executed when input clock pulse.
- result_t XIMC_API **command_add_sync_in_action_calb** (device_t id, const command_add_sync_in_action_calb_t ∗the_command_add_sync_in_action_calb, const calibration_t ∗calibration)
- result_t XIMC_API command_power_off (device_t id)

  Immediately power off motor regardless its state.
- result_t XIMC_API command_move (device_t id, int Position, int uPosition)

  Upon receiving the command "move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified to the Position, uPosition.
- result_t XIMC_API **command_move_calb** (device_t id, float Position, const calibration_t ∗calibration)
- result_t XIMC_API command_movr (device_t id, int DeltaPosition, int uDeltaPosition)

Upon receiving the command "movr" engine starts to move with pre-set parameters (speed, acceleration, hold), left or right (depending on the sign of DeltaPosition) by the number of pulses specified in the fields DeltaPosition, uDeltaPosition.

- result_t XIMC_API **command_movr_calb** (device_t id, float DeltaPosition, const calibration_t ∗calibration)
- result_t XIMC_API command_home (device_t id)

  The positive direction is to the right.
- result_t XIMC_API command_left (device_t id)

  Start continous moving to the left.
- result_t XIMC_API command_right (device_t id)

  Start continous moving to the right.
- result_t XIMC_API command_loft (device_t id)

  Upon receiving the command "loft" the engine is shifted from the current point to a distance GENG :: Antiplay, then move to the same point.
- result_t XIMC_API command_sstp (device_t id)

  Soft stop engine.
- result_t XIMC_API get_position (device_t id, get_position_t ∗the_get_position)

  Reads the value position in steps and micro for stepper motor and encoder steps all engines.
- result_t XIMC_API **get_position_calb** (device_t id, get_position_calb_t ∗the_get_position_calb, const calibration_t ∗calibration)
- result_t XIMC_API set_position (device_t id, const set_position_t ∗the_set_position)

  Sets any position value in steps and micro for stepper motor and encoder steps of all engines.
- result_t XIMC_API **set_position_calb** (device_t id, const set_position_calb_t ∗the_set_position_calb, const calibration_t ∗calibration)
- result_t XIMC_API command_zero (device_t id)

  Sets the current position and the position in which the traffic moves by the move command and movr zero for all cases, except for movement to the target position.

Group of commands to save and load settings

- result_t XIMC_API command_save_settings (device_t id)

  Save all settings from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.
- result_t XIMC_API command_read_settings (device_t id)

  Read all settings from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.
- result_t XIMC_API command_save_robust_settings (device_t id)

  Save important settings (calibration coefficients and etc.) from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.
- result_t XIMC_API command_read_robust_settings (device_t id)

  Read important settings (calibration coefficients and etc.) from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.
- result_t XIMC_API command_eesave_settings (device_t id)

  Save settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.
- result_t XIMC_API command_eeread_settings (device_t id)

  Read settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.
- result_t XIMC_API command_start_measurements (device_t id)

  Start measurements and buffering of speed, following error.
- result_t XIMC_API get_measurements (device_t id, measurements_t ∗measurements)

  A command to read the data buffer to build a speed graph and a sequence error.
- result_t XIMC_API get_chart_data (device_t id, chart_data_t ∗chart_data)

  Return device electrical parameters, useful for charts.
- result_t XIMC_API get_serial_number (device_t id, unsigned int ∗SerialNumber)

  Read device serial number.
- result_t XIMC_API get_firmware_version (device_t id, unsigned int ∗Major, unsigned int ∗Minor, unsigned int ∗Release)

  Read controller's firmware version.
- result_t XIMC_API service_command_updf (device_t id)

  Command puts the controller to update the firmware.

Service commands

- result_t XIMC_API set_serial_number (device_t id, const serial_number_t ∗serial_number)
  Write device serial number and hardware version to controller's flash memory.
- result_t XIMC_API get_analog_data (device_t id, analog_data_t ∗analog_data)
  Read analog data structure that contains raw analog data from ADC embedded on board.
- result_t XIMC_API get_debug_read (device_t id, debug_read_t ∗debug_read)
  Read data from firmware for debug purpose.
- result_t XIMC_API set_debug_write (device_t id, const debug_write_t ∗debug_write)
  Write data to firmware for debug purpose.

Group of commands to work with EEPROM

- result_t XIMC_API set_stage_name (device_t id, const stage_name_t ∗stage_name)
  Write user stage name from EEPROM.
- result_t XIMC_API get_stage_name (device_t id, stage_name_t ∗stage_name)
  Read user stage name from EEPROM.
- result_t XIMC_API set_stage_information (device_t id, const stage_information_t ∗stage_information)
  Set stage information to EEPROM.
- result_t XIMC_API get_stage_information (device_t id, stage_information_t ∗stage_information)
  Read stage information from EEPROM.
- result_t XIMC_API set_stage_settings (device_t id, const stage_settings_t ∗stage_settings)
  Set stage settings to EEPROM.
- result_t XIMC_API get_stage_settings (device_t id, stage_settings_t ∗stage_settings)
  Read stage settings from EEPROM.
- result_t XIMC_API set_motor_information (device_t id, const motor_information_t ∗motor_information)
  Set motor information to EEPROM.
- result_t XIMC_API get_motor_information (device_t id, motor_information_t ∗motor_information)
  Read motor information from EEPROM.
- result_t XIMC_API set_motor_settings (device_t id, const motor_settings_t ∗motor_settings)
  Set motor settings to EEPROM.
- result_t XIMC_API get_motor_settings (device_t id, motor_settings_t ∗motor_settings)
  Read motor settings from EEPROM.
- result_t XIMC_API set_encoder_information (device_t id, const encoder_information_t ∗encoder_information)
  Set encoder information to EEPROM.
- result_t XIMC_API get_encoder_information (device_t id, encoder_information_t ∗encoder_information)
  Read encoder information from EEPROM.
- result_t XIMC_API set_encoder_settings (device_t id, const encoder_settings_t ∗encoder_settings)
  Set encoder settings to EEPROM.
- result_t XIMC_API get_encoder_settings (device_t id, encoder_settings_t ∗encoder_settings)
  Read encoder settings from EEPROM.
- result_t XIMC_API set_hallsensor_information (device_t id, const hallsensor_information_t ∗hallsensor_information)
  Set hall sensor information to EEPROM.
- result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t ∗hallsensor_information)
  Read hall sensor information from EEPROM.
- result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t ∗hallsensor_settings)
  Set hall sensor settings to EEPROM.
- result_t XIMC_API get_hallsensor_settings (device_t id, hallsensor_settings_t ∗hallsensor_settings)
  Read hall sensor settings from EEPROM.
- result_t XIMC_API set_gear_information (device_t id, const gear_information_t ∗gear_information)
  Set gear information to EEPROM.
- result_t XIMC_API get_gear_information (device_t id, gear_information_t ∗gear_information)
  Read gear information from EEPROM.
- result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t ∗gear_settings)
  Set gear settings to EEPROM.
- result_t XIMC_API get_gear_settings (device_t id, gear_settings_t ∗gear_settings)

  Read gear settings from EEPROM.
- result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t ∗accessories_settings)

  Set additional accessories information to EEPROM.
- result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t ∗accessories_settings)

  Read additional accessories information from EEPROM.
- result_t XIMC_API get_bootloader_version (device_t id, unsigned int ∗Major, unsigned int ∗Minor, unsigned int ∗Release)

  Read controller's firmware version.
- result_t XIMC_API get_init_random (device_t id, init_random_t ∗init_random)

  Read random number from controller.
- result_t XIMC_API get_globally_unique_identifier (device_t id, globally_unique_identifier_t ∗globally_unique_identifier)

  This value is unique to each individual die but is not a random value.
- result_t XIMC_API command_change_motor (device_t id, const command_change_motor_t ∗the_command_change_motor)

  Change motor - command for switching output relay.
- result_t XIMC_API goto_firmware (device_t id, uint8_t ∗ret)

  Reboot to firmware.
- result_t XIMC_API has_firmware (const char ∗uri, uint8_t ∗ret)

  Check for firmware on device.
- result_t XIMC_API command_update_firmware (const char ∗uri, const uint8_t ∗data, uint32_t data_size)

  Update firmware.
- result_t XIMC_API write_key (const char ∗uri, uint8_t ∗key)

  Write controller key.
- result_t XIMC_API command_reset (device_t id)

  Reset controller.
- result_t XIMC_API command_clear_fram (device_t id)

  Clear controller FRAM.

Boards and drivers control

Functions for searching and opening/closing devices

- typedef char ∗ pchar

  Nevermind.
- typedef void(XIMC_CALLCONV ∗ logging_callback_t)(int loglevel, const wchar_t ∗message, void ∗user_data)

  Logging callback prototype.
- device_t XIMC_API open_device (const char ∗uri)

  Open a device with OS uri uri and return identifier of the device which can be used in calls.
- result_t XIMC_API close_device (device_t ∗id)

  Close specified device.
- result_t XIMC_API probe_device (const char ∗uri)

  Check if a device with OS uri uri is XIMC device.
- result_t XIMC_API set_bindy_key (const char ∗keyfilepath)

  Set network encryption layer (bindy) key.
- device_enumeration_t XIMC_API enumerate_devices (int enumerate_flags, const char ∗hints)

  Enumerate all devices that looks like valid.
- result_t XIMC_API free_enumerate_devices (device_enumeration_t device_enumeration)

  Free memory returned by enumerate_devices.
- int XIMC_API get_device_count (device_enumeration_t device_enumeration)

  Get device count.
- pchar XIMC_API get_device_name (device_enumeration_t device_enumeration, int device_index)

  Get device name from the device enumeration.

- result_t XIMC_API get_enumerate_device_serial (device_enumeration_t device_enumeration, int device_index, uint32_t ∗serial)

  Get device serial number from the device enumeration.
- result_t XIMC_API get_enumerate_device_information (device_enumeration_t device_enumeration, int device_index, device_information_t ∗device_information)

  Get device information from the device enumeration.
- result_t XIMC_API get_enumerate_device_controller_name (device_enumeration_t device_enumeration, int device_index, controller_name_t ∗controller_name)

  Get controller name from the device enumeration.
- result_t XIMC_API get_enumerate_device_stage_name (device_enumeration_t device_enumeration, int device_index, stage_name_t ∗stage_name)

  Get stage name from the device enumeration.
- result_t XIMC_API get_enumerate_device_network_information (device_enumeration_t device_enumeration, int device_index, device_network_information_t ∗device_network_information)

  Get device network information from the device enumeration.
- result_t XIMC_API reset_locks ()

  Reset library locks in a case of deadlock.
- result_t XIMC_API ximc_fix_usbser_sys (const char ∗device_uri)

  Fix for errors in Windows USB driver stack.
- void XIMC_API msec_sleep (unsigned int msec)

  Sleeps for a specified amount of time.
- void XIMC_API ximc_version (char ∗version)

  Returns a library version.
- void XIMC_API logging_callback_stderr_wide (int loglevel, const wchar_t ∗message, void ∗user_data)

  Simple callback for logging to stderr in wide chars.
- void XIMC_API logging_callback_stderr_narrow (int loglevel, const wchar_t ∗message, void ∗user_data)

  Simple callback for logging to stderr in narrow (single byte) chars.
- void XIMC_API set_logging_callback (logging_callback_t logging_callback, void ∗user_data)

  Sets a logging callback.
- result_t XIMC_API get_status (device_t id, status_t ∗status)

  Return device state.
- result_t XIMC_API get_status_calb (device_t id, status_calb_t ∗status, const calibration_t ∗calibration)

  Calibrated device state.
- result_t XIMC_API get_device_information (device_t id, device_information_t ∗device_information)

  Return device information.
- result_t XIMC_API command_wait_for_stop (device_t id, uint32_t refresh_interval_ms)

  Wait for stop.
- result_t XIMC_API command_homezero (device_t id)

  Make home command, wait until it is finished and make zero command.

6.1.1 Detailed Description

Header file for libximc library.

6.1.2 Macro Definition Documentation

6.1.2.1 #define ALARM_ON_DRIVER_OVERHEATING 0x01

If this flag is set enter Alarm state on driver overheat signal.

6.1.2.2 #define BORDER_IS_ENCODER 0x01

Borders are fixed by predetermined encoder values, if set; borders position on limit switches, if not set.

6.1.2.3 #define BORDER_STOP_LEFT 0x02

Motor should stop on left border.

6.1.2.4 #define BORDER_STOP_RIGHT 0x04

Motor should stop on right border.

6.1.2.5 #define BORDERS_SWAP_MISSET_DETECTION 0x08

Motor should stop on both borders.
Need to save motor then wrong border settings is set

6.1.2.6 #define BRAKE_ENABLED 0x01

Brake control is enabled, if this flag is set.

6.1.2.7 #define BRAKE_ENG_PWROFF 0x02

Brake turns off power of step motor, if this flag is set.

6.1.2.8 #define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04

Pushed left button corresponds to open contact, if this flag is set.

6.1.2.9 #define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08

Pushed right button corresponds to open contact, if this flag is set.

6.1.2.10 #define CONTROL_MODE_BITS 0x03

Bits to control engine by joystick or buttons.

6.1.2.11 #define CONTROL_MODE_JOY 0x01

Control by joystick.

6.1.2.12 #define CONTROL_MODE_LR 0x02

Control by left/right buttons.

6.1.2.13 #define CONTROL_MODE_OFF 0x00

Control is disabled.

6.1.2.14 #define CTP_ALARM_ON_ERROR 0x04

Set ALARM on mismatch, if flag set.

6.1.2.15 #define CTP_BASE 0x02

Position control is based on revolution sensor, if this flag is set; otherwise it is based on encoder.

6.1.2.16 #define CTP_ENABLED 0x01

Position control is enabled, if flag set.

6.1.2.17 #define CTP_ERROR_CORRECTION 0x10

Correct errors which appear when slippage if the flag is set.

It works only with the encoder. Incompatible with flag CTP_ALARM_ON_ERROR.

6.1.2.18 #define DRIVER_TYPE_DISCRETE_FET 0x01

Driver with discrete FET keys.

Default option.

6.1.2.19 #define DRIVER_TYPE_EXTERNAL 0x03

External driver.

6.1.2.20 #define DRIVER_TYPE_INTEGRATE 0x02

Driver with integrated IC.

6.1.2.21 #define EEPROM_PRECEDENCE 0x01

If the flag is set settings from external EEPROM override controller settings.

6.1.2.22 #define ENC_STATE_ABSENT 0x00

Encoder is absent.

6.1.2.23 #define ENC_STATE_MALFUNC 0x02

Encoder is connected and malfunctioning.

6.1.2.24 #define ENC_STATE_OK 0x04

Encoder is connected and working properly.

6.1.2.25 #define ENC_STATE_REVERS 0x03

Encoder is connected and operational but counts in other direction.

6.1.2.26 #define ENC_STATE_UNKNOWN 0x01

Encoder state is unknown.

6.1.2.27 #define ENDER_SW1_ACTIVE_LOW 0x02

1 - Limit switch connnected to pin SW1 is triggered by a low level on pin.

6.1.2.28 #define ENDER_SW2_ACTIVE_LOW 0x04

1 - Limit switch connnected to pin SW2 is triggered by a low level on pin.

6.1.2.29 #define ENDER_SWAP 0x01

First limit switch on the right side, if set; otherwise on the left side.

6.1.2.30 #define ENGINE_ACCEL_ON 0x10

Acceleration enable flag.

If it set, motion begins with acceleration and ends with deceleration.

6.1.2.31 #define ENGINE_ANTIPLAY 0x08

Play compensation flag.

If it set, engine makes backlash (play) compensation procedure and reach the predetermined position accurately on low speed.

6.1.2.32 #define ENGINE_CURRENT_AS_RMS 0x02

Engine current meaning flag.

If the flag is unset, then engine current value is interpreted as maximum amplitude value. If the flag is set, then engine current value is interpreted as root mean square current value (for stepper) or as the current value calculated from the maximum heat dissipation (bldc).

6.1.2.33 #define ENGINE_LIMIT_CURR 0x40

Maximum motor current limit enable flag(is only used with DC motor).

6.1.2.34 #define ENGINE_LIMIT_RPM 0x80

Maximum motor speed limit enable flag.

6.1.2.35 #define ENGINE_LIMIT_VOLT 0x20

Maximum motor voltage limit enable flag(is only used with DC motor).

6.1.2.36 #define ENGINE_MAX_SPEED 0x04

Max speed flag.

If it is set, engine uses maximum speed achievable with the present engine settings as nominal speed.

6.1.2.37 #define ENGINE_REVERSE 0x01

Reverse flag.

It determines motor shaft rotation direction that corresponds to feedback counts increasing. If not set (default), motor shaft rotation direction under positive voltage corresponds to feedback counts increasing and vice versa. Change it if you see that positive directions on motor and feedback are opposite.

6.1.2.38 #define ENGINE_TYPE_2DC 0x02

2 DC motors.

6.1.2.39 #define ENGINE_TYPE_BRUSHLESS 0x05

Brushless motor.

6.1.2.40 #define ENGINE_TYPE_DC 0x01

DC motor.

6.1.2.41 #define ENGINE_TYPE_NONE 0x00

A value that shouldn't be used.

6.1.2.42 #define ENGINE_TYPE_STEP 0x03

Step motor.

6.1.2.43 #define ENGINE_TYPE_TEST 0x04

Duty cycle are fixed.

Used only manufacturer.

6.1.2.44 #define ENUMERATE_PROBE 0x01

Check if a device with OS name name is XIMC device.

Be carefuly with this flag because it sends some data to the device.

6.1.2.45 #define EXTIO_SETUP_INVERT 0x02

Interpret EXTIO states and fronts inverted if flag is set.

Falling front as input event and low logic level as active state.

6.1.2.46 #define EXTIO_SETUP_MODE_IN_ALARM 0x05

Set Alarm when the signal goes to the active state.

6.1.2.47 #define EXTIO_SETUP_MODE_IN_BITS 0x0F

Bits of the behaviour selector when the signal on input goes to the active state.

6.1.2.48 #define EXTIO_SETUP_MODE_IN_HOME 0x04

Issue HOME command.

6.1.2.49 #define EXTIO_SETUP_MODE_IN_MOVR 0x03

Issue MOVR command with last used settings.

6.1.2.50 #define EXTIO_SETUP_MODE_IN_NOP 0x00

Do nothing.

6.1.2.51 #define EXTIO_SETUP_MODE_IN_PWOF 0x02

Issue PWOF command, powering off all engine windings.

6.1.2.52 #define EXTIO_SETUP_MODE_IN_STOP 0x01

Issue STOP command, ceasing the engine movement.

6.1.2.53 #define EXTIO_SETUP_MODE_OUT_ALARM 0x30

EXTIO pin stays active during Alarm state.

6.1.2.54 #define EXTIO_SETUP_MODE_OUT_BITS 0xF0

Bits of the output behaviour selection.

6.1.2.55 #define EXTIO_SETUP_MODE_OUT_MOTOR_FOUND 0x50

EXTIO pin stays active when motor is connected (first winding).

6.1.2.56 #define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40

EXTIO pin stays active when windings are powered.

6.1.2.57 #define EXTIO_SETUP_MODE_OUT_MOVING 0x20

EXTIO pin stays active during moving state.

6.1.2.58 #define EXTIO_SETUP_MODE_OUT_OFF 0x00

EXTIO pin always set in inactive state.

6.1.2.59 #define EXTIO_SETUP_MODE_OUT_ON 0x10

EXTIO pin always set in active state.

6.1.2.60 #define EXTIO_SETUP_OUTPUT 0x01

EXTIO works as output if flag is set, works as input otherwise.

6.1.2.61 #define FEEDBACK_EMF 0x04

Feedback by EMF.

6.1.2.62 #define FEEDBACK_ENC_REVERSE 0x01

Reverse count of encoder.

6.1.2.63 #define FEEDBACK_ENC_TYPE_AUTO 0x00

Auto detect encoder type.

6.1.2.64 #define FEEDBACK_ENC_TYPE_BITS 0xC0

Bits of the encoder type.

6.1.2.65 #define FEEDBACK_ENC_TYPE_DIFFERENTIAL 0x80

Differential encoder.

6.1.2.66 #define FEEDBACK_ENC_TYPE_SINGLE_ENDED 0x40

Single ended encoder.

6.1.2.67 #define FEEDBACK_ENCODER 0x01

Feedback by encoder.

6.1.2.68 #define FEEDBACK_NONE 0x05

Feedback is absent.

6.1.2.69 #define H_BRIDGE_ALERT 0x04

If this flag is set then turn off the power unit with a signal problem in one of the transistor bridge.

6.1.2.70 #define HOME_DIR_FIRST 0x001

Flag defines direction of 1st motion after execution of home command.

Direction is right, if set; otherwise left.

6.1.2.71 #define HOME_DIR_SECOND 0x002

Flag defines direction of 2nd motion.

Direction is right, if set; otherwise left.

6.1.2.72 #define HOME_HALF_MV 0x008

If the flag is set, the stop signals are ignored in start of second movement the first half-turn.

6.1.2.73 #define HOME_MV_SEC_EN 0x004

Use the second phase of calibration to the home position, if set; otherwise the second phase is skipped.

6.1.2.74 #define HOME_STOP_FIRST_BITS 0x030

Bits of the first stop selector.

6.1.2.75 #define HOME_STOP_FIRST_LIM 0x030

First motion stops by limit switch.

6.1.2.76 #define HOME_STOP_FIRST_REV 0x010

First motion stops by revolution sensor.

6.1.2.77 #define HOME_STOP_FIRST_SYN 0x020

First motion stops by synchronization input.

6.1.2.78 #define HOME_STOP_SECOND_BITS 0x0C0

Bits of the second stop selector.

6.1.2.79 #define HOME_STOP_SECOND_LIM 0x0C0

Second motion stops by limit switch.

6.1.2.80 #define HOME_STOP_SECOND_REV 0x040

Second motion stops by revolution sensor.

6.1.2.81 #define HOME_STOP_SECOND_SYN 0x080

Second motion stops by synchronization input.

6.1.2.82 #define HOME_USE_FAST 0x100

Use the fast algorithm of calibration to the home position, if set; otherwise the traditional algorithm.

6.1.2.83 #define JOY_REVERSE 0x01

Joystick action is reversed.

Joystick deviation to the upper values correspond to negative speeds and vice versa.

6.1.2.84 #define LOW_UPWR_PROTECTION 0x02

If this flag is set turn off motor when voltage is lower than LowUpwrOff.

6.1.2.85 #define MICROSTEP_MODE_FRAC_128 0x08

1/128 step mode.

6.1.2.86 #define MICROSTEP_MODE_FRAC_16 0x05

1/16 step mode.

6.1.2.87 #define MICROSTEP_MODE_FRAC_2 0x02

1/2 step mode.

6.1.2.88 #define MICROSTEP_MODE_FRAC_256 0x09

1/256 step mode.

6.1.2.89 #define MICROSTEP_MODE_FRAC_32 0x06

1/32 step mode.

6.1.2.90 #define MICROSTEP_MODE_FRAC_4 0x03

1/4 step mode.

6.1.2.91 #define MICROSTEP_MODE_FRAC_64 0x07

1/64 step mode.

6.1.2.92 #define MICROSTEP_MODE_FRAC_8 0x04

1/8 step mode.

6.1.2.93 #define MICROSTEP_MODE_FULL 0x01

Full step mode.

6.1.2.94 #define MOVE_STATE_ANTIPLAY 0x04

Motor is playing compensation, if flag set.

6.1.2.95 #define MOVE_STATE_MOVING 0x01

This flag indicates that controller is trying to move the motor.

Don't use this flag for waiting of completion of the movement command. Use MVCMD_RUNNING flag from the MvCmdSts field instead.

6.1.2.96 #define MOVE_STATE_TARGET_SPEED 0x02

Target speed is reached, if flag set.

6.1.2.97 #define MVCMD_ERROR 0x40

Finish state (1 - move command have finished with an error, 0 - move command have finished correctly).

This flags is actual when MVCMD_RUNNING signals movement finish.

6.1.2.98 #define MVCMD_HOME 0x06

Command home.

6.1.2.99 #define MVCMD_LEFT 0x03

Command left.

6.1.2.100 #define MVCMD_LOFT 0x07

Command loft.

6.1.2.101 #define MVCMD_MOVE 0x01

Command move.

6.1.2.102 #define MVCMD_MOVR 0x02

Command movr.

6.1.2.103 #define MVCMD_NAME_BITS 0x3F

Move command bit mask.

6.1.2.104 #define MVCMD_RIGHT 0x04

Command rigt.

6.1.2.105 #define MVCMD_RUNNING 0x80

Move command state (0 - move command have finished, 1 - move command is being executed).

6.1.2.106 #define MVCMD_SSTP 0x08

Command soft stop.

6.1.2.107 #define MVCMD_STOP 0x05

Command stop.

6.1.2.108 #define MVCMD_UKNWN 0x00

Unknown command.

6.1.2.109 #define POWER_OFF_ENABLED 0x02

Power off enabled after PowerOffDelay, if this flag is set.

6.1.2.110 #define POWER_REDUCT_ENABLED 0x01

Current reduction enabled after CurrReductDelay, if this flag is set.

6.1.2.111 #define POWER_SMOOTH_CURRENT 0x04

Current ramp-up/down is performed smoothly during current_set_time, if this flag is set.

6.1.2.112 #define PWR_STATE_MAX 0x05

Motor windings are powered by maximum current driver can provide at this voltage.

6.1.2.113 #define PWR_STATE_NORM 0x03

Motor windings are powered by nominal current.

6.1.2.114 #define PWR_STATE_OFF 0x01

Motor windings are disconnected from the driver.

6.1.2.115 #define PWR_STATE_REDUCT 0x04

Motor windings are powered by reduced current to lower power consumption.

6.1.2.116 #define PWR_STATE_UNKNOWN 0x00

Unknown state, should never happen.

6.1.2.117 #define REV_SENS_INV 0x08

Sensor is active when it 0 and invert makes active level 1.

That is, if you do not invert, it is normal logic - 0 is the activation.

6.1.2.118 #define SETPOS_IGNORE_ENCODER 0x02

Will not reload encoder state if this flag is set.

6.1.2.119 #define SETPOS_IGNORE_POSITION 0x01

Will not reload position in steps/microsteps if this flag is set.

6.1.2.120 #define STATE_ALARM 0x000040

Controller is in alarm state indicating that something dangerous had happened.

Most commands are ignored in this state. To reset the flag a STOP command must be issued.

6.1.2.121 #define STATE_BORDERS_SWAP_MISSET 0x008000

Engine stuck at the wrong edge.

6.1.2.122 #define STATE_BRAKE 0x0200

State of Brake pin.

6.1.2.123 #define STATE_BUTTON_LEFT 0x0008

Button ”left” state (1 if pressed).

6.1.2.124 #define STATE_BUTTON_RIGHT 0x0004

Button ”right” state (1 if pressed).

6.1.2.125 #define STATE_CONTR 0x00003F

Flags of controller states.

6.1.2.126 #define STATE_CONTROLLER_OVERHEAT 0x000200

Controller overheat.

6.1.2.127 #define STATE_CTP_ERROR 0x000080

Control position error(is only used with stepper motor).

6.1.2.128 #define STATE_CURRENT_MOTOR0 0x000000

Motor 0.

6.1.2.129 #define STATE_CURRENT_MOTOR1 0x040000

Motor 1.

6.1.2.130 #define STATE_CURRENT_MOTOR2 0x080000

Motor 2.

6.1.2.131 #define STATE_CURRENT_MOTOR3 0x0C0000

Motor 3.

6.1.2.132 #define STATE_CURRENT_MOTOR_BITS 0x0C0000

Bits indicating the current operating motor on boards with multiple outputs for engine mounting.

6.1.2.133 #define STATE_DIG_SIGNAL 0xFFFF

Flags of digital signals.

6.1.2.134 #define STATE_EEPROM_CONNECTED 0x000010

EEPROM with settings is connected.

6.1.2.135 #define STATE_ENC_A 0x2000

State of encoder A pin.

6.1.2.136 #define STATE_ENC_B 0x4000

State of encoder B pin.

6.1.2.137 #define STATE_ERRC 0x000001

Command error encountered.

6.1.2.138 #define STATE_ERRD 0x000002

Data integrity error encountered.

6.1.2.139 #define STATE_ERRV 0x000004

Value error encountered.

6.1.2.140 #define STATE_GPIO_LEVEL 0x0020

State of external GPIO pin.

6.1.2.141 #define STATE_GPIO_PINOUT 0x0010

External GPIO works as Out, if flag set; otherwise works as In.

6.1.2.142 #define STATE_LEFT_EDGE 0x0002

Engine stuck at the left edge.

6.1.2.143 #define STATE_LOW_USB_VOLTAGE 0x002000

USB voltage is insufficient for normal operation.

6.1.2.144 #define STATE_OVERLOAD_POWER_CURRENT 0x000800

Power current exceeds safe limit.

6.1.2.145 #define STATE_OVERLOAD_POWER_VOLTAGE 0x000400

Power voltage exceeds safe limit.

6.1.2.146 #define STATE_OVERLOAD_USB_CURRENT 0x004000

USB current exceeds safe limit.

6.1.2.147 #define STATE_OVERLOAD_USB_VOLTAGE 0x001000

USB voltage exceeds safe limit.

6.1.2.148 #define STATE_POWER_OVERHEAT 0x000100

Power driver overheat.

6.1.2.149 #define STATE_REV_SENSOR 0x0400

State of Revolution sensor pin.

6.1.2.150 #define STATE_RIGHT_EDGE 0x0001

Engine stuck at the right edge.

6.1.2.151 #define STATE_SECUR 0x73FFC0

Flags of security.

6.1.2.152 #define STATE_SYNC_INPUT 0x0800

State of Sync input pin.

6.1.2.153 #define STATE_SYNC_OUTPUT 0x1000

State of Sync output pin.

6.1.2.154 #define SYNCIN_ENABLED 0x01

Synchronization in mode is enabled, if this flag is set.

6.1.2.155 #define SYNCIN_GOTOPOSITION 0x04

The engine is go to position specified in Position and uPosition, if this flag is set.

And it is shift on the Position and uPosition, if this flag is unset

6.1.2.156 #define SYNCIN_INVERT 0x02

Trigger on falling edge if flag is set, on rising edge otherwise.

6.1.2.157 #define SYNCOUT_ENABLED 0x01

Synchronization out pin follows the synchronization logic, if set.

It governed by SYNCOUT_STATE flag otherwise.

6.1.2.158 #define SYNCOUT_IN_STEPS 0x08

Use motor steps/encoder pulses instead of milliseconds for output pulse generation if the flag is set.

6.1.2.159 #define SYNCOUT_INVERT 0x04

Low level is active, if set, and high level is active otherwise.

6.1.2.160 #define SYNCOUT_ONPERIOD 0x40

Generate synchronization pulse every SyncOutPeriod encoder pulses.

6.1.2.161 #define SYNCOUT_ONSTART 0x10

Generate synchronization pulse when movement starts.

6.1.2.162 #define SYNCOUT_ONSTOP 0x20

Generate synchronization pulse when movement stops.

6.1.2.163 #define SYNCOUT_STATE 0x02

When output state is fixed by negative SYNCOUT_ENABLED flag, the pin state is in accordance with this flag state.

6.1.2.164 #define UART_PARITY_BITS 0x03

Bits of the parity.

6.1.2.165 #define WIND_A_STATE_ABSENT 0x00

Winding A is disconnected.

6.1.2.166 #define WIND_A_STATE_MALFUNC 0x02

Winding A is short-circuited.

6.1.2.167 #define WIND_A_STATE_OK 0x03

Winding A is connected and working properly.

6.1.2.168 #define WIND_A_STATE_UNKNOWN 0x01

Winding A state is unknown.

6.1.2.169 #define WIND_B_STATE_ABSENT 0x00

Winding B is disconnected.

6.1.2.170 #define WIND_B_STATE_MALFUNC 0x20

Winding B is short-circuited.

6.1.2.171 #define WIND_B_STATE_OK 0x30

Winding B is connected and working properly.

6.1.2.172 #define WIND_B_STATE_UNKNOWN 0x10

Winding B state is unknown.

6.1.2.173 #define XIMC_API

Library import macro Macros allows to automatically import function from shared library.

It automatically expands to dllimport on msvc when including header file

6.1.3 Typedef Documentation

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

Logging callback prototype.

Parameters

loglevel	a loglevel
message	a message

6.1.4 Function Documentation

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

Close specified device.

Parameters

id	an identifier of device

6.1.4.2 **result_t XIMC_API** command_add_sync_in_action (**device_t** id, const **command_add_sync_in_action_t** * the_command_add_sync_in_action)

This command adds one element of the FIFO commands that are executed when input clock pulse.

Each pulse synchronization or perform that action, which is described in SSNI, if the buffer is empty, or the oldest loaded into the buffer action to temporarily replace the speed and coordinate in SSNI. In the latter case this action is erased from the buffer. The number of remaining empty buffer elements can be found in the structure of GETS.

Parameters

id	an identifier of device

6.1.4.3 **result_t XIMC_API** command_change_motor (**device_t** id, const **command_change_motor_t** * the_command_change_motor)

Change motor - command for switching output relay.

Parameters

id	an identifier of device

6.1.4.4 **result_t XIMC_API** command_clear_fram (**device_t** id)

Clear controller FRAM.

Can be used by manufacturer only

Parameters

id	an identifier of device

6.1.4.5 **result_t XIMC_API** command_eeread_settings (**device_t** id)

Read settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.

Parameters

id	an identifier of device

6.1.4.6 **result_t XIMC_API** command_eesave_settings (**device_t** id)

Save settings from controller's RAM to stage's EEPROM memory, which spontaneity connected to stage and it isn't change without it mechanical reconstruction.

Can be used by manufacturer only.

Parameters

id	an identifier of device

6.1.4.7 **result_t XIMC_API** command_home (**device_t** id)

The positive direction is to the right.

A value of zero reverses the direction of the direction of the flag, the set speed. Restriction imposed by the trailer, act the same, except that the limit switch contact does not stop. Limit the maximum speed, acceleration and deceleration function. 1) moves the motor according to the speed FastHome, uFastHome and flag HOME_DIR_FAST until limit switch, if the flag is set HOME_STOP_ENDS, until the signal from the input synchronization if the flag HOME_STOP_SYNC (as accurately as possible is important to catch the moment of operation limit switch) or until the signal is received from the speed sensor, if the flag HOME_STOP_REV_SN 2) then moves according to the speed SlowHome, uSlowHome and flag HOME_DIR_SLOW until signal from the clock input, if the flag HOME_MV_SEC. If the flag HOME_MV_SEC reset skip this paragraph. 3) then move the motor according to the speed FastHome, uFastHome and flag HOME_DIR_SLOW a distance HomeDelta, uHomeDelta. description of flags and variable see in description for commands GHOM/SHOM

Parameters

id	an identifier of device

See Also

home_settings_t
get_home_settings
set_home_settings

6.1.4.8 **result_t XIMC_API** command_homezero (**device_t** id)

Make home command, wait until it is finished and make zero command.

This is a convinient way to calibrate zero position.

Parameters

	id	an identifier of device
out	*ret*	RESULT_OK if controller has finished home & zero correctly or result of first controller query that returned anything other than RESULT_OK.

6.1.4.9 **result_t XIMC_API** command_left (**device_t** id)

Start continous moving to the left.

Parameters

id	an identifier of device

6.1.4.10 **result_t XIMC_API** command_loft (**device_t** id)

Upon receiving the command "loft" the engine is shifted from the current point to a distance GENG :: Antiplay, then move to the same point.

Parameters

id	an identifier of device

6.1.4.11 **result_t XIMC_API** command_move (**device_t** id, int Position, int uPosition)

Upon receiving the command "move" the engine starts to move with pre-set parameters (speed, acceleration, retention), to the point specified to the Position, uPosition.

For stepper motor uPosition sets the microstep, for DC motor this field is not used.

Parameters

Position	position to move.
uPosition	part of the position to move, microsteps. Range: -255..255.
id	an identifier of device

6.1.4.12 **result_t XIMC_API** command_movr (**device_t** id, int DeltaPosition, int uDeltaPosition)

Upon receiving the command "movr" engine starts to move with pre-set parameters (speed, acceleration, hold), left or right (depending on the sign of DeltaPosition) by the number of pulses specified in the fields DeltaPosition, uDeltaPosition.

For stepper motor uDeltaPosition sets the microstep, for DC motor this field is not used.

Parameters

DeltaPosition	shift from initial position.
uDeltaPosition	part of the offset shift, microsteps. Range: -255..255.
id	an identifier of device

6.1.4.13 **result_t XIMC_API** command_power_off (**device_t** id)

Immediately power off motor regardless its state.

Shouldn't be used during motion as the motor could be power on again automatically to continue movement. The command is designed for manual motor power off. When automatic power off after stop is required, use power management system.

Parameters

id	an identifier of device

See Also

get_power_settings
set_power_settings

6.1.4.14 **result_t XIMC_API** command_read_robust_settings (**device_t** id)

Read important settings (calibration coefficients and etc.) from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.

Parameters

id	an identifier of device

6.1.4.15 **result_t XIMC_API** command_read_settings (**device_t** id)

Read all settings from controller's flash memory to controller's RAM, replacing previous data in controller's RAM.

Parameters

id	an identifier of device

6.1.4.16 **result_t XIMC_API** command_reset (**device_t** id)

Reset controller.

Can be used by manufacturer only

Parameters

id	an identifier of device

6.1.4.17 **result_t XIMC_API** command_right (**device_t** id)

Start continous moving to the right.

Parameters

id	an identifier of device

6.1.4.18 **result_t XIMC_API** command_save_robust_settings (**device_t** id)

Save important settings (calibration coefficients and etc.) from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.

Parameters

id	an identifier of device

6.1.4.19 **result_t XIMC_API** command_save_settings (**device_t** id)

Save all settings from controller's RAM to controller's flash memory, replacing previous data in controller's flash memory.

Parameters

id	an identifier of device

6.1.4.20 **result_t XIMC_API** command_sstp (**device_t** id)

Soft stop engine.

The motor stops with deceleration speed.

Parameters

id	an identifier of device

6.1.4.21 **result_t XIMC_API** command_start_measurements (**device_t** id)

Start measurements and buffering of speed, following error.

Parameters

id	an identifier of device

6.1.4.22 **result_t XIMC_API** command_stop (**device_t** id)

Immediately stop the engine, the transition to the STOP, mode key BREAK (winding short-circuited), the regime ”retention” is deactivated for DC motors, keeping current in the windings for stepper motors (with Power management settings).

Parameters

id	an identifier of device

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

Update firmware.

Service command

Parameters

uri	a uri of device
data	firmware byte stream
data_size	size of byte stream

6.1.4.24 **result_t XIMC_API** command_wait_for_stop (**device_t** id, uint32_t refresh_interval_ms)

Wait for stop.

Parameters

	id	an identifier of device
	refresh_interval_ms	Status refresh interval. The function waits this number of milliseconds between get_status requests to the controller. Recommended value of this parameter is 10 ms. Use values of less than 3 ms only when necessary - small refresh interval values do not significantly increase response time of the function, but they create substantially more traffic in controller-computer data channel.
`out`	*ret*	RESULT_OK if controller has stopped and result of the first get_status command which returned anything other than RESULT_OK otherwise.

6.1.4.25 **result_t XIMC_API** command_zero (**device_t** id)

Sets the current position and the position in which the traffic moves by the move command and movr zero for all cases, except for movement to the target position.

In the latter case, set the zero current position and the target position counted so that the absolute position of the destination is the same. That is, if we were at 400 and moved to 500, then the command Zero makes the current position of 0, and the position of the destination - 100. Does not change the mode of movement that is if the motion is carried, it continues, and if the engine is in the ”hold”, the type of retention remains.

Parameters

id	an identifier of device

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

Enumerate all devices that looks like valid.

Parameters

`in`	*enumerate_flags*	enumerate devices flags
`in`	*hints*	extended information hints is a string of form ”key=value\nkey2=value2”. Unrecognized key-value pairs are ignored. Key list: addr - used together with ENUMERATE_NETWORK flag. Non-null value is a remote host name or a comma-separated list of host names which contain the devices to be found, absent value means broadcast discovery. adapter_addr - used together with ENUMERATE_NETWORK flag. Non-null value is a IP address of network adapter. Remote ximc device must be on the same local network as the adapter. To enumerate network devices you must call set_bindy_key first.

6.1.4.27 **result_t XIMC_API** free_enumerate_devices (**device_enumeration_t** device_enumeration)

Free memory returned by *enumerate_devices*.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data

6.1.4.28 **result_t XIMC_API** get_accessories_settings (**device_t** id, **accessories_settings_t** ∗ accessories_settings)

Read additional accessories information from EEPROM.

Parameters

	id	an identifier of device
`out`	*accessories_settings*	structure contains information about additional accessories

6.1.4.29 **result_t XIMC_API** get_analog_data (**device_t** id, **analog_data_t** ∗ analog_data)

Read analog data structure that contains raw analog data from ADC embedded on board.

This function used for device testing and deep recalibraton by manufacturer only.

Parameters

	id	an identifier of device
out	*analog_data*	analog data coefficients

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

Read controller's firmware version.

Parameters

	id	an identifier of device
out	*Major*	major version
out	*Minor*	minor version
out	*Release*	release version

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

Read settings of brake control.

Parameters

	id	an identifier of device
out	*brake_settings*	structure contains settings of brake control

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

Read calibration settings.

This function fill structure with calibration settings.

See Also

calibration_settings_t

Parameters

	id	an identifier of device
out	*calibration_settings*	calibration settings

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

Return device electrical parameters, useful for charts.

Useful function that fill structure with snapshot of controller voltages and currents.

See Also

chart_data_t

Parameters

	id	an identifier of device
out	*chart_data*	structure with snapshot of controller parameters.

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

Read settings of motor control.

When choosing CTL_MODE = 1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i = 0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL_MODE = 2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout [i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i +1] to acceleration, as usual.

Parameters

	id	an identifier of device
out	*control_settings*	structure contains settings motor control by joystick or buttons left/right.

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

Read user controller name and flags of setting from FRAM.

Parameters

	id	an identifier of device
out	*controller_name*	structure contains previously set user controller name

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

Read settings of control position(is only used with stepper motor).

When controlling the step motor with encoder (CTP_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE_CTP_ERROR. When controlling the step motor with speed sensor (CTP_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE_CTP_ERROR.

Parameters

	id	an identifier of device
out	*ctp_settings*	structure contains settings of control position

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

Read data from firmware for debug purpose.

Its use depends on context, firmware version and previous history.

Parameters

	id	an identifier of device
`out`	*debug_read*	Debug data.

6.1.4.38 int **XIMC_API** get_device_count (**device_enumeration_t** device_enumeration)

Get device count.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data

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

Return device information.

All fields must point to allocated string buffers with at least 10 bytes. Works with both raw or initialized device.

Parameters

	id	an identifier of device
`out`	*device_information*	device information Device information.

See Also

get_device_information

6.1.4.40 **pchar XIMC_API** get_device_name (**device_enumeration_t** device_enumeration, int device_index)

Get device name from the device enumeration.

Returns *device_index* device name.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data
`in`	*device_index*	device index

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

Read border and limit switches settings.

See Also

set_edges_settings

Parameters

	id	an identifier of device
`out`	*edges_settings*	edges settings, specify types of borders, motor behaviour and electrical behaviour of limit switches

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

Read encoder information from EEPROM.

Parameters

	id	an identifier of device
out	*encoder_information*	structure contains information about encoder

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

Read encoder settings from EEPROM.

Parameters

	id	an identifier of device
out	*encoder_settings*	structure contains encoder settings

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

Read engine settings.

This function fill structure with set of useful motor settings stored in controller's memory. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics.

See Also

set_engine_settings

Parameters

	id	an identifier of device
out	*engine_settings*	engine settings

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

Return engine type and driver type.

Parameters

	id	an identifier of device
out	*EngineType*	engine type
out	*DriverType*	driver type

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

Get controller name from the device enumeration.

Returns *device_index* device controller name.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data
`in`	*device_index*	device index
`out`	*controller_name*	controller name

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

Get device information from the device enumeration.

Returns *device_index* device information.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data
`in`	*device_index*	device index
`out`	*device_information*	device information data

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

Get device network information from the device enumeration.

Returns *device_index* device network information.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data
`in`	*device_index*	device index
`out`	*device_network_information*	device network information data

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

Get device serial number from the device enumeration.

Returns *device_index* device serial number.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data
`in`	*device_index*	device index
`out`	*serial*	device serial number

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

Get stage name from the device enumeration.

Returns *device_index* device stage name.

Parameters

`in`	*device_enumeration*	opaque pointer to an enumeration device data
`in`	*device_index*	device index
`out`	*stage_name*	stage name

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

Read EXTIO settings.

This function reads a structure with a set of EXTIO settings from controller's memory.

See Also

set_extio_settings

Parameters

	id	an identifier of device
`out`	*extio_settings*	EXTIO settings

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

Feedback settings.

Parameters

	id	an identifier of device
`out`	*IPS*	number of encoder counts per shaft revolution. Range: 1..65535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version.
`out`	*FeedbackType*	type of feedback
`out`	*FeedbackFlags*	flags of feedback
`out`	*CountsPerTurn*	number of encoder counts per shaft revolution. Range: 1..4294967295. To use the CountsPerTurn field, write 0 in the IPS field, otherwise the value from the IPS field will be used.

6.1.4.53 **result_t XIMC_API** get_firmware_version (**device_t** id, unsigned int * Major, unsigned int * Minor, unsigned int * Release)

Read controller's firmware version.

Parameters

	id	an identifier of device
`out`	*Major*	major version
`out`	*Minor*	minor version
`out`	*Release*	release version

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

Read gear information from EEPROM.

Parameters

	id	an identifier of device
out	*gear_information*	structure contains information about step gearhead

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

Read gear settings from EEPROM.

Parameters

	id	an identifier of device
out	*gear_settings*	structure contains step gearhead settings

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

This value is unique to each individual die but is not a random value.

This unique device identifier can be used to initiate secure boot processes or as a serial number for USB or other end applications.

Parameters

	id	an identifier of device
out	*the*	result of fields 0-3 concatenated defines the unique 128-bit device identifier.

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

Read hall sensor information from EEPROM.

Parameters

	id	an identifier of device
out	*hallsensor_information*	structure contains information about hall sensor

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

Read hall sensor settings from EEPROM.

Parameters

	id	an identifier of device
out	*hallsensor_settings*	structure contains hall sensor settings

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

Read home settings.

This function fill structure with settings of calibrating position.

See Also

home_settings_t

Parameters

	id	an identifier of device
`out`	*home_settings*	calibrating position settings

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

Read random number from controller.

Parameters

	id	an identifier of device
`out`	*random*	sequence generated by the controller

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

Read settings of joystick.

If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture. !/attachments/download/5563/range25p.png! The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this: !/attachments/download/3092/ExpJoystick.png! The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Parameters

	id	an identifier of device
`out`	*joystick_settings*	structure contains joystick settings

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

A command to read the data buffer to build a speed graph and a sequence error.

Filling the buffer starts with the command "start_measurements". The buffer holds 25 points, the points are taken with a period of 1 ms. To create a robust system, read data every 20 ms, if the buffer is completely full, then it is recommended to repeat the readings every 5 ms until the buffer again becomes filled with 20 points.

See Also

get_measurements_t

Parameters

	id	an identifier of device
out	*get_measurements*	structure with buffer and its length.

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

Read motor information from EEPROM.

Parameters

	id	an identifier of device
out	*motor_information*	structure contains motor information

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

Read motor settings from EEPROM.

Parameters

	id	an identifier of device
out	*motor_settings*	structure contains motor settings

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

Read command setup movement (speed, acceleration, threshold and etc).

Parameters

	id	an identifier of device
out	*move_settings*	structure contains move settings: speed, acceleration, deceleration etc.

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

Read userdata from FRAM.

Parameters

	id	an identifier of device
out	*nonvolatile_memory*	structure contains previously set userdata

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

Read PID settings.

This function fill structure with set of motor PID settings stored in controller's memory. These settings specify behaviour of PID routine for positioner. These factors are slightly different for different positioners. All boards are supplied with standard set of PID setting on controller's flash memory.

See Also

set_pid_settings

Parameters

	id	an identifier of device
`out`	*pid_settings*	pid settings

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

Reads the value position in steps and micro for stepper motor and encoder steps all engines.

Parameters

	id	an identifier of device
`out`	*position*	structure contains move settings: speed, acceleration, deceleration etc.

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

Read settings of step motor power control.

Used with stepper motor only.

Parameters

	id	an identifier of device
`out`	*power_settings*	structure contains settings of step motor power control

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

Read protection settings.

Parameters

	id	an identifier of device
`out`	*secure_settings*	critical parameter settings to protect the hardware

See Also

status_t::flags

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

Read device serial number.

Parameters

	id	an identifier of device
`out`	*serial*	serial number

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

Read stage information from EEPROM.

Parameters

	id	an identifier of device
out	*stage_-information*	structure contains stage information

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

Read user stage name from EEPROM.

Parameters

	id	an identifier of device
out	*stage_name*	structure contains previously set user stage name

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

Read stage settings from EEPROM.

Parameters

	id	an identifier of device
out	*stage_settings*	structure contains stage settings

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

Return device state.

Parameters

	id	an identifier of device
out	*status*	structure with snapshot of controller status Device state. Useful structure that contains current controller status, including speed, position and boolean flags.

See Also

get_status

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

Calibrated device state.

Useful structure that contains current controller status, including speed, position and boolean flags.

See Also

get_status

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

Read input synchronization settings.

This function fill structure with set of input synchronization settings, modes, periods and flags, that specify behaviour of input synchronization. All boards are supplied with standard set of these settings.

See Also

set_sync_in_settings

Parameters

	id	an identifier of device
`out`	*sync_in_settings*	synchronization settings

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

Read output synchronization settings.

This function fill structure with set of output synchronization settings, modes, periods and flags, that specify behaviour of output synchronization. All boards are supplied with standard set of these settings.

See Also

set_sync_out_settings

Parameters

	id	an identifier of device
`out`	*sync_out_settings*	synchronization settings

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

Read UART settings.

This function fill structure with UART settings.

See Also

uart_settings_t

Parameters

	Speed	UART speed
`out`	*uart_settings*	UART settings

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

Reboot to firmware.

Parameters

	id	an identifier of device

`out`	*ret*	RESULT_OK, if reboot to firmware is possible. Reboot is done after reply to this command. RESULT_NO_FIRMWARE, if firmware is not found. RESULT_ALREADY_IN_FIRMWARE, if this command was sent when controller is already in firmware.

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

Check for firmware on device.

Parameters

	uri	a uri of device
`out`	*ret*	non-zero if firmware existed

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

Simple callback for logging to stderr in narrow (single byte) chars.

Parameters

loglevel	a loglevel
message	a message

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

Simple callback for logging to stderr in wide chars.

Parameters

loglevel	a loglevel
message	a message

6.1.4.84 void **XIMC_API** msec_sleep (unsigned int msec)

Sleeps for a specified amount of time.

Parameters

msec	time in milliseconds

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

Open a device with OS uri *uri* and return identifier of the device which can be used in calls.

Parameters

in	*uri*	a device uri Device uri has form "xi-com:port" or "xi-net://host/serial" or "xi-emu-:///file". In case of USB-COM port the "port" is the OS device uri. For example "xi-com:\\\\.\\COM3" in Windows or "xi-com:/dev/tty.s123" in Linux/Mac. In case of network device the "host" is an IPv4 address or fully qualified domain uri (F-QDN), "serial" is the device serial number in hexadecimal system. For example "xi-net://192.168.0.1/00001234" or "xi-net://hostname.com/89ABCDEF". Note-: to open network device you must call set_bindy_key first. In case of virtual device the "file" is the full filename with device memory state, if it doesn't exist then it is initialized with default values. For example "xi-emu:///C:/dir/file.bin" in Windows or "xi-emu:///home/user/file.bin" in Linux/Mac.

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

Check if a device with OS uri *uri* is XIMC device.

Be carefuly with this call because it sends some data to the device.

Parameters

in	*uri*	- a device uri

6.1.4.87 **result_t XIMC_API** service_command_updf (**device_t** id)

Command puts the controller to update the firmware.

After receiving this command, the firmware board sets a flag (for loader), sends echo reply and restarts the controller.

6.1.4.88 **result_t XIMC_API** set_accessories_settings (**device_t** id, const **accessories_settings_t** * accessories_settings)

Set additional accessories information to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*accessories_settings*	structure contains information about additional accessories

6.1.4.89 **result_t XIMC_API** set_bindy_key (const char * keyfilepath)

Set network encryption layer (bindy) key.

Parameters

in	*keyfilepath*	full path to the bindy keyfile When using network-attached devices this function must be called before enumerate_devices and open_device functions.

6.1.4.90 **result_t XIMC_API** set_brake_settings (**device_t** id, const **brake_settings_t** * brake_settings)

Set settings of brake control.

Parameters

	id	an identifier of device
in	*brake_settings*	structure contains settings of brake control

6.1.4.91 **result_t XIMC_API** set_calibration_settings (**device_t** id, const **calibration_settings_t** * calibration_settings)

Set calibration settings.

This function send structure with calibration settings to controller's memory.

See Also

calibration_settings_t

Parameters

	id	an identifier of device
in	*calibration_settings*	calibration settings

6.1.4.92 **result_t XIMC_API** set_control_settings (**device_t** id, const **control_settings_t** * control_settings)

Set settings of motor control.

When choosing CTL_MODE = 1 switches motor control with the joystick. In this mode, the joystick to the maximum engine tends Move at MaxSpeed [i], where i = 0 if the previous use This mode is not selected another i. Buttons switch the room rate i. When CTL_MODE = 2 is switched on motor control using the Left / right. When you click on the button motor starts to move in the appropriate direction at a speed MaxSpeed [0], at the end of time Timeout [i] motor move at a speed MaxSpeed [i+1]. at Transition from MaxSpeed [i] on MaxSpeed [i +1] to acceleration, as usual.

Parameters

	id	an identifier of device
in	*control_settings*	structure contains settings motor control by joystick or buttons left/right.

6.1.4.93 **result_t XIMC_API** set_controller_name (**device_t** id, const **controller_name_t** * controller_name)

Write user controller name and flags of setting from FRAM.

Parameters

	id	an identifier of device
in	*controller_name*	structure contains previously set user controller name

6.1.4.94 **result_t XIMC_API** set_ctp_settings (**device_t** id, const **ctp_settings_t** * ctp_settings)

Set settings of control position(is only used with stepper motor).

When controlling the step motor with encoder (CTP_BASE 0) it is possible to detect the loss of steps. The controller knows the number of steps per revolution (GENG :: StepsPerRev) and the encoder resolution (GFBS :: IPT). When the control (flag CTP_ENABLED), the controller stores the current position in the footsteps of SM and the current position of the encoder. Further, at each step of the position encoder is converted into steps and if the difference is greater CTPMinError, a flag STATE_CTP_ERROR. When controlling the step motor with speed sensor (CTP_BASE 1), the position is controlled by him. The active edge of input clock controller stores the current value of steps. Further, at each turn checks how many steps shifted. When a mismatch CTPMinError a flag STATE_CTP_ERROR.

Parameters

	id	an identifier of device
`in`	*ctp_settings*	structure contains settings of control position

6.1.4.95 **result_t XIMC_API** set_debug_write (**device_t** id, const **debug_write_t** ∗ debug_write)

Write data to firmware for debug purpose.

Parameters

	id	an identifier of device
`in`	*debug_write*	Debug data.

6.1.4.96 **result_t XIMC_API** set_edges_settings (**device_t** id, const **edges_settings_t** ∗ edges_settings)

Set border and limit switches settings.

See Also

set_edges_settings

Parameters

	id	an identifier of device
`in`	*edges_settings*	edges settings, specify types of borders, motor behaviour and electrical behaviour of limit switches

6.1.4.97 **result_t XIMC_API** set_encoder_information (**device_t** id, const **encoder_information_t** ∗ encoder_information)

Set encoder information to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
`in`	*encoder_information*	structure contains information about encoder

6.1.4.98 **result_t XIMC_API** set_encoder_settings (**device_t** id, const **encoder_settings_t** ∗ encoder_settings)

Set encoder settings to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
`in`	*encoder_settings*	structure contains encoder settings

6.1.4.99 **result_t XIMC_API** set_engine_settings (**device_t** id, const **engine_settings_t** * engine_settings)

Set engine settings.

This function send structure with set of engine settings to controller's memory. These settings specify motor shaft movement algorithm, list of limitations and rated characteristics. Use it when you change motor, encoder, positioner etc. Please note that wrong engine settings lead to device malfunction, can lead to irreversible damage of board.

See Also

get_engine_settings

Parameters

	id	an identifier of device
`in`	*engine_settings*	engine settings

6.1.4.100 **result_t XIMC_API** set_entype_settings (**device_t** id, const **entype_settings_t** * entype_settings)

Set engine type and driver type.

Parameters

	id	an identifier of device
`in`	*EngineType*	engine type
`in`	*DriverType*	driver type

6.1.4.101 **result_t XIMC_API** set_extio_settings (**device_t** id, const **extio_settings_t** * extio_settings)

Set EXTIO settings.

This function writes a structure with a set of EXTIO settings to controller's memory. By default input event are signalled through rising front and output states are signalled by high logic state.

See Also

get_extio_settings

Parameters

	id	an identifier of device
`in`	*extio_settings*	EXTIO settings

6.1.4.102 **result_t XIMC_API** set_feedback_settings (**device_t** id, const **feedback_settings_t** * feedback_settings)

Feedback settings.

Parameters

	id	an identifier of device
in	*IPS*	number of encoder counts per shaft revolution. Range: 1..65535. The field is obsolete, it is recommended to write 0 to IPS and use the extended CountsPerTurn field. You may need to update the controller firmware to the latest version.
in	*FeedbackType*	type of feedback
in	*FeedbackFlags*	flags of feedback
in	*CountsPerTurn*	number of encoder counts per shaft revolution. Range: 1..4294967295. To use the CountsPerTurn field, write 0 in the IPS field, otherwise the value from the IPS field will be used.

6.1.4.103 **result_t XIMC_API** set_gear_information (**device_t** id, const **gear_information_t** * gear_information)

Set gear information to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*gear_information*	structure contains information about step gearhead

6.1.4.104 **result_t XIMC_API** set_gear_settings (**device_t** id, const **gear_settings_t** * gear_settings)

Set gear settings to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*gear_settings*	structure contains step gearhead settings

6.1.4.105 **result_t XIMC_API** set_hallsensor_information (**device_t** id, const **hallsensor_information_t** * hallsensor_information)

Set hall sensor information to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*hallsensor_information*	structure contains information about hall sensor

6.1.4.106 **result_t XIMC_API** set_hallsensor_settings (**device_t** id, const **hallsensor_settings_t** * hallsensor_settings)

Set hall sensor settings to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*hallsensor_settings*	structure contains hall sensor settings

6.1.4.107 **result_t XIMC_API** set_home_settings (**device_t** id, const **home_settings_t** * home_settings)

Set home settings.

This function send structure with calibrating position settings to controller's memory.

See Also

home_settings_t

Parameters

	id	an identifier of device
in	*home_settings*	calibrating position settings

6.1.4.108 **result_t XIMC_API** set_joystick_settings (**device_t** id, const **joystick_settings_t** * joystick_settings)

Set settings of joystick.

If joystick position is outside DeadZone limits from the central position a movement with speed, defined by the joystick DeadZone edge to 100% deviation, begins. Joystick positions inside DeadZone limits correspond to zero speed (soft stop of motion) and positions beyond Low and High limits correspond MaxSpeed [i] or -MaxSpeed [i] (see command SCTL), where i = 0 by default and can be changed with left/right buttons (see command SCTL). If next speed in list is zero (both integer and microstep parts), the button press is ignored. First speed in list shouldn't be zero. The DeadZone ranges are illustrated on the following picture. !/attachments/download/5563/range25p.png! The relationship between the deviation and the rate is exponential, allowing no switching speed combine high mobility and accuracy. The following picture illustrates this: !/attachments/download/3092/ExpJoystick.png! The nonlinearity parameter is adjustable. Setting it to zero makes deviation/speed relation linear.

Parameters

	id	an identifier of device
in	*joystick_settings*	structure contains joystick settings

6.1.4.109 void **XIMC_API** set_logging_callback (**logging_callback_t** logging_callback, void * user_data)

Sets a logging callback.

Call resets a callback to default (stderr, syslog) if NULL passed.

Parameters

logging_callback	a callback for log messages

6.1.4.110 **result_t XIMC_API** set_motor_information (**device_t** id, const **motor_information_t** * motor_information)

Set motor information to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*motor_information*	structure contains motor information

6.1.4.111 **result_t XIMC_API** set_motor_settings (**device_t** id, const **motor_settings_t** * motor_settings)

Set motor settings to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
in	*motor_settings*	structure contains motor information

6.1.4.112 **result_t XIMC_API** set_move_settings (**device_t** id, const **move_settings_t** * move_settings)

Set command setup movement (speed, acceleration, threshold and etc).

Parameters

	id	an identifier of device
in	*move_settings*	structure contains move settings: speed, acceleration, deceleration etc.

6.1.4.113 **result_t XIMC_API** set_nonvolatile_memory (**device_t** id, const **nonvolatile_memory_t** * nonvolatile_memory)

Write userdata into FRAM.

Parameters

	id	an identifier of device
in	*nonvolatile_memory*	structure contains previously set userdata

6.1.4.114 **result_t XIMC_API** set_pid_settings (**device_t** id, const **pid_settings_t** * pid_settings)

Set PID settings.

This function send structure with set of PID factors to controller's memory. These settings specify behaviour of PID routine for positioner. These factors are slightly different for different positioners. All boards are supplied with standard set of PID setting on controller's flash memory. Please use it for loading new PID settings when you change positioner. Please note that wrong PID settings lead to device malfunction.

See Also

get_pid_settings

Parameters

	id	an identifier of device
in	*pid_settings*	pid settings

6.1.4.115 **result_t XIMC_API** set_position (**device_t** id, const **set_position_t** * the_set_position)

Sets any position value in steps and micro for stepper motor and encoder steps of all engines.

It means, that changing main indicator of position.

Parameters

	id	an identifier of device
`out`	*position*	structure contains move settings: speed, acceleration, deceleration etc.

6.1.4.116 **result_t XIMC_API** set_power_settings (**device_t** id, const **power_settings_t** * power_settings)

Set settings of step motor power control.

Used with stepper motor only.

Parameters

	id	an identifier of device
`in`	*power_settings*	structure contains settings of step motor power control

6.1.4.117 **result_t XIMC_API** set_secure_settings (**device_t** id, const **secure_settings_t** * secure_settings)

Set protection settings.

Parameters

id	an identifier of device
secure_settings	structure with secure data

See Also

status_t::flags

6.1.4.118 **result_t XIMC_API** set_serial_number (**device_t** id, const **serial_number_t** * serial_number)

Write device serial number and hardware version to controller's flash memory.

Along with the new serial number and hardware version a ”Key” is transmitted. The SN and hardware version are changed and saved when keys match. Can be used by manufacturer only.

Parameters

	id	an identifier of device
`in`	*serial*	number structure contains new serial number and secret key.

6.1.4.119 **result_t XIMC_API** set_stage_information (**device_t** id, const **stage_information_t** * stage_information)

Set stage information to EEPROM.

Can be used by manufacturer only.

Parameters

	id	an identifier of device
`in`	*stage_information*	structure contains stage information

6.1.4.120 **result_t XIMC_API** set_stage_name (**device_t** id, const **stage_name_t** * stage_name)

Write user stage name from EEPROM.

Parameters

	id	an identifier of device
`in`	*stage_name*	structure contains previously set user stage name

6.1.4.121 **result_t XIMC_API** set_stage_settings (**device_t** id, const **stage_settings_t** * stage_settings)

Set stage settings to EEPROM.

Can be used by manufacturer only

Parameters

	id	an identifier of device
`in`	*stage_settings*	structure contains stage settings

6.1.4.122 **result_t XIMC_API** set_sync_in_settings (**device_t** id, const **sync_in_settings_t** * sync_in_settings)

Set input synchronization settings.

This function send structure with set of input synchronization settings, that specify behaviour of input synchronization, to controller's memory. All boards are supplied with standard set of these settings.

See Also

get_sync_in_settings

Parameters

	id	an identifier of device
`in`	*sync_in_settings*	synchronization settings

6.1.4.123 **result_t XIMC_API** set_sync_out_settings (**device_t** id, const **sync_out_settings_t** * sync_out_settings)

Set output synchronization settings.

This function send structure with set of output synchronization settings, that specify behaviour of output synchronization, to controller's memory. All boards are supplied with standard set of these settings.

See Also

get_sync_out_settings

Parameters

	id	an identifier of device
in	*sync_out_-settings*	synchronization settings

6.1.4.124 **result_t XIMC_API** set_uart_settings (**device_t** id, const **uart_settings_t** * uart_settings)

Set UART settings.

This function send structure with UART settings to controller's memory.

See Also

uart_settings_t

Parameters

	Speed	UART speed
in	*uart_settings*	UART settings

6.1.4.125 **result_t XIMC_API** write_key (const char * uri, uint8_t * key)

Write controller key.

Can be used by manufacturer only

Parameters

	uri	a uri of device
in	*key*	protection key. Range: 0..4294967295

6.1.4.126 **result_t XIMC_API** ximc_fix_usbser_sys (const char * device_uri)

Fix for errors in Windows USB driver stack.

USB subsystem on Windows does not always work correctly. The following bugs are possible: the device cannot be opened at all, or the device can be opened and written to, but it will not respond with data. These errors can be fixed by device reconnection or removal-rescan in device manager. ximc_fix_usbser_sys() is a shortcut function to do the remove-rescan process. You should call this function if libximc library cannot open the device which was not physically removed from the system or if the device does not respond.

6.1.4.127 void **XIMC_API** ximc_version (char * version)

Returns a library version.

Parameters

version	a buffer to hold a version string, 32 bytes is enough

Index