

libxime
2.2.0

Создано системой Doxygen 1.8.1

Сб 2 Ноя 2013 23:24:15

Оглавление

1	Введение	1
1.1	О библиотеке	1
1.2	Требования к установленному программному обеспечению	1
1.2.1	Для сборки библиотеки	1
1.2.2	Для использования библиотеки	2
2	Как пересобрать библиотеку	3
2.1	Сборка для UNIX	3
2.2	Сборка для Linux на основе Debian	3
2.3	Сборка для Linux на основе RedHat	3
2.4	Сборка для FreeBSD	4
2.5	Сборка для Mac OS X	4
2.6	Сборка в ОС Windows	4
2.7	Доступ к исходным кодам	4
3	Как использовать с...	5
3.1	Использование на C	5
3.1.1	Visual C++	5
3.1.2	MinGW	5
3.1.3	C++ Builder	5
3.1.4	XCode	6
3.1.5	GCC	6
3.2	.NET	6
3.3	Delphi	6
3.4	MATLAB	6
4	Структуры данных	8
4.1	Структура accessories_settings_t	8
4.1.1	Подробное описание	8
4.1.2	Поля	9
4.1.2.1	MBRatedCurrent	9
4.1.2.2	MBRatedVoltage	9

4.1.2.3	MBTorque	9
4.1.2.4	TSGrad	9
4.1.2.5	TSTMin	9
4.2	Структура <code>add_sync_in_action_calb_t</code>	9
4.3	Структура <code>add_sync_in_action_t</code>	9
4.3.1	Подробное описание	10
4.3.2	Поля	10
4.3.2.1	Speed	10
4.3.2.2	uSpeed	10
4.4	Структура <code>analog_data_t</code>	10
4.4.1	Подробное описание	11
4.5	Структура <code>brake_settings_t</code>	11
4.5.1	Подробное описание	12
4.5.2	Поля	12
4.5.2.1	t1	12
4.5.2.2	t2	12
4.5.2.3	t3	12
4.5.2.4	t4	12
4.6	Структура <code>calibration_t</code>	12
4.6.1	Подробное описание	13
4.7	Структура <code>chart_data_t</code>	13
4.7.1	Подробное описание	13
4.8	Структура <code>control_settings_calb_t</code>	14
4.8.1	Поля	14
4.8.1.1	MaxClickTime	14
4.8.1.2	Timeout	14
4.9	Структура <code>control_settings_t</code>	14
4.9.1	Подробное описание	15
4.9.2	Поля	15
4.9.2.1	MaxClickTime	15
4.9.2.2	MaxSpeed	15
4.9.2.3	Timeout	15
4.9.2.4	uMaxSpeed	15
4.10	Структура <code>controller_name_t</code>	15
4.10.1	Подробное описание	16
4.10.2	Поля	16
4.10.2.1	ControllerName	16
4.11	Структура <code>ctp_settings_t</code>	16
4.11.1	Подробное описание	16
4.11.2	Поля	16

4.11.2.1	CTPMinError	16
4.12	Структура debug_read_t	17
4.12.1	Подробное описание	17
4.13	Структура device_information_t	17
4.13.1	Подробное описание	17
4.14	Структура edges_settings_calb_t	17
4.15	Структура edges_settings_t	18
4.15.1	Подробное описание	18
4.15.2	Поля	18
4.15.2.1	LeftBorder	18
4.15.2.2	RightBorder	18
4.15.2.3	uLeftBorder	19
4.15.2.4	uRightBorder	19
4.16	Структура encoder_information_t	19
4.16.1	Подробное описание	19
4.16.2	Поля	19
4.16.2.1	Manufacturer	19
4.16.2.2	PartNumber	19
4.17	Структура encoder_settings_t	19
4.17.1	Подробное описание	20
4.17.2	Поля	20
4.17.2.1	MaxCurrentConsumption	20
4.17.2.2	MaxOperatingFrequency	20
4.17.2.3	SupplyVoltageMax	20
4.17.2.4	SupplyVoltageMin	20
4.18	Структура engine_settings_calb_t	21
4.18.1	Поля	21
4.18.1.1	Antiplay	21
4.18.1.2	NomCurrent	21
4.18.1.3	NomSpeed	21
4.18.1.4	NomVoltage	21
4.18.1.5	StepsPerRev	21
4.19	Структура engine_settings_t	22
4.19.1	Подробное описание	22
4.19.2	Поля	22
4.19.2.1	Antiplay	22
4.19.2.2	NomCurrent	22
4.19.2.3	NomSpeed	23
4.19.2.4	NomVoltage	23
4.19.2.5	StepsPerRev	23

4.19.2.6	uNomSpeed	23
4.20	Структура entype_settings_t	23
4.20.1	Подробное описание	23
4.21	Структура extio_settings_t	24
4.21.1	Подробное описание	24
4.22	Структура feedback_settings_t	24
4.22.1	Подробное описание	25
4.23	Структура gear_information_t	25
4.23.1	Подробное описание	25
4.23.2	Поля	25
4.23.2.1	Manufacturer	25
4.23.2.2	PartNumber	25
4.24	Структура gear_settings_t	25
4.24.1	Подробное описание	26
4.24.2	Поля	26
4.24.2.1	Efficiency	26
4.24.2.2	InputInertia	26
4.24.2.3	MaxOutputBacklash	26
4.24.2.4	RatedInputSpeed	26
4.24.2.5	RatedInputTorque	26
4.24.2.6	ReductionIn	27
4.24.2.7	ReductionOut	27
4.25	Структура get_position_calb_t	27
4.26	Структура get_position_t	27
4.26.1	Подробное описание	27
4.27	Структура hallsensor_information_t	27
4.27.1	Подробное описание	28
4.27.2	Поля	28
4.27.2.1	Manufacturer	28
4.27.2.2	PartNumber	28
4.28	Структура hallsensor_settings_t	28
4.28.1	Подробное описание	28
4.28.2	Поля	29
4.28.2.1	MaxCurrentConsumption	29
4.28.2.2	MaxOperatingFrequency	29
4.28.2.3	SupplyVoltageMax	29
4.28.2.4	SupplyVoltageMin	29
4.29	Структура home_settings_calb_t	29
4.30	Структура home_settings_t	29
4.30.1	Подробное описание	30

4.30.2 Поля	30
4.30.2.1 FastHome	30
4.30.2.2 HomeDelta	30
4.30.2.3 SlowHome	30
4.30.2.4 uFastHome	30
4.30.2.5 uHomeDelta	31
4.30.2.6 uSlowHome	31
4.31 Структура joystick_settings_t	31
4.31.1 Подробное описание	31
4.31.2 Поля	32
4.31.2.1 DeadZone	32
4.31.2.2 JoyCenter	32
4.31.2.3 JoyHighEnd	32
4.31.2.4 JoyLowEnd	32
4.32 Структура motor_information_t	32
4.32.1 Подробное описание	32
4.32.2 Поля	32
4.32.2.1 Manufacturer	32
4.32.2.2 PartNumber	33
4.33 Структура motor_settings_t	33
4.33.1 Подробное описание	34
4.33.2 Поля	34
4.33.2.1 DetentTorque	34
4.33.2.2 MaxCurrent	34
4.33.2.3 MaxCurrentTime	34
4.33.2.4 MaxSpeed	34
4.33.2.5 MechanicalTimeConstant	34
4.33.2.6 NoLoadCurrent	35
4.33.2.7 NoLoadSpeed	35
4.33.2.8 NominalCurrent	35
4.33.2.9 NominalPower	35
4.33.2.10 NominalSpeed	35
4.33.2.11 NominalTorque	35
4.33.2.12 NominalVoltage	35
4.33.2.13 RotorInertia	35
4.33.2.14 SpeedConstant	35
4.33.2.15 SpeedTorqueGradient	36
4.33.2.16 StallTorque	36
4.33.2.17 TorqueConstant	36
4.33.2.18 WindingInductance	36

4.33.2.19 WindingResistance	36
4.34 Структура move_settings_calb_t	36
4.35 Структура move_settings_t	36
4.35.1 Подробное описание	37
4.35.2 Поля	37
4.35.2.1 Accel	37
4.35.2.2 AntiplaySpeed	37
4.35.2.3 Decel	37
4.35.2.4 Speed	37
4.35.2.5 uAntiplaySpeed	37
4.35.2.6 uSpeed	38
4.36 Структура pid_settings_t	38
4.36.1 Подробное описание	38
4.37 Структура power_settings_t	38
4.37.1 Подробное описание	39
4.37.2 Поля	39
4.37.2.1 CurrentSetTime	39
4.37.2.2 CurrReductDelay	39
4.37.2.3 HoldCurrent	39
4.37.2.4 PowerOffDelay	39
4.38 Структура secure_settings_t	39
4.38.1 Подробное описание	40
4.38.2 Поля	40
4.38.2.1 CriticalIpwr	40
4.38.2.2 CriticalIusb	40
4.38.2.3 CriticalT	40
4.38.2.4 CriticalUpwr	40
4.38.2.5 CriticalUusb	40
4.38.2.6 LowUpwrOff	40
4.38.2.7 MinimumUusb	41
4.39 Структура serial_number_t	41
4.39.1 Подробное описание	41
4.40 Структура set_position_calb_t	41
4.41 Структура set_position_t	41
4.41.1 Подробное описание	42
4.42 Структура stage_information_t	42
4.42.1 Подробное описание	42
4.42.2 Поля	42
4.42.2.1 Manufacturer	42
4.42.2.2 PartNumber	42

4.43 Структура <code>stage_name_t</code>	42
4.43.1 Подробное описание	43
4.43.2 Поля	43
4.43.2.1 <code>PositionerName</code>	43
4.44 Структура <code>stage_settings_t</code>	43
4.44.1 Подробное описание	44
4.44.2 Поля	44
4.44.2.1 <code>HorizontalLoadCapacity</code>	44
4.44.2.2 <code>LeadScrewPitch</code>	44
4.44.2.3 <code>MaxCurrentConsumption</code>	44
4.44.2.4 <code>MaxSpeed</code>	44
4.44.2.5 <code>SupplyVoltageMax</code>	44
4.44.2.6 <code>SupplyVoltageMin</code>	44
4.44.2.7 <code>TravelRange</code>	44
4.44.2.8 <code>VerticalLoadCapacity</code>	44
4.45 Структура <code>status_calb_t</code>	45
4.45.1 Поля	45
4.45.1.1 <code>CurPosition</code>	45
4.46 Структура <code>status_t</code>	46
4.46.1 Подробное описание	47
4.46.2 Поля	47
4.46.2.1 <code>CurPosition</code>	47
4.46.2.2 <code>uCurPosition</code>	47
4.46.2.3 <code>uCurSpeed</code>	47
4.47 Структура <code>sync_in_settings_calb_t</code>	47
4.47.1 Поля	47
4.47.1.1 <code>ClutterTime</code>	47
4.48 Структура <code>sync_in_settings_t</code>	48
4.48.1 Подробное описание	48
4.48.2 Поля	48
4.48.2.1 <code>ClutterTime</code>	48
4.48.2.2 <code>Speed</code>	48
4.48.2.3 <code>uSpeed</code>	48
4.49 Структура <code>sync_out_settings_calb_t</code>	49
4.49.1 Поля	49
4.49.1.1 <code>SyncOutPeriod</code>	49
4.49.1.2 <code>SyncOutPulseSteps</code>	49
4.50 Структура <code>sync_out_settings_t</code>	49
4.50.1 Подробное описание	50
4.50.2 Поля	50

4.50.2.1	Accuracy	50
4.50.2.2	SyncOutPeriod	50
4.50.2.3	SyncOutPulseSteps	50
4.50.2.4	uAccuracy	50
4.51	Структура <code>uart_settings_t</code>	50
4.51.1	Подробное описание	51
5	Файлы	52
5.1	Файл <code>ximc.h</code>	52
5.1.1	Подробное описание	74
5.1.2	Макросы	74
5.1.2.1	ALARM_ON_DRIVER_OVERHEATING	74
5.1.2.2	BORDER_IS_ENCODER	74
5.1.2.3	BORDERS_SWAP_MISSET_DETECTION	74
5.1.2.4	DRIVER_TYPE_DISCRETE_FET	74
5.1.2.5	ENGINE_ACCEL_ON	74
5.1.2.6	ENGINE_ANTIPLAY	74
5.1.2.7	ENGINE_LIMIT_CURR	75
5.1.2.8	ENGINE_LIMIT_RPM	75
5.1.2.9	ENGINE_LIMIT_VOLT	75
5.1.2.10	ENGINE_MAX_SPEED	75
5.1.2.11	ENGINE_REVERSE	75
5.1.2.12	ENUMERATE_PROBE	75
5.1.2.13	HOME_DIR_FIRST	75
5.1.2.14	HOME_DIR_SECOND	75
5.1.2.15	JOY_REVERSE	76
5.1.2.16	MVCMD_ERROR	76
5.1.2.17	POWER_OFF_ENABLED	76
5.1.2.18	POWER_REDUCT_ENABLED	76
5.1.2.19	REV_SENS_INV	76
5.1.2.20	STATE_ALARM	76
5.1.2.21	SYNCIN_INVERT	76
5.1.2.22	SYNCOUT_ENABLED	76
5.1.2.23	SYNCOUT_STATE	76
5.1.2.24	XIMC_API	77
5.1.3	Типы	77
5.1.3.1	logging_callback_t	77
5.1.4	Функции	77
5.1.4.1	close_device	77
5.1.4.2	command_clear_fram	77

5.1.4.3	<code>command_eeread_settings</code>	77
5.1.4.4	<code>command_eesave_settings</code>	77
5.1.4.5	<code>command_home</code>	78
5.1.4.6	<code>command_left</code>	78
5.1.4.7	<code>command_loft</code>	78
5.1.4.8	<code>command_move</code>	78
5.1.4.9	<code>command_movr</code>	79
5.1.4.10	<code>command_power_off</code>	79
5.1.4.11	<code>command_read_settings</code>	79
5.1.4.12	<code>command_reset</code>	80
5.1.4.13	<code>command_right</code>	80
5.1.4.14	<code>command_save_settings</code>	80
5.1.4.15	<code>command_sstp</code>	80
5.1.4.16	<code>command_stop</code>	80
5.1.4.17	<code>command_update_firmware</code>	80
5.1.4.18	<code>command_zero</code>	81
5.1.4.19	<code>enumerate_devices</code>	81
5.1.4.20	<code>free_enumerate_devices</code>	81
5.1.4.21	<code>get_accessories_settings</code>	81
5.1.4.22	<code>get_analog_data</code>	81
5.1.4.23	<code>get_bootloader_version</code>	82
5.1.4.24	<code>get_brake_settings</code>	82
5.1.4.25	<code>get_chart_data</code>	82
5.1.4.26	<code>get_control_settings</code>	82
5.1.4.27	<code>get_controller_name</code>	83
5.1.4.28	<code>get_ctp_settings</code>	83
5.1.4.29	<code>get_debug_read</code>	83
5.1.4.30	<code>get_device_count</code>	83
5.1.4.31	<code>get_device_information</code>	84
5.1.4.32	<code>get_device_name</code>	84
5.1.4.33	<code>get_edges_settings</code>	84
5.1.4.34	<code>get_encoder_information</code>	84
5.1.4.35	<code>get_encoder_settings</code>	85
5.1.4.36	<code>get_engine_settings</code>	85
5.1.4.37	<code>get_entype_settings</code>	85
5.1.4.38	<code>get_enumerate_device_information</code>	85
5.1.4.39	<code>get_enumerate_device_serial</code>	86
5.1.4.40	<code>get_extio_settings</code>	86
5.1.4.41	<code>get_feedback_settings</code>	86
5.1.4.42	<code>get_firmware_version</code>	86

5.1.4.43	<code>get_gear_information</code>	87
5.1.4.44	<code>get_gear_settings</code>	87
5.1.4.45	<code>get_hallsensor_information</code>	87
5.1.4.46	<code>get_hallsensor_settings</code>	87
5.1.4.47	<code>get_home_settings</code>	87
5.1.4.48	<code>get_joystick_settings</code>	88
5.1.4.49	<code>get_motor_information</code>	88
5.1.4.50	<code>get_motor_settings</code>	88
5.1.4.51	<code>get_move_settings</code>	89
5.1.4.52	<code>get_pid_settings</code>	89
5.1.4.53	<code>get_position</code>	89
5.1.4.54	<code>get_power_settings</code>	89
5.1.4.55	<code>get_secure_settings</code>	90
5.1.4.56	<code>get_serial_number</code>	90
5.1.4.57	<code>get_stage_information</code>	90
5.1.4.58	<code>get_stage_name</code>	90
5.1.4.59	<code>get_stage_settings</code>	90
5.1.4.60	<code>get_status</code>	91
5.1.4.61	<code>get_status_calb</code>	91
5.1.4.62	<code>get_sync_in_settings</code>	91
5.1.4.63	<code>get_sync_out_settings</code>	91
5.1.4.64	<code>get_uart_settings</code>	91
5.1.4.65	<code>goto_firmware</code>	92
5.1.4.66	<code>has_firmware</code>	92
5.1.4.67	<code>logging_callback_stderr_narrow</code>	92
5.1.4.68	<code>logging_callback_stderr_wide</code>	92
5.1.4.69	<code>msec_sleep</code>	92
5.1.4.70	<code>open_device</code>	93
5.1.4.71	<code>probe_device</code>	93
5.1.4.72	<code>service_command_updf</code>	93
5.1.4.73	<code>set_accessories_settings</code>	93
5.1.4.74	<code>set_add_sync_in_action</code>	93
5.1.4.75	<code>set_brake_settings</code>	94
5.1.4.76	<code>set_control_settings</code>	94
5.1.4.77	<code>set_controller_name</code>	94
5.1.4.78	<code>set_ctp_settings</code>	94
5.1.4.79	<code>set_edges_settings</code>	95
5.1.4.80	<code>set_encoder_information</code>	95
5.1.4.81	<code>set_encoder_settings</code>	95
5.1.4.82	<code>set_engine_settings</code>	95

5.1.4.83	set_entype_settings	96
5.1.4.84	set_extio_settings	96
5.1.4.85	set_feedback_settings	96
5.1.4.86	set_gear_information	97
5.1.4.87	set_gear_settings	97
5.1.4.88	set_hallsensor_information	97
5.1.4.89	set_hallsensor_settings	97
5.1.4.90	set_home_settings	97
5.1.4.91	set_joystick_settings	98
5.1.4.92	set_logging_callback	98
5.1.4.93	set_motor_information	98
5.1.4.94	set_motor_settings	99
5.1.4.95	set_move_settings	99
5.1.4.96	set_pid_settings	99
5.1.4.97	set_position	99
5.1.4.98	set_power_settings	100
5.1.4.99	set_secure_settings	100
5.1.4.100	set_serial_number	100
5.1.4.101	set_stage_information	100
5.1.4.102	set_stage_name	101
5.1.4.103	set_stage_settings	101
5.1.4.104	set_sync_in_settings	101
5.1.4.105	set_sync_out_settings	101
5.1.4.106	set_uart_settings	102
5.1.4.107	write_key	102
5.1.4.108	ximc_fix_usbser_sys	102
5.1.4.109	ximc_version	102

Глава 1

Введение

1.1 О библиотеке

Спасибо, что вы выбрали мультиплатформенную библиотеку XIMC! Этот документ содержит всю необходимую информацию о библиотеке XIMC. Она использует распространенный и проверенный интерфейс виртуального последовательного порта, поэтому вы можете работать с модулями управления моторами через эту библиотеку практически под всеми под ОС, в том числе Windows 7, Windows Vista, Windows XP, Windows Server 2003, Windows 2000, Linux, Mac OS X. Библиотека XIMC поддерживает подключение и отключение устройств "на лету". Каждый запущенный экземпляр управляющей программы может работать только с одним устройством. Множественный доступ управляющих программ к одному и тому же устройству не допускается.

1.2 Требования к установленному программному обеспечению

1.2.1 Для сборки библиотеки

Для Windows:

- Windows 2000 или старше, 64-битная система (если планируется собирать обе архитектуры) или 32-битная система
- Microsoft Visual C++ 2008 или старше
- cygwin с tar, bison, flex

Для Linux или FreeBSD:

- 64-битная и/или 32-битная система
- gcc 4 или новее
- стандартные autotools: autoconf, autoheader, aclocal, automake, autoreconf, libtool
- gmake
- doxygen - для сборки документации
- LaTeX distribution (teTeX or texlive) - для сборки документации
- flex 2.5.30+
- bison
- mercurial (для сборки версии для разработки из hg)

Для Mac OS X:

- XCode 4
- doxygen
- mactex
- autotools
- mercurial (для сборки версии для разработки из hg)

Для зависимости от mercurial. При использовании mercurial включите расширение 'purge' путем добавления в `~/.hgrc` следующих строк:

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

1.2.2 Для использования библиотеки

Поддерживаемые операционные системы (32 и 64 бита):

- Mac OS X 10.6
- Windows 2000 или старше
- Autotools-совместимый unix. Библиотека устанавливается из бинарного вида.
- Linux на основе debian. DEB собирается на Debian Squeeze 6
- Linux на основе rpm. RPM собирается на OpenSUSE 10
- FreeBSD 9.

Требования сборки:

- Windows: Microsoft Visual C++ 2008 или mingw (в данный момент не поддерживается)
- UNIX: gcc 4, gmake
- Mac OS X: XCode 4

Глава 2

Как пересобрать библиотеку

2.1 Сборка для UNIX

Обобщенная версия собирается обычными autotools.

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

Собранные файлы (библиотека, заголовочные файлы, документация) устанавливаются в локальную директорию `./dist/local`. Это билд для разработчика. Иногда необходимо указать дополнительные параметры командной строки для вашей системы. Проконсультируйтесь с последующими параграфами.

2.2 Сборка для Linux на основе Debian

Требования: 64-битная или 32-битная система на основе debian, ubuntu Примерный набор пакетов: gcc, autotools, autoconf, libtool, dpkg-dev, flex, bison, doxygen, texlive, mercurial

Необходимо соблюдать парность архитектуры библиотеки и системы: 64-битная библиотека может быть собрана только на 64-битной системе, а 32-битная - только на 32-битной.

Для сборки библиотеки и пакета запустите скрипт:

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

Пакеты располагаются в `./dist/latest/deb`, локально установленные файлы в `./dist/local`.

2.3 Сборка для Linux на основе RedHat

Требования: 64-батная система на основе redhat (Fedora, Red Hat, SUSE)

Примерный набор пакетов: gcc, autotools, autoconf, libtool, flex, bison, doxygen, texlive, mercurial

Возможно собрать 32-битную и 64-битную библиотеки на 64-битной системе, однако 64-битная библиотека не может быть собрана на 32-битной системе.

Для сборки библиотеки и пакета запустите скрипт:

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

Пакеты располагаются в `./dist/latest/rpm`, локально установленные файлы в `./dist/local`.

2.4 Сборка для FreeBSD

Требования: 64-битная или 32-битная FreeBSD

Примерный набор пакетов: gcc, autotools, autoconf, libtool, flex, bison, doxygen, teTeX, mercurial

Необходимо соблюдать парность архитектуры библиотеки и системы.

Внимание! Для простой сборки библиотеки необходимы следующие параметры:

```
$ ./build.sh lib LEX=/usr/local/bin/flex CXXFLAGS=-I/usr/local/include/flex
```

Также необходимо пропатчить configure.ac для исключения SOVER из названия пакета (freebsd) не использует соглашения linux по версиям библиотек).

Для сборки библиотеки и пакета запустите следующий скрипт. Требуются права суперпользователя для инсталляции порта и специальное дерево /usr/ports/local. Проконсультируйтесь со скриптом для деталей.

```
$ ./build.sh libfreebsd
```

Пакеты располагаются в ./dist/latest/freebsd.

2.5 Сборка для Mac OS X

Для сборки библиотеки и пакета запустите скрипт:

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

Собранная библиотека (классическая и фреймворк), приложения (классическая и фреймворк) и документация располагаются в ./dist/latest/macosx, локально инсталлированные файлы в ./dist/local.

2.6 Сборка в ОС Windows

Требования: 64-битный windows (сборочный скрипт собирает обе архитектуры), cygwin (должен быть установлен в пути по умолчанию), mercurial.

Запустите скрипт:

```
$ ./build.sh libfreebsd
```

Собранные файлы располагаются в ./dist/latest/win32 и ./dist/latest/win64

2.7 Доступ к исходным кодам

Исходные коды XIMC могут выданы по отдельному запросу.

Глава 3

Как использовать С...

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

3.1 Использование на С

3.1.1 Visual C++

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

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

Откройте проект `examples/testapp/testapp.sln`, выполните сборку и запустите приложение из среды разработки.

3.1.2 MinGW

MinGW это вариант GCC для платформы win32. Требуется установка пакета MinGW. В данный момент не поддерживается.

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

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

Далее скопируйте `libximc.dll` в текущую директорию и запустите `testapp.exe`.

3.1.3 C++ Builder

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

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

Затем скомпилируйте тестовое приложение:

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

3.1.4 XCode

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

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

3.1.5 GCC

Убедитесь, что libximc (с помощью rpm, deb, пакета freebsd или tarболла) установлена на вашей системе. Пакеты должны устанавливаться с помощью package manager'a вашей ОС. Для OS X предоставляется обычная библиотека dylib и фреймворк.

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

testapp может быть собран следующим образом с установленной библиотекой:

```
$ make
```

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

Затем запустите приложение с помощью:

```
$ make run
```

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

3.2 .NET

Для использования в .NET предлагается обертка wrappers/csharp/ximcnet.dll. Она распространяется в двух различных архитектурах и зависит от .NET 2.0.

Тестовые приложения на языке C# для Visual Studio 2008 расположены в директориях testcs (для C#) и testvbnet (для VB.NET). Откройте проекты, скомпилируйте и запустите.

3.3 Delphi

Обертка для использования в Delphi libximc.dll предлагается как модуль wrappers/pascal/ximc.pas

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

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

3.4 MATLAB

Тестовая программа на MATLAB testximc.m располагается в директории examples/testmatlab. Укажите в первых строках расположение библиотеки XIMC и запустите программу как:

\$ testximc

Глава 4

Структуры данных

4.1 Структура accessories_settings_t

Информация о дополнительных аксессуарах.

Поля данных

- char [MagneticBrakeInfo](#) [25]
Производитель и номер магнитного тормоза, Максимальная длина строки: 24 символов.
- float [MBRatedVoltage](#)
Номинальное напряжение для управления магнитным тормозом (В).
- float [MBRatedCurrent](#)
Номинальный ток для управления магнитным тормозом (А).
- float [MBTorque](#)
Удерживающий момент (мН м).
- unsigned int [MBSettings](#)
[Флаги настроек энкодера.](#)
- char [TemperatureSensorInfo](#) [25]
Производитель и номер температурного датчика, Максимальная длина строки: 24 символов.
- float [TSMIn](#)
Минимальная измеряемая температура (град Цельсия).
- float [TSMax](#)
Максимальная измеряемая температура (град Цельсия) Тип данных: float.
- float [TSGrad](#)
Температурный градиент (В/град Цельсия).
- unsigned int [TSSettings](#)
[Флаги настроек температурного датчика.](#)
- unsigned int [LimitSwitchesSettings](#)
[Флаги настроек температурного датчика.](#)

4.1.1 Подробное описание

Информация о дополнительных аксессуарах.

См. также

```
set_accessories_settings  
get_accessories_settings  
get_accessories_settings, set_accessories_settings
```

4.1.2 Поля

4.1.2.1 float MBRatedCurrent

Номинальный ток для управления магнитным тормозом (А).

Тип данных: float.

4.1.2.2 float MBRatedVoltage

Номинальное напряжение для управления магнитным тормозом (В).

Тип данных: float.

4.1.2.3 float MBTorque

Удерживающий момент (мН м).

Тип данных: float.

4.1.2.4 float TSGrad

Температурный градиент (В/град Цельсия).

Тип данных: float.

4.1.2.5 float TSMin

Минимальная измеряемая температура (град Цельсия).

Тип данных: float.

4.2 Структура `add_sync_in_action_calb_t`

Поля данных

- float [Position](#)
Желаемая позиция или смещение.
- float [Speed](#)
Заданная скорость.

4.3 Структура `add_sync_in_action_t`

Это команда добавляет один элемент в буфер FIFO команд.

Поля данных

- int [Position](#)
Желаемая позиция или смещение (целая часть)
- int [uPosition](#)
Дробная часть позиции или смещения в микрошагах (-255..255)(используется только с шаговым двигателем).
- unsigned int [Speed](#)

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

- unsigned int `uSpeed`

Заданная скорость в микрошагах в секунду.

4.3.1 Подробное описание

Это команда добавляет один элемент в буфер FIFO команд.

См. также

`set_add_sync_in_action`

4.3.2 Поля

4.3.2.1 unsigned int Speed

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

Диапазон: 0..1000000.

4.3.2.2 unsigned int uSpeed

Заданная скорость в микрошагах в секунду.

Используется только с шаговым мотором. Диапазон: 0..255.

4.4 Структура `analog_data_t`

Аналоговые данные.

Поля данных

- unsigned int `A1Voltage_ADC`
"Выходное напряжение на 1 выводе обмотки A" необработанные данные с АЦП.
- unsigned int `A2Voltage_ADC`
"Выходное напряжение на 2 выводе обмотки A" необработанные данные с АЦП.
- unsigned int `B1Voltage_ADC`
"Выходное напряжение на 1 выводе обмотки B" необработанные данные с АЦП.
- unsigned int `B2Voltage_ADC`
"Выходное напряжение на 2 выводе обмотки B" необработанные данные с АЦП.
- unsigned int `SupVoltage_ADC`
"Напряжение питания ключей H-моста" необработанные данные с АЦП.
- unsigned int `ACurrent_ADC`
"Ток через обмотку A" необработанные данные с АЦП.
- unsigned int `BCurrent_ADC`
"Ток через обмотку B" необработанные данные с АЦП.
- unsigned int `FullCurrent_ADC`
"Полный ток" необработанные данные с АЦП.
- unsigned int `Temp_ADC`
Напряжение с датчика температуры, необработанные данные с АЦП.
- unsigned int `Joy_ADC`
Джойстик, необработанные данные с АЦП.

- unsigned int `Pot_ADC`
"Потенциометр" необработанные данные с АЦП
- unsigned int `L5_ADC`
Напряжение питания USB после current sense резистора, необработанные данные с АЦП.
- unsigned int `H5_ADC`
Напряжение питания USB, необработанные данные с АЦП
- int `A1Voltage`
"Выходное напряжение на 1 выводе обмотки А" откалиброванные данные.
- int `A2Voltage`
"Выходное напряжение на 2 выводе обмотки А" откалиброванные данные.
- int `B1Voltage`
"Выходное напряжение на 1 выводе обмотки В" откалиброванные данные.
- int `B2Voltage`
"Выходное напряжение на 2 выводе обмотки В" откалиброванные данные.
- int `SupVoltage`
"Напряжение питания ключей Н-моста" откалиброванные данные.
- int `ACurrent`
"Ток через обмотку А" откалиброванные данные.
- int `BCurrent`
"Ток через обмотку В" откалиброванные данные.
- int `FullCurrent`
"Полный ток" откалиброванные данные.
- int `Temp`
Температура, откалиброванные данные.
- int `Joy`
Джойстик во внутренних единицах [0, 10000].
- int `Pot`
Потенциометр во внутренних единицах [0, 10000].
- int `L5`
Напряжение питания USB после current sense резистора
- int `H5`
Напряжение питания USB.
- unsigned int deprecated
- int `R`
Сопротивление обмоток двигателя(для шагового двигателя), в мОм
- int `L`
Псевдоиндуктивность обмоток двигателя(для шагового двигателя), в мкГн

4.4.1 Подробное описание

Аналоговые данные.

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

См. также

`get_analog_data`
`get_analog_data`

4.5 Структура `brake_settings_t`

Настройки тормоза.

Поля данных

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

4.5.1 Подробное описание

Настройки тормоза.

Эта структура содержит параметры управления тормозом.

См. также

[set_brake_settings](#)
[get_brake_settings](#)
[get_brake_settings, set_brake_settings](#)

4.5.2 Поля

4.5.2.1 unsigned int t1

Время в мс между включением питания мотора и отключением тормоза.

Диапазон: 0..65535.

4.5.2.2 unsigned int t2

Время в мс между отключением тормоза и готовностью к движению.

Все команды движения начинают выполняться только по истечении этого времени. Диапазон: 0..65535.

4.5.2.3 unsigned int t3

Время в мс между остановкой мотора и включением тормоза.

Диапазон: 0..65535.

4.5.2.4 unsigned int t4

Время в мс между включением тормоза и отключением питания мотора.

Диапазон: 0..65535.

4.6 Структура calibration_t

Calibration companion structure TODO docme.

Поля данных

- `double A`
Multiplier.
- `unsigned int MicrostepMode`
Microstep mode.

4.6.1 Подробное описание

Calibration companion structure TODO docme.

4.7 Структура `chart_data_t`

Дополнительное состояние устройства.

Поля данных

- `int WindingVoltageA`
В случае ШД, напряжение на обмотке А; в случае бесщеточного, напряжение на первой обмотке; в случае DC на единственной.
- `int WindingVoltageB`
В случае ШД, напряжение на обмотке В; в случае бесщеточного, напряжение на второй обмотке; в случае DC не используется.
- `int WindingVoltageC`
В случае бесщеточного, напряжение на третьей обмотке; в случае ШД и DC не используется.
- `int WindingCurrentA`
В случае ШД, ток в обмотке А; в случае бесщеточного, ток в первой обмотке; в случае DC в единственной.
- `int WindingCurrentB`
В случае ШД, ток в обмотке В; в случае бесщеточного, ток в второй обмотке; в случае DC не используется.
- `int WindingCurrentC`
В случае бесщеточного, ток в третьей обмотке; в случае ШД и DC не используется.
- `unsigned int Pot`
Положение потенциометра в десяти тысячных долях [0, 10000].
- `unsigned int Joy`
Положение джойстика в десяти тысячных долях [0, 10000].
- `int DutyCycle`
Коэффициент заполнения ШИМ.

4.7.1 Подробное описание

Дополнительное состояние устройства.

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

См. также

`get_chart_data`
`get_chart_data`

4.8 Структура control_settings_calb_t

Поля данных

- float [MaxSpeed](#) [10]
Массив скоростей, использующийся при управлении джойстиком или кнопками влево/вправо.
- unsigned int [Timeout](#) [9]
timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).
- unsigned int [MaxClickTime](#)
Максимальное время клика.
- unsigned int [Flags](#)
[Флаги управления.](#)
- float [DeltaPosition](#)
Смещение (дельта) позиции

4.8.1 Поля

4.8.1.1 unsigned int MaxClickTime

Максимальное время клика.

До истечения этого времени первая скорость не включается.

4.8.1.2 unsigned int Timeout[9]

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

Диапазон: 0..65535.

4.9 Структура control_settings_t

Настройки управления.

Поля данных

- unsigned int [MaxSpeed](#) [10]
Массив скоростей (в полных шагах), использующийся при управлении джойстиком или кнопками влево/вправо.
- unsigned int [uMaxSpeed](#) [10]
Массив скоростей (в 1/256 микрошагах), использующийся при управлении джойстиком или кнопками влево/вправо.
- unsigned int [Timeout](#) [9]
timeout[i] - время в мс, по истечении которого устанавливается скорость max_speed[i+1] (используется только при управлении кнопками).
- unsigned int [MaxClickTime](#)
Максимальное время клика.
- unsigned int [Flags](#)
[Флаги управления.](#)
- int [DeltaPosition](#)
Смещение (дельта) позиции
- int [uDeltaPosition](#)
Дробная часть смещения в микрошагах (-255..255) используется только с шаговым двигателем

4.9.1 Подробное описание

Настройки управления.

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

См. также

```
set_control_settings  
get_control_settings  
get_control_settings, set_control_settings
```

4.9.2 Поля

4.9.2.1 `unsigned int MaxClickTime`

Максимальное время клика.

До истечения этого времени первая скорость не включается.

4.9.2.2 `unsigned int MaxSpeed[10]`

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

Диапазон: 0..1000000.

4.9.2.3 `unsigned int Timeout[9]`

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

Диапазон: 0..65535.

4.9.2.4 `unsigned int uMaxSpeed[10]`

Массив скоростей (в 1/256 микрошагах), использующийся при управлении джойстиком или кнопками влево/вправо.

Диапазон: 0..255.

4.10 Структура `controller_name_t`

Пользовательское имя контроллера и флаги настройки.

Поля данных

- `char ControllerName[17]`
Пользовательское имя контроллера.
- `unsigned int CtrlFlags`
Флаги настроек контроллера.

4.10.1 Подробное описание

Пользовательское имя контроллера и флаги настройки.

См. также

[get_controller_name](#), [set_controller_name](#)

4.10.2 Поля

4.10.2.1 `char ControllerName[17]`

Пользовательское имя контроллера.

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

4.11 Структура `ctp_settings_t`

Настройки контроля позиции(для шагового двигателя).

Поля данных

- unsigned int [CTPMinError](#)
Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг `STATE_RT_ERROR`.
- unsigned int [CTPFlags](#)
[Флаги контроля позиции.](#)

4.11.1 Подробное описание

Настройки контроля позиции(для шагового двигателя).

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

См. также

[set_ctp_settings](#)
[get_ctp_settings](#)
[get_ctp_settings](#), [set_ctp_settings](#)

4.11.2 Поля

4.11.2.1 unsigned int `CTPMinError`

Минимальное отличие шагов ШД от положения энкодера, устанавливающее флаг `STATE_RT_ERROR`.

Измеряется в шагах ШД. Диапазон: 0..255.

4.12 Структура debug_read_t

Отладочные данные.

Поля данных

- unsigned int [DebugData](#) [128]
Отладочные данные.

4.12.1 Подробное описание

Отладочные данные.

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

См. также

[get_debug_read](#)

4.13 Структура device_information_t

Команда чтения информации о контроллере.

Поля данных

- char [Manufacturer](#) [5]
Производитель
- char [ManufacturerId](#) [3]
Идентификатор производителя
- char [ProductDescription](#) [9]
Описание продукта

4.13.1 Подробное описание

Команда чтения информации о контроллере.

Контроллер отвечает на эту команду в любом состоянии. Поле [Manufacturer](#) для всех XI** девайсов должно содержать строку "XIMC" (по нему производится валидация). Остальные поля содержат информацию об устройстве.

См. также

[get_device_information](#)
[get_device_information_impl](#)

4.14 Структура edges_settings_calb_t

Поля данных

- unsigned int [BorderFlags](#)

- Флаги границ.
 - unsigned int [EnderFlags](#)
 - Флаги концевых выключателей.
 - float [LeftBorder](#)
 - Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.
 - float [RightBorder](#)
 - Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.

4.15 Структура edges_settings_t

Настройки границ.

Поля данных

- unsigned int [BorderFlags](#)
 - Флаги границ.
- unsigned int [EnderFlags](#)
 - Флаги концевых выключателей.
- int [LeftBorder](#)
 - Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.
- int [uLeftBorder](#)
 - Позиция левой границы в 1/256 микрошагах(используется только с шаговым двигателем).
- int [RightBorder](#)
 - Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.
- int [uRightBorder](#)
 - Позиция правой границы в 1/256 микрошагах(используется только с шаговым двигателем).

4.15.1 Подробное описание

Настройки границ.

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

См. также

```
set_edges_settings
get_edges_settings
get_edges_settings, set_edges_settings
```

4.15.2 Поля

4.15.2.1 int LeftBorder

Позиция левой границы, используется если установлен флаг BORDER_IS_ENCODER.

Диапазон: -2147483647..2147483647.

4.15.2.2 int RightBorder

Позиция правой границы, используется если установлен флаг BORDER_IS_ENCODER.

Диапазон: -2147483647..2147483647.

4.15.2.3 `int uLeftBorder`

Позиция левой границы в 1/256 микрошагах(используется только с шаговым двигателем).

Диапазон: -255..255.

4.15.2.4 `int uRightBorder`

Позиция правой границы в 1/256 микрошагах(используется только с шаговым двигателем).

Диапазон: -255..255.

4.16 Структура `encoder_information_t`

Информация об энкодере.

Поля данных

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

4.16.1 Подробное описание

Информация об энкодере.

См. также

```
set_encoder_information  
get_encoder_information  
get_encoder_information, set_encoder_information
```

4.16.2 Поля

4.16.2.1 `char Manufacturer[17]`

Производитель.

Максимальная длина строки: 16 символов.

4.16.2.2 `char PartNumber[25]`

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.17 Структура `encoder_settings_t`

Настройки энкодера.

Поля данных

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

4.17.1 Подробное описание

Настройки энкодера.

См. также

```
set_encoder_settings  
get_encoder_settings  
get_encoder_settings, set_encoder_settings
```

4.17.2 Поля

4.17.2.1 float MaxCurrentConsumption

Максимальное потребление тока (мА).

Тип данных: float.

4.17.2.2 float MaxOperatingFrequency

Максимальная частота (кГц).

Тип данных: float.

4.17.2.3 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

4.17.2.4 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

4.18 Структура engine_settings_calb_t

Поля данных

- unsigned int [NomVoltage](#)
Номинальное напряжение мотора.
- unsigned int [NomCurrent](#)
Номинальный ток через мотор.
- float [NomSpeed](#)
Номинальная скорость.
- unsigned int [EngineFlags](#)
Флаги параметров мотора.
- float [Antiplay](#)
Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны.
- unsigned int [MicrostepMode](#)
Флаги параметров микрошагового режима.
- unsigned int [StepsPerRev](#)
Количество полных шагов на оборот(используется только с шаговым двигателем).

4.18.1 Поля

4.18.1.1 float Antiplay

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

Используется, если установлен флаг ENGINE_ANTIPLAY.

4.18.1.2 unsigned int NomCurrent

Номинальный ток через мотор.

Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE_LIMIT_CURR). Диапазон: 1..65535

4.18.1.3 float NomSpeed

Номинальная скорость.

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

4.18.1.4 unsigned int NomVoltage

Номинальное напряжение мотора.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT(используется только с DC двигателем). Диапазон: 1..65535

4.18.1.5 unsigned int StepsPerRev

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

Диапазон: 1..65535.

4.19 Структура engine_settings_t

Настройки мотора.

Поля данных

- unsigned int [NomVoltage](#)
Номинальное напряжение мотора.
- unsigned int [NomCurrent](#)
Номинальный ток через мотор.
- unsigned int [NomSpeed](#)
Номинальная скорость (в целых шагах/с или грм для DC и шагового двигателя в режиме ведущего энкодера).
- unsigned int [uNomSpeed](#)
Микрошаговая часть номинальной скорости мотора (используется только с шаговым двигателем).
- unsigned int [EngineFlags](#)
Флаги параметров мотора.
- int [Antiplay](#)
Количество шагов двигателя или импульсов энкодера, на которое позиционер будет отъезжать от заданной позиции для подхода к ней с одной и той же стороны.
- unsigned int [MicrostepMode](#)
Флаги параметров микрошагового режима.
- unsigned int [StepsPerRev](#)
Количество полных шагов на оборот(используется только с шаговым двигателем).

4.19.1 Подробное описание

Настройки мотора.

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

См. также

[set_engine_settings](#)
[get_engine_settings](#)
[get_engine_settings](#), [set_engine_settings](#)

4.19.2 Поля

4.19.2.1 int Antiplay

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

Используется, если установлен флаг ENGINE_ANTIPLAY. Диапазон: -32768..32767

4.19.2.2 unsigned int NomCurrent

Номинальный ток через мотор.

Ток стабилизируется для шаговых и может быть ограничен для DC(если установлен флаг ENGINE_LIMIT_CURR). Диапазон: 1..65535

4.19.2.3 unsigned int NomSpeed

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

Контроллер будет сохранять скорость мотора не выше номинальной, если установлен флаг ENGINE_LIMIT_RPM. Диапазон: 1..1000000.

4.19.2.4 unsigned int NomVoltage

Номинальное напряжение мотора.

Контроллер будет сохранять напряжение на моторе не выше номинального, если установлен флаг ENGINE_LIMIT_VOLT(используется только с DC двигателем). Диапазон: 1..65535

4.19.2.5 unsigned int StepsPerRev

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

Диапазон: 1..65535.

4.19.2.6 unsigned int uNomSpeed

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

Диапазон: 0..255.

4.20 Структура entype_settings_t

Настройки типа мотора и типа силового драйвера.

Поля данных

- unsigned int [EngineType](#)
Флаги, определяющие тип мотора.
- unsigned int [DriverType](#)
Флаги, определяющие тип силового драйвера.

4.20.1 Подробное описание

Настройки типа мотора и типа силового драйвера.

Эта структура содержит настройки типа мотора и типа силового драйвера.

Аргументы

id	идентификатор устройства
EngineType	тип мотора
DriverType	тип силового драйвера

См. также

[get_entype_settings](#), [set_entype_settings](#)

4.21 Структура `extio_settings_t`

Настройки EXTIO.

Поля данных

- unsigned int [EXTIOSetupFlags](#)
Флаги настройки работы внешнего ввода/вывода.
- unsigned int [EXTIOModeFlags](#)
Флаги настройки режимов внешнего ввода/вывода.

4.21.1 Подробное описание

Настройки EXTIO.

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

См. также

[get_extio_settings](#)
[set_extio_settings](#)
[get_extio_settings](#), [set_extio_settings](#)

4.22 Структура `feedback_settings_t`

Настройки обратной связи.

Поля данных

- unsigned int [IPS](#)
Количество измеряемых отсчётов энкодера на оборот
- unsigned int [FeedbackType](#)
Тип обратной связи.
- unsigned int [FeedbackFlags](#)
Флаги обратной связи.
- unsigned int [HallSPR](#)
Количество отсчётов датчиков Холла на оборот.
- int [HallShift](#)
Фазовый сдвиг между выходным сигналом на обмотках BLDC двигателя и входным сигналом на датчиках Холла(0 - при активном только датчике холла А подается положительный потенциал на обмотку А и отрицательный потенциал на обмотку В).

4.22.1 Подробное описание

Настройки обратной связи.

Эта структура содержит настройки обратной связи.

См. также

[get_feedback_settings](#), [set_feedback_settings](#)

4.23 Структура gear_information_t

Информация о редукторе.

Поля данных

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

4.23.1 Подробное описание

Информация о редукторе.

См. также

[set_gear_information](#)
[get_gear_information](#)
[get_gear_information](#), [set_gear_information](#)

4.23.2 Поля

4.23.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

4.23.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.24 Структура gear_settings_t

Настройки редуктора.

Поля данных

- float [ReductionIn](#)
Входной коэффициент редуктора.

- float [ReductionOut](#)
Выходной коэффициент редуктора.
- float [RatedInputTorque](#)
Максимальный крутящий момент (Н м).
- float [RatedInputSpeed](#)
Максимальная скорость на входном валу редуктора (об/мин).
- float [MaxOutputBacklash](#)
Выходной люфт редуктора (градус).
- float [InputInertia](#)
Эквивалентная входная инерция редуктора(г см²).
- float [Efficiency](#)
КПД редуктора (%).

4.24.1 Подробное описание

Настройки редуктора.

См. также

[set_gear_settings](#)
[get_gear_settings](#)
[get_gear_settings](#), [set_gear_settings](#)

4.24.2 Поля

4.24.2.1 float Efficiency

КПД редуктора (%).

Тип данных: float.

4.24.2.2 float InputInertia

Эквивалентная входная инерция редуктора(г см²).

Тип данных: float.

4.24.2.3 float MaxOutputBacklash

Выходной люфт редуктора (градус).

Тип данных: float.

4.24.2.4 float RatedInputSpeed

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

Тип данных: float.

4.24.2.5 float RatedInputTorque

Максимальный крутящий момент (Н м).

Тип данных: float.

4.24.2.6 float `ReductionIn`

Входной коэффициент редуктора.

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

4.24.2.7 float `ReductionOut`

Выходной коэффициент редуктора.

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

4.25 Структура `get_position_calb_t`

Поля данных

- float [Position](#)
Позиция двигателя.
- long long [EncPosition](#)
Позиция энкодера.

4.26 Структура `get_position_t`

Данные о позиции.

Поля данных

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

4.26.1 Подробное описание

Данные о позиции.

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

См. также

[get_position](#)

4.27 Структура `hallsensor_information_t`

Информация об датчиках Холла.

Поля данных

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

4.27.1 Подробное описание

Информация об датчиках Холла.

См. также

[set_hallsensor_information](#)
[get_hallsensor_information](#)
[get_hallsensor_information, set_hallsensor_information](#)

4.27.2 Поля

4.27.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

4.27.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.28 Структура hallsensor_settings_t

Настройки датчиков Холла.

Поля данных

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

4.28.1 Подробное описание

Настройки датчиков Холла.

См. также

[set_hallsensor_settings](#)
[get_hallsensor_settings](#)
[get_hallsensor_settings](#), [set_hallsensor_settings](#)

4.28.2 Поля

4.28.2.1 float MaxCurrentConsumption

Максимальное потребление тока (мА).

Тип данных: float.

4.28.2.2 float MaxOperatingFrequency

Максимальная частота (кГц).

Тип данных: float.

4.28.2.3 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

4.28.2.4 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

4.29 Структура home_settings_calb_t

Поля данных

- float [FastHome](#)
Скорость первого движения.
- float [SlowHome](#)
Скорость второго движения.
- float [HomeDelta](#)
Расстояние отхода от точки останова.
- unsigned int [HomeFlags](#)
[Флаги настроек команды home.](#)

4.30 Структура home_settings_t

Настройки калибровки позиции.

Поля данных

- unsigned int [FastHome](#)
Скорость первого движения.

- unsigned int [uFastHome](#)
Дробная часть скорости первого движения в микрошагах(используется только с шаговым двигателем).
- unsigned int [SlowHome](#)
Скорость второго движения.
- unsigned int [uSlowHome](#)
Дробная часть скорости второго движения в микрошагах(используется только с шаговым двигателем).
- int [HomeDelta](#)
Расстояние отхода от точки останова.
- int [uHomeDelta](#)
Дробная часть расстояния отхода от точки останова в микрошагах(используется только с шаговым двигателем).
- unsigned int [HomeFlags](#)
Флаги настроек команды home.

4.30.1 Подробное описание

Настройки калибровки позиции.

Эта структура содержит настройки, используемые при калибровке позиции.

См. также

[get_home_settings](#)
[set_home_settings](#)
[command_home](#)
[get_home_settings](#), [set_home_settings](#)

4.30.2 Поля

4.30.2.1 unsigned int FastHome

Скорость первого движения.

Диапазон: 0..1000000

4.30.2.2 int HomeDelta

Расстояние отхода от точки останова.

Диапазон: -2147483647..2147483647.

4.30.2.3 unsigned int SlowHome

Скорость второго движения.

Диапазон: 0..1000000.

4.30.2.4 unsigned int uFastHome

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

Диапазон: 0..255.

4.30.2.5 int uHomeDelta

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

Диапазон: -255..255.

4.30.2.6 unsigned int uSlowHome

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

Диапазон: 0..255.

4.31 Структура joystick_settings_t

Настройки джойстика.

Поля данных

- unsigned int [JoyLowEnd](#)
Значение в шагах джойстика, соответствующее нижней границе диапазона отклонения устройства.
- unsigned int [JoyCenter](#)
Значение в шагах джойстика, соответствующее неотклонённому устройству.
- unsigned int [JoyHighEnd](#)
Значение в шагах джойстика, соответствующее верхней границе диапазона отклонения устройства.
- unsigned int [ExpFactor](#)
Фактор экспоненциальной нелинейности отклика джойстика.
- unsigned int [DeadZone](#)
Отклонение от среднего положения, которое не вызывает начала движения (в десятых долях процента).
- unsigned int [JoyFlags](#)
Флаги джойстика.

4.31.1 Подробное описание

Настройки джойстика.

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

См. также

[set_joystick_settings](#)
[get_joystick_settings](#)
[get_joystick_settings, set_joystick_settings](#)

4.31.2 Поля

4.31.2.1 `unsigned int DeadZone`

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

Максимальное мёртвое отклонение $\pm 25.5\%$, что составляет половину рабочего диапазона джойстика.

4.31.2.2 `unsigned int JoyCenter`

Значение в шагах джойстика, соответствующее неотклонённому устройству.

Должно лежать в пределах $[0, 10000]$.

4.31.2.3 `unsigned int JoyHighEnd`

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

Должно лежать в пределах $[0, 10000]$.

4.31.2.4 `unsigned int JoyLowEnd`

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

Должно лежать в пределах $[0, 10000]$.

4.32 Структура `motor_information_t`

Информация о двигателе.

Поля данных

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

4.32.1 Подробное описание

Информация о двигателе.

См. также

`set_motor_information`
`get_motor_information`
`get_motor_information, set_motor_information`

4.32.2 Поля

4.32.2.1 `char Manufacturer[17]`

Производитель.

Максимальная длина строки: 16 символов.

4.32.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.33 Структура motor_settings_t

Настройки двигателя.

Поля данных

- unsigned int [MotorType](#)
Флаг типа двигателя.
- unsigned int [ReservedField](#)
Зарезервировано
- unsigned int [Poles](#)
Кол-во пар полюсов у DC или BLDC двигателя или кол-во шагов на оборот для шагового двигателя.
- unsigned int [Phases](#)
Кол-во фаз у BLDC двигателя.
- float [NominalVoltage](#)
Номинальное напряжение на обмотке (В).
- float [NominalCurrent](#)
Максимальный постоянный ток в обмотке для DC и BLDC двигателей, номинальный ток в обмотке для шаговых двигателей (А).
- float [NominalSpeed](#)
Номинальная скорость (об/мин).
- float [NominalTorque](#)
Номинальный крутящий момент (мН м).
- float [NominalPower](#)
Номинальная мощность(Вт).
- float [WindingResistance](#)
Сопротивление обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (Ом).
- float [WindingInductance](#)
Индуктивность обмотки DC двигателя, каждой из двух обмоток шагового двигателя или каждой из трёх обмоток BLDC двигателя (мГн).
- float [RotorInertia](#)
Инерция ротора (г см²).
- float [StallTorque](#)
Крутящий момент удержания позиции для шагового двигателя или крутящий момент при неподвижном роторе для других типов двигателей (мН м).
- float [DetentTorque](#)
Момент удержания позиции с незапитанными обмотками (мН м).
- float [TorqueConstant](#)
Константа крутящего момента, определяющая коэффициент пропорциональности максимального момента силы ротора от протекающего в обмотке тока (мН м/А).
- float [SpeedConstant](#)
Константа скорости, определяющая значение или амплитуду напряжения наведённой индукции при вращении ротора DC или BLDC двигателя (об/мин / В) или шагового двигателя (шаг/с / В).
- float [SpeedTorqueGradient](#)

- Градиент крутящего момента (об/мин / мН м).
- float [MechanicalTimeConstant](#)
Механическая постоянная времени (мс).
- float [MaxSpeed](#)
Максимальная разрешённая скорость для шаговых двигателей (шаг/с) или для DC и BLDC двигателей (об/мин).
- float [MaxCurrent](#)
Максимальный ток в обмотке (А).
- float [MaxCurrentTime](#)
Безопасная длительность максимального тока в обмотке (мс).
- float [NoLoadCurrent](#)
Ток потребления в холостом режиме (А).
- float [NoLoadSpeed](#)
Скорость в холостом режиме (об/мин).

4.33.1 Подробное описание

Настройки двигателя.

См. также

```
set_motor_settings  
get_motor_settings  
get_motor_settings, set_motor_settings
```

4.33.2 Поля

4.33.2.1 float DetentTorque

Момент удержания позиции с незапитанными обмотками (мН м).

Тип данных: float.

4.33.2.2 float MaxCurrent

Максимальный ток в обмотке (А).

Тип данных: float.

4.33.2.3 float MaxCurrentTime

Безопасная длительность максимального тока в обмотке (мс).

Тип данных: float.

4.33.2.4 float MaxSpeed

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

Тип данных: float.

4.33.2.5 float MechanicalTimeConstant

Механическая постоянная времени (мс).

Тип данных: float.

4.33.2.6 float NoLoadCurrent

Ток потребления в холостом режиме (А).

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

4.33.2.7 float NoLoadSpeed

Скорость в холостом режиме (об/мин).

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

4.33.2.8 float NominalCurrent

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

Тип данных: float.

4.33.2.9 float NominalPower

Номинальная мощность(Вт).

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

4.33.2.10 float NominalSpeed

Номинальная скорость (об/мин).

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

4.33.2.11 float NominalTorque

Номинальный крутящий момент (мН м).

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

4.33.2.12 float NominalVoltage

Номинальное напряжение на обмотке (В).

Тип данных: float.

4.33.2.13 float RotorInertia

Инерция ротора (г см²).

Тип данных: float.

4.33.2.14 float SpeedConstant

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

Тип данных: float.

4.33.2.15 float SpeedTorqueGradient

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

Тип данных: float.

4.33.2.16 float StallTorque

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

Тип данных: float.

4.33.2.17 float TorqueConstant

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

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

4.33.2.18 float WindingInductance

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

Тип данных: float.

4.33.2.19 float WindingResistance

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

Тип данных: float.

4.34 Структура move_settings_calb_t

Поля данных

- float [Speed](#)
Заданная скорость.
- float [Accel](#)
Ускорение, заданное в шагах в секунду²(Ш/Д) или в оборотах в минуту за секунду(DC).
- float [Decel](#)
Торможение, заданное в шагах в секунду²(Ш/Д) или в оборотах в минуту за секунду(DC).
- float [AntiplaySpeed](#)
Скорость в режиме антилюфта.

4.35 Структура move_settings_t

Настройки движения.

Поля данных

- unsigned int [Speed](#)
Заданная скорость (для ШД: шагов/с, для DC: rpm).
- unsigned int [uSpeed](#)
Заданная скорость в 1/256 микрошагах в секунду.
- unsigned int [Accel](#)
Ускорение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).
- unsigned int [Decel](#)
Торможение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).
- unsigned int [AntiplaySpeed](#)
Скорость в режиме антилюфта, заданная в целых шагах/с(ШД) или в оборотах/с(DC).
- unsigned int [uAntiplaySpeed](#)
Скорость в режиме антилюфта, выраженная в 1/256 микрошагах в секунду.

4.35.1 Подробное описание

Настройки движения.

См. также

[set_move_settings](#)
[get_move_settings](#)
[get_move_settings, set_move_settings](#)

4.35.2 Поля

4.35.2.1 unsigned int Accel

Ускорение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).

Диапазон: 0..65535.

4.35.2.2 unsigned int AntiplaySpeed

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

Диапазон: 0..1000000.

4.35.2.3 unsigned int Decel

Торможение, заданное в шагах в секунду²(ШД) или в оборотах в минуту за секунду(DC).

Диапазон: 0..65535.

4.35.2.4 unsigned int Speed

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

Диапазон: 0..1000000.

4.35.2.5 unsigned int uAntiplaySpeed

Скорость в режиме антилюфта, выраженная в 1/256 микрошагах в секунду.

Используется только с шаговым мотором. Диапазон: 0..255.

4.35.2.6 unsigned int uSpeed

Заданная скорость в 1/256 микрошагах в секунду.

Используется только с шаговым мотором. Диапазон: 0..255.

4.36 Структура pid_settings_t

Настройки ПИД.

Поля данных

- unsigned int [KpU](#)
Пропорциональный коэффициент ПИД контура по напряжению
- unsigned int [KiU](#)
Интегральный коэффициент ПИД контура по напряжению
- unsigned int [KdU](#)
Дифференциальный коэффициент ПИД контура по напряжению

4.36.1 Подробное описание

Настройки ПИД.

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

См. также

[set_pid_settings](#)
[get_pid_settings](#)
[get_pid_settings, set_pid_settings](#)

4.37 Структура power_settings_t

Настройки питания шагового мотора.

Поля данных

- unsigned int [HoldCurrent](#)
Ток мотора в режиме удержания, в процентах от номинального.
- unsigned int [CurrReductDelay](#)
Время в мс от перехода в состояние STOP до уменьшения тока.
- unsigned int [PowerOffDelay](#)
Время в с от перехода в состояние STOP до отключения питания мотора.
- unsigned int [CurrentSetTime](#)
Время в мс, требуемое для набора номинального тока от 0% до 100%.
- unsigned int [PowerFlags](#)
[Флаги параметров питания шагового мотора.](#)

4.37.1 Подробное описание

Настройки питания шагового мотора.

См. также

```
set_move_settings  
get_move_settings  
get_power_settings, set_power_settings
```

4.37.2 Поля

4.37.2.1 `unsigned int CurrentSetTime`

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

Диапазон: 0..65535.

4.37.2.2 `unsigned int CurrReductDelay`

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

Диапазон: 0..65535.

4.37.2.3 `unsigned int HoldCurrent`

Ток мотора в режиме удержания, в процентах от номинального.

Диапазон: 0..100.

4.37.2.4 `unsigned int PowerOffDelay`

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

Диапазон: 0..65535.

4.38 Структура `secure_settings_t`

Эта структура содержит необработанные данные с АЦП и нормированные значения.

Поля данных

- `unsigned int LowUpwrOff`
Нижний порог напряжения на силовой части для выключения, в мВ.
- `unsigned int CriticalIpwr`
Максимальный ток силовой части, вызывающий состояние ALARM, в мА.
- `unsigned int CriticalUpwr`
Максимальное напряжение на силовой части, вызывающее состояние ALARM, в мВ.
- `unsigned int CriticalT`
Максимальная температура контроллера, вызывающая состояние ALARM, в десятых долях градуса Цельсия.
- `unsigned int CriticalIusb`
Максимальный ток USB, вызывающий состояние ALARM, в мА.
- `unsigned int CriticalUusb`

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

- unsigned int `MinimumUusb`

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

- unsigned int `Flags`

Флаги критических параметров.

4.38.1 Подробное описание

Эта структура содержит необработанные данные с АЦП и нормированные значения.

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

См. также

`get_secure_settings`
`set_secure_settings`
`get_secure_settings, set_secure_settings`

4.38.2 Поля

4.38.2.1 unsigned int `CriticalIpwr`

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

Диапазон: 0..65535.

4.38.2.2 unsigned int `CriticalIusb`

Максимальный ток USB, вызывающий состояние ALARM, в мА.

Диапазон: 0..65535.

4.38.2.3 unsigned int `CriticalT`

Максимальная температура контроллера, вызывающая состояние ALARM, в десятых долях градуса Цельсия.

Диапазон: 0..65535.

4.38.2.4 unsigned int `CriticalUpwr`

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

Диапазон: 0..65535.

4.38.2.5 unsigned int `CriticalUusb`

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

Диапазон: 0..65535.

4.38.2.6 unsigned int `LowUpwrOff`

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

Диапазон: 0..65535.

4.38.2.7 `unsigned int MinimumUusb`

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

Диапазон: 0..65535.

4.39 Структура `serial_number_t`

Структура с серийным номером.

Поля данных

- `unsigned int SN`
Новый серийный номер платы.
- `unsigned int Key [32]`
Ключ защиты для установки серийного номера (256 бит).

4.39.1 Подробное описание

Структура с серийным номером.

Вместе с новым серийным номером передаётся "Ключ", только при совпадении которого происходит изменение и сохранение серийного номера. Функция используется только производителем.

См. также

[set_serial_number](#)

4.40 Структура `set_position_calb_t`

Поля данных

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

4.41 Структура `set_position_t`

Данные о позиции.

Поля данных

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

- unsigned int [PosFlags](#)
Флаги установки положения.

4.41.1 Подробное описание

Данные о позиции.

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

См. также

[set_position](#)

4.42 Структура stage_information_t

Информация о позиционере.

Поля данных

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

4.42.1 Подробное описание

Информация о позиционере.

См. также

[set_stage_information](#)
[get_stage_information](#)
[get_stage_information](#), [set_stage_information](#)

4.42.2 Поля

4.42.2.1 char Manufacturer[17]

Производитель.

Максимальная длина строки: 16 символов.

4.42.2.2 char PartNumber[25]

Серия и номер модели.

Максимальная длина строки: 24 символа.

4.43 Структура stage_name_t

Пользовательское имя подвижки.

Поля данных

- char [PositionerName](#) [17]
Пользовательское имя подвижки.

4.43.1 Подробное описание

Пользовательское имя подвижки.

См. также

[get_stage_name](#), [set_stage_name](#)

4.43.2 Поля

4.43.2.1 char PositionerName[17]

Пользовательское имя подвижки.

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

4.44 Структура stage_settings_t

Настройки позиционера.

Поля данных

- float [LeadScrewPitch](#)
Шаг ходового винта в мм.
- char [Units](#) [9]
Единицы измерения расстояния, используемые в полях MaxSpeed и TravelRange (шаги, градусы, мм, ...), Максимальная длина строки: 8 символов.
- float [MaxSpeed](#)
Максимальная скорость (Units/c).
- float [TravelRange](#)
Диапазон перемещения (Units).
- float [SupplyVoltageMin](#)
Минимальное напряжение питания (В).
- float [SupplyVoltageMax](#)
Максимальное напряжение питания (В).
- float [MaxCurrentConsumption](#)
Максимальный ток потребления (А).
- float [HorizontalLoadCapacity](#)
Горизонтальная грузоподъемность (кг).
- float [VerticalLoadCapacity](#)
Вертикальная грузоподъемность (кг).

4.44.1 Подробное описание

Настройки позиционера.

См. также

[set_stage_settings](#)
[get_stage_settings](#)
[get_stage_settings](#), [set_stage_settings](#)

4.44.2 Поля

4.44.2.1 float HorizontalLoadCapacity

Горизонтальная грузоподъемность (кг).

Тип данных: float.

4.44.2.2 float LeadScrewPitch

Шаг ходового винта в мм.

Тип данных: float.

4.44.2.3 float MaxCurrentConsumption

Максимальный ток потребления (А).

Тип данных: float.

4.44.2.4 float MaxSpeed

Максимальная скорость (Units/c).

Тип данных: float.

4.44.2.5 float SupplyVoltageMax

Максимальное напряжение питания (В).

Тип данных: float.

4.44.2.6 float SupplyVoltageMin

Минимальное напряжение питания (В).

Тип данных: float.

4.44.2.7 float TravelRange

Диапазон перемещения (Units).

Тип данных: float.

4.44.2.8 float VerticalLoadCapacity

Вертикальная грузоподъемность (кг).

Тип данных: float.

4.45 Структура status_calb_t

Поля данных

- unsigned int [MoveSts](#)
Флаги состояния движения.
- unsigned int [MvCmdSts](#)
Состояние команды движения.
- unsigned int [PWRSts](#)
Флаги состояния питания шагового мотора.
- unsigned int [EncSts](#)
Состояние энкодера.
- unsigned int [WindSts](#)
Состояние обмоток.
- float [CurPosition](#)
Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.
- long long [EncPosition](#)
Текущая позиция по данным с энкодера в импульсах энкодера, используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзывания.
- float [CurSpeed](#)
Текущая скорость.
- int [Ipwr](#)
Ток потребления силовой части.
- int [Upwr](#)
Напряжение на силовой части.
- int [Iusb](#)
Ток потребления по USB.
- int [Uusb](#)
Напряжение на USB.
- int [CurT](#)
Температура процессора в десятых долях градусов цельсия.
- unsigned int [Flags](#)
Флаги состояния.
- unsigned int [GPIOFlags](#)
Флаги состояния GPIO входов.
- unsigned int [CmdBufFreeSpace](#)
Это поле показывает количество свободных ячеек буфера цепочки синхронизации.

4.45.1 Поля

4.45.1.1 float CurPosition

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

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

4.46 Структура status_t

Состояние устройства.

Поля данных

- unsigned int [MoveSts](#)
Флаги состояния движения.
- unsigned int [MvCmdSts](#)
Состояние команды движения.
- unsigned int [PWRSts](#)
Флаги состояния питания шагового мотора.
- unsigned int [EncSts](#)
Состояние энкодера.
- unsigned int [WindSts](#)
Состояние обмоток.
- int [CurPosition](#)
Первичное поле, в котором хранится текущая позиция, как бы ни была устроена обратная связь.
- int [uCurPosition](#)
Дробная часть текущей позиции в микрошагах (-255..255).
- long long [EncPosition](#)
Текущая позиция по данным с энкодера в импульсах энкодера, используется только если энкодер установлен, активизирован и не является основным датчиком положения, например при использовании энкодера совместно с шаговым двигателем для контроля проскальзывания.
- int [CurSpeed](#)
Текущая скорость.
- int [uCurSpeed](#)
Дробная часть текущей скорости в микрошагах (-255..255).
- int [Ipwr](#)
Ток потребления силовой части.
- int [Upwr](#)
Напряжение на силовой части.
- int [Iusb](#)
Ток потребления по USB.
- int [Uusb](#)
Напряжение на USB.
- int [CurT](#)
Температура процессора в десятых долях градусов цельсия.
- unsigned int [Flags](#)
Флаги состояния.
- unsigned int [GPIOFlags](#)
Флаги состояния GPIO входов.
- unsigned int [CmdBufFreeSpace](#)
Это поле показывает количество свободных ячеек буфера цепочки синхронизации.

4.46.1 Подробное описание

Состояние устройства.

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

См. также

`get_status_impl`

4.46.2 Поля

4.46.2.1 `int CurPosition`

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

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

4.46.2.2 `int uCurPosition`

Дробная часть текущей позиции в микрошагах (-255..255).

Используется только с шаговым двигателем.

4.46.2.3 `int uCurSpeed`

Дробная часть текущей скорости в микрошагах (-255..255).

Используется только с шаговым двигателем.

4.47 Структура `sync_in_settings_calb_t`

Поля данных

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

4.47.1 Поля

4.47.1.1 unsigned int `ClutterTime`

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

Диапазон: 0..65535

4.48 Структура `sync_in_settings_t`

Настройки входной синхронизации.

Поля данных

- unsigned int [SyncInFlags](#)
Флаги настроек синхронизации входа.
- unsigned int [ClutterTime](#)
Минимальная длительность входного импульса синхронизации для защиты от дребезга (мкс).
- int [Position](#)
Желаемая позиция или смещение (целая часть)
- int [uPosition](#)
Дробная часть позиции или смещения в микрошагах (-255..255)(используется только с шаговым двигателем).
- unsigned int [Speed](#)
Заданная скорость (для ШД: шагов/с, для DC: rpm).
- unsigned int [uSpeed](#)
Заданная скорость в микрошагах в секунду.

4.48.1 Подробное описание

Настройки входной синхронизации.

Эта структура содержит все настройки, определяющие поведение входа синхронизации.

См. также

[get_sync_in_settings](#)
[set_sync_in_settings](#)
[get_sync_in_settings](#), [set_sync_in_settings](#)

4.48.2 Поля

4.48.2.1 unsigned int `ClutterTime`

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

Диапазон: 0..65535

4.48.2.2 unsigned int `Speed`

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

Диапазон: 0..1000000.

4.48.2.3 unsigned int `uSpeed`

Заданная скорость в микрошагах в секунду.

Используется только с шаговым мотором. Диапазон: 0..255.

4.49 Структура `sync_out_settings_calb_t`

Поля данных

- unsigned int [SyncOutFlags](#)
Флаги настроек синхронизации выхода.
- unsigned int [SyncOutPulseSteps](#)
Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг `SYNCOUT_IN_STEPS`, или в микросекундах если флаг сброшен.
- unsigned int [SyncOutPeriod](#)
Период генерации импульсов, используется при установленном флаге `SYNCOUT_ONPERIOD`.
- float [Accuracy](#)
Это окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.

4.49.1 Поля

4.49.1.1 unsigned int `SyncOutPeriod`

Период генерации импульсов, используется при установленном флаге `SYNCOUT_ONPERIOD`.

Диапазон: 0..65535

4.49.1.2 unsigned int `SyncOutPulseSteps`

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

Диапазон: 0..65535

4.50 Структура `sync_out_settings_t`

Настройки выходной синхронизации.

Поля данных

- unsigned int [SyncOutFlags](#)
Флаги настроек синхронизации выхода.
- unsigned int [SyncOutPulseSteps](#)
Определяет длительность выходных импульсов в шагах/импульсах энкодера, когда установлен флаг `SYNCOUT_IN_STEPS`, или в микросекундах если флаг сброшен.
- unsigned int [SyncOutPeriod](#)
Период генерации импульсов, используется при установленном флаге `SYNCOUT_ONPERIOD`.
- unsigned int [Accuracy](#)
Это окрестность вокруг целевой координаты, попадание в которую считается попаданием в целевую позицию и генерируется импульс по остановке.
- unsigned int [uAccuracy](#)
Это окрестность вокруг целевой координаты в микрошагах (используется только с шаговым двигателем).

4.50.1 Подробное описание

Настройки выходной синхронизации.

Эта структура содержит все настройки, определяющие поведение выхода синхронизации.

См. также

[get_sync_out_settings](#)
[set_sync_out_settings](#)
[get_sync_out_settings](#), [set_sync_out_settings](#)

4.50.2 Поля

4.50.2.1 unsigned int Accuracy

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

Диапазон: 0..4294967295.

4.50.2.2 unsigned int SyncOutPeriod

Период генерации импульсов, используется при установленном флаге `SYNCOUT_ONPERIOD`.

Диапазон: 0..65535

4.50.2.3 unsigned int SyncOutPulseSteps

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

Диапазон: 0..65535

4.50.2.4 unsigned int uAccuracy

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

Диапазон: 0..255.

4.51 Структура `uart_settings_t`

Настройки UART.

Поля данных

- unsigned int [Speed](#)
Скорость UART.
- unsigned int [UARTSetupFlags](#)
Флаги настроек четности команды `uart`.

4.51.1 Подробное описание

Настройки UART.

Эта структура содержит настройки UART.

См. также

- [get_uart_settings](#)
- [set_uart_settings](#)
- [get_uart_settings, set_uart_settings](#)

Глава 5

Файлы

5.1 Файл `ximc.h`

Заголовочный файл для библиотеки `libximc`.

Структуры данных

- struct `calibration_t`
Calibration companion structure TODO docme.
- struct `feedback_settings_t`
Настройки обратной связи.
- struct `home_settings_t`
Настройки калибровки позиции.
- struct `home_settings_calb_t`
- struct `move_settings_t`
Настройки движения.
- struct `move_settings_calb_t`
- struct `engine_settings_t`
Настройки мотора.
- struct `engine_settings_calb_t`
- struct `entype_settings_t`
Настройки типа мотора и типа силового драйвера.
- struct `power_settings_t`
Настройки питания шагового мотора.
- struct `secure_settings_t`
Эта структура содержит необработанные данные с АЦП и нормированные значения.
- struct `edges_settings_t`
Настройки границ.
- struct `edges_settings_calb_t`
- struct `pid_settings_t`
Настройки ПИД.
- struct `sync_in_settings_t`
Настройки входной синхронизации.
- struct `sync_in_settings_calb_t`
- struct `sync_out_settings_t`
Настройки выходной синхронизации.
- struct `sync_out_settings_calb_t`
- struct `extio_settings_t`

- Настройки EXTIO.
 - struct [brake_settings_t](#)
- Настройки тормоза.
 - struct [control_settings_t](#)
- Настройки управления.
 - struct [control_settings_calb_t](#)
 - struct [joystick_settings_t](#)
- Настройки джойстика.
 - struct [ctp_settings_t](#)
- Настройки контроля позиции(для шагового двигателя).
 - struct [uart_settings_t](#)
- Настройки UART.
 - struct [controller_name_t](#)
- Пользовательское имя контроллера и флаги настройки.
 - struct [add_sync_in_action_t](#)
- Это команда добавляет один элемент в буфер FIFO команд.
 - struct [add_sync_in_action_calb_t](#)
 - struct [get_position_t](#)
- Данные о позиции.
 - struct [get_position_calb_t](#)
 - struct [set_position_t](#)
- Данные о позиции.
 - struct [set_position_calb_t](#)
 - struct [status_t](#)
- Состояние устройства.
 - struct [status_calb_t](#)
 - struct [chart_data_t](#)
- Дополнительное состояние устройства.
 - struct [device_information_t](#)
- Команда чтения информации о контроллере.
 - struct [serial_number_t](#)
- Структура с серийным номером.
 - struct [analog_data_t](#)
- Аналоговые данные.
 - struct [debug_read_t](#)
- Отладочные данные.
 - struct [stage_name_t](#)
- Пользовательское имя подвижки.
 - struct [stage_information_t](#)
- Информация о позиционере.
 - struct [stage_settings_t](#)
- Настройки позиционера.
 - struct [motor_information_t](#)
- Информация о двигателе.
 - struct [motor_settings_t](#)
- Настройки двигателя.
 - struct [encoder_information_t](#)
- Информация об энкодере.
 - struct [encoder_settings_t](#)
- Настройки энкодера.
 - struct [hallsensor_information_t](#)

- Информация об датчиках Холла.
• struct [hallsensor_settings_t](#)
Настройки датчиков Холла.
- struct [gear_information_t](#)
Информация о редукторе.
- struct [gear_settings_t](#)
Настройки редуктора.
- struct [accessories_settings_t](#)
Информация о дополнительных аксессуарах.

Макросы

- #define [XIMC_API](#)
Library import macro Macros allows to automatically import function from shared library.
- #define [XIMC_CALLCONV](#)
Library calling convention macros.
- #define [device_undefined](#) -1
Макрос, означающий неопределенное устройство

Результаты выполнения команд

- #define [result_ok](#) 0
выполнено успешно
- #define [result_error](#) -1
общая ошибка
- #define [result_not_implemented](#) -2
функция не определена
- #define [result_value_error](#) -3
ошибка записи значения
- #define [result_nodevice](#) -4
устройство не подключено

Уровень логирования

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

Флаги поиска устройств

- #define [ENUMERATE_PROBE](#) 0x01
Проверять, является ли устройство XIMC-совместимым.
- #define [ENUMERATE_ALL_COM](#) 0x02
Проверять все COM-устройства

Флаги состояния движения

Возвращаются командой `get_status`.

См. также

```
get_status
status_t::move_state
status_t::MoveSts, get_status_impl
```

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

Флаги настроек контроллера

См. также

```
set_controller_name
get_controller_name
controller_name_t::CtrlFlags, get_controller_name, set_controller_name
```

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

Флаги состояния питания шагового мотора

Возвращаются командой `get_status`.

См. также

```
status_t::power_state
get_status
status_t::PWRSts, get_status_impl
```

- #define **PWR_STATE_UNKNOWN** 0x00
Неизвестное состояние, которое не должно никогда реализовываться.
- #define **PWR_STATE_OFF** 0x01
Обмотки мотора разомкнуты и не управляются драйвером.
- #define **PWR_STATE_NORM** 0x03
Обмотки запитаны номинальным током.
- #define **PWR_STATE_REDUCT** 0x04
Обмотки намеренно запитаны уменьшенным током от рабочего для снижения потребляемой мощности.
- #define **PWR_STATE_MAX** 0x05
Обмотки запитаны максимально доступным током, который может выдать схема при данном напряжении питания.

Флаги состояния

Содержат бинарные значения состояния контроллера. Могут быть объединены с помощью логического ИЛИ.

См. также

```
status_t::flags
get_status
status_t::Flags, get_status_impl
```

- #define **STATE_CONTR** 0x0003F
Флаги состояния контроллера.
- #define **STATE_ERRC** 0x00001

- Недопустимая команда.
- `#define STATE_ERRD 0x00002`
- Нарушение целостности данных.
- `#define STATE_ERRV 0x00004`
- Недопустимое значение данных.
- `#define STATE_EEPROM_CONNECTED 0x00010`
- Подключена память EEPROM с настройками.
- `#define STATE_SECUR 0x3FFC0`
- Флаги опасности.
- `#define STATE_ALARM 0x00040`
- Контроллер находится в состоянии ALARM, показывая, что случилась какая-то опасная ситуация.
- `#define STATE_CTP_ERROR 0x00080`
- Контроль позиции нарушен (используется только с шаговым двигателем).
- `#define STATE_POWER_OVERHEAT 0x00100`
- Перегрелась силовая часть платы.
- `#define STATE_CONTROLLER_OVERHEAT 0x00200`
- Перегрелась микросхема контроллера.
- `#define STATE_OVERLOAD_POWER_VOLTAGE 0x00400`
- Превышено напряжение на силовой части.
- `#define STATE_OVERLOAD_POWER_CURRENT 0x00800`
- Превышен максимальный ток потребления силовой части.
- `#define STATE_OVERLOAD_USB_VOLTAGE 0x01000`
- Превышено напряжение на USB.
- `#define STATE_LOW_USB_VOLTAGE 0x02000`
- Слишком низкое напряжение на USB.
- `#define STATE_OVERLOAD_USB_CURRENT 0x04000`
- Превышен максимальный ток потребления USB.
- `#define STATE_BORDERS_SWAP_MISSET 0x08000`
- Достижение неверной границы.
- `#define STATE_LOW_POWER_VOLTAGE 0x10000`
- Напряжение на силовой части ниже чем напряжение Low Voltage Protection.
- `#define STATE_H_BRIDGE_FAULT 0x20000`
- Получен сигнал от драйвера о неисправности

Флаги состояния GPIO входов

Содержат бинарные значения состояния контроллера. Могут быть объединены с помощью логического ИЛИ.

См. также

- ```
status_t::flags
get_status
status_t::GPIOFlags, get_status_impl
```
- `#define STATE_DIG_SIGNAL 0xFFFF`
  - Флаги цифровых сигналов.
  - `#define STATE_RIGHT_EDGE 0x0001`
  - Достижение правой границы.
  - `#define STATE_LEFT_EDGE 0x0002`
  - Достижение левой границы.
  - `#define STATE_BUTTON_RIGHT 0x0004`
  - Состояние кнопки "вправо" (1, если нажата).
  - `#define STATE_BUTTON_LEFT 0x0008`
  - Состояние кнопки "влево" (1, если нажата).
  - `#define STATE_GPIO_PINOUT 0x0010`

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

- #define `STATE_GPIO_LEVEL` 0x0020  
Состояние ввода/вывода общего назначения.
- #define `STATE_HALL_A` 0x0040  
Состояние вывода датчика холла(а)(флаг "1", если датчик активен).
- #define `STATE_HALL_B` 0x0080  
Состояние вывода датчика холла(б)(флаг "1", если датчик активен).
- #define `STATE_HALL_C` 0x0100  
Состояние вывода датчика холла(с)(флаг "1", если датчик активен).
- #define `STATE_BRAKE` 0x0200  
Состояние вывода управления тормозом(флаг "1" - если на тормоз подаётся питание, "0" - если тормоз не запитан).
- #define `STATE_REV_SENSOR` 0x0400  
Состояние вывода датчика оборотов(флаг "1", если датчик активен).
- #define `STATE_SYNC_INPUT` 0x0800  
Состояние входа синхронизации(1, если вход синхронизации активен).
- #define `STATE_SYNC_OUTPUT` 0x1000  
Состояние выхода синхронизации(1, если выход синхронизации активен).
- #define `STATE_ENC_A` 0x2000  
Состояние ножки А энкодера(флаг "1", если энкодер активен).
- #define `STATE_ENC_B` 0x4000  
Состояние ножки В энкодера(флаг "1", если энкодер активен).

Состояние энкодера

Состояние энкодера, подключенного к контроллеру.

См. также

```
status_t::encsts
get_status
status_t::EncSts, get_status_impl
```

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

Состояние обмоток

Состояние обмоток двигателя, подключенного к контроллеру.

См. также

```
status_t::windsts
get_status
status_t::WindSts, get_status_impl
```

- #define `WIND_A_STATE_ABSENT` 0x00  
Обмотка А не подключена.
- #define `WIND_A_STATE_UNKNOWN` 0x01  
Состояние обмотки А неизвестно.
- #define `WIND_A_STATE_MALFUNC` 0x02

- Короткое замыкание на обмотке А.
- `#define WIND_A_STATE_OK 0x03`  
Обмотка А работает адекватно.
- `#define WIND_B_STATE_ABSENT 0x00`  
Обмотка В не подключена.
- `#define WIND_B_STATE_UNKNOWN 0x10`  
Состояние обмотки В неизвестно.
- `#define WIND_B_STATE_MALFUNC 0x20`  
Короткое замыкание на обмотке В.
- `#define WIND_B_STATE_OK 0x30`  
Обмотка В работает адекватно.

#### Состояние команды движения

Состояние команды движения (касается `command_move`, `command_movr`, `command_left`, `command_right`, `command_stop`, `command_home`, `command_loft`, `command_sstp`) и статуса её выполнения (выполняется, завершено, ошибка)

См. также

```
status_t::mvcmdsts
get_status
status_t::MvCmdSts, get_status_impl
```

- `#define MVCMD_NAME_BITS 0x3F`  
Битовая маска активной команды.
- `#define MVCMD_UKNWN 0x00`  
Неизвестная команда.
- `#define MVCMD_MOVE 0x01`  
Команда move.
- `#define MVCMD_MOVR 0x02`  
Команда movr.
- `#define MVCMD_LEFT 0x03`  
Команда left.
- `#define MVCMD_RIGHT 0x04`  
Команда right.
- `#define MVCMD_STOP 0x05`  
Команда stop.
- `#define MVCMD_HOME 0x06`  
Команда home.
- `#define MVCMD_LOFT 0x07`  
Команда loft.
- `#define MVCMD_SSTP 0x08`  
Команда плавной остановки(SSTP).
- `#define MVCMD_ERROR 0x40`  
Состояние завершения движения (1 - команда движения выполнена с ошибкой, 0 - команда движения выполнена корректно).
- `#define MVCMD_RUNNING 0x80`  
Состояние команды движения (0 - команда движения выполнена, 1 - команда движения сейчас выполняется).

#### Флаги параметров мотора

Определяют настройки движения и работу ограничителей. Возвращаются командой `get_engine_settings`. Могут быть объединены с помощью логического ИЛИ.

См. также

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

- `#define ENGINE_REVERSE 0x01`  
Флаг реверса.
- `#define ENGINE_MAX_SPEED 0x04`  
Флаг максимальной скорости.
- `#define ENGINE АнтиPLAY 0x08`  
Компенсация люфта.
- `#define ENGINE_ACCEL_ON 0x10`  
Ускорение.
- `#define ENGINE_LIMIT_VOLT 0x20`  
Номинальное напряжение мотора.
- `#define ENGINE_LIMIT_CURR 0x40`  
Номинальный ток мотора.
- `#define ENGINE_LIMIT_RPM 0x80`  
Номинальная частота вращения мотора.

Флаги параметров микрошагового режима

Определяют деление шага в микрошаговом режиме. Используются с шаговыми моторами. Возвращаются командой `get_engine_settings`. Могут быть объединены с помощью логического ИЛИ.

См. также

```
engine_settings_t::flags
set_engine_settings
get_engine_settings
engine_settings_t::MicrostepMode, get_engine_settings, set_engine_settings
```

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

Флаги, определяющие тип мотора

Определяют тип мотора. Возвращаются командой `get_entype_settings`.

См. также

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

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

Флаги, определяющие тип силового драйвера

Определяют тип силового драйвера. Возвращаются командой `get_entype_settings`.

См. также

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

- `#define DRIVER_TYPE_DISCRETE_FET 0x01`  
Силовой драйвер на дискретных мосфет-ключках.
- `#define DRIVER_TYPE_INTEGRATE 0x02`  
Силовой драйвер с использованием ключей, интегрированных в микросхему.
- `#define DRIVER_TYPE_EXTERNAL 0x03`  
Внешний силовой драйвер.

Флаги параметров питания шагового мотора

Возвращаются командой `get_power_settings`.

См. также

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

Флаги критических параметров.

Возвращаются командой `get_secure_settings`.



См. также

```
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`  
Если флаг установлен, то войти в состояние Alarm при получении сигнала подступающего перегрева с драйвера.
- `#define LOW_UPWR_PROTECTION 0x02`  
Если установлен, то выключать силовую часть при напряжении меньшем LowUpwrOff.
- `#define H_BRIDGE_ALERT 0x04`  
Если установлен, то выключать силовую часть при сигнале неполадки в одном из транзисторных мостов.
- `#define ALARM_ON_BORDERS_SWAP_MISSET 0x08`  
Если флаг установлен, то войти в состояние Alarm при получении сигнала с противоположного концевика.
- `#define ALARM_FLAGS_STICKING 0x10`  
Если флаг установлен, то только по команде STOP возможен сброс всех флагов ALARM.
- `#define USB_BREAK_RECONNECT 0x20`  
Если флаг установлен, то будет включен блок перезагрузки USB при поломке связи.

Флаги установки положения

Возвращаются командой `get_position`.

См. также

```
get_position
set_position
set_position_t::PosFlags, set_position
```

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

Тип обратной связи.

См. также

```
set_feedback_settings
get_feedback_settings
feedback_settings_t::FeedbackType, get_feedback_settings, set_feedback_settings
```

- `#define FEEDBACK_ENCODER 0x01`  
Обратная связь с помощью энкодера.
- `#define FEEDBACK_ENCODERHALL 0x03`  
Обратная связь с помощью датчика Холла.
- `#define FEEDBACK_EMF 0x04`  
Обратная связь по ЭДС.
- `#define FEEDBACK_NONE 0x05`  
Обратная связь отсутствует.

Флаги обратной связи.

См. также

```
set_feedback_settings
get_feedback_settings
feedback_settings_t::FeedbackFlags, get_feedback_settings, set_feedback_settings
```

- #define **FEEDBACK\_ENC\_REVERSE** 0x01  
Обратный счет у энкодера.
- #define **FEEDBACK\_HALL\_REVERSE** 0x02  
Обратный счёт позиции по датчикам Холла.

Флаги настроек синхронизации входа

См. также

```
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  
Включение необходимости импульса синхронизации для начала движения.
- #define **SYNCIN\_INVERT** 0x02  
Если установлен - срабатывает по переходу из 1 в 0.
- #define **SYNCIN\_GOTOPOSITION** 0x04  
Если флаг установлен, то двигатель смещается к позиции, установленной в Position и uPosition, иначе двигатель смещается на Position и uPosition.

Флаги настроек синхронизации выхода

См. также

```
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  
Синхронизация выхода работает согласно настройкам, если флаг установлен.
- #define **SYNCOUT\_STATE** 0x02  
Когда значение выхода управляется напрямую (см.
- #define **SYNCOUT\_INVERT** 0x04  
Нулевой логический уровень является активным, если флаг установлен, а единичный - если флаг сброшен.
- #define **SYNCOUT\_IN\_STEPS** 0x08  
Если флаг установлен использовать шаги/импульсы энкодера для выходных импульсов синхронизации вместо миллисекунд.
- #define **SYNCOUT\_ONSTART** 0x10  
Генерация синхронизирующего импульса при начале движения.
- #define **SYNCOUT\_ONSTOP** 0x20  
Генерация синхронизирующего импульса при остановке.
- #define **SYNCOUT\_ONPERIOD** 0x40  
Выдать импульс синхронизации после прохождения SyncOutPeriod отсчётов.

Флаги настройки работы внешнего ввода/вывода

См. также

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

Флаги настройки режимов внешнего ввода/вывода

См. также

```
extio_settings_t::extio_mode_flags
get_extio_settings
set_extio_settings
extio_settings_t::EXTIOModeFlags, get_extio_settings, set_extio_settings
```

- `#define EXTIO_SETUP_MODE_IN_NOP 0x00`  
Ничего не делать.
- `#define EXTIO_SETUP_MODE_IN_STOP 0x01`  
По переднему фронту входного сигнала делается остановка двигателя (эквивалент команды STOP).
- `#define EXTIO_SETUP_MODE_IN_PWOF 0x02`  
Выполняет команду PWOF, обесточивая обмотки двигателя.
- `#define EXTIO_SETUP_MODE_IN_MOVR 0x03`  
Выполняется команда MOVR с последними настройками.
- `#define EXTIO_SETUP_MODE_IN_HOME 0x04`  
Выполняется команда HOME.
- `#define EXTIO_SETUP_MODE_OUT_OFF 0x00`  
Ножка всегда в неактивном состоянии.
- `#define EXTIO_SETUP_MODE_OUT_ON 0x10`  
Ножка всегда в активном состоянии.
- `#define EXTIO_SETUP_MODE_OUT_MOVING 0x20`  
Ножка находится в активном состоянии при движении.
- `#define EXTIO_SETUP_MODE_OUT_ALARM 0x30`  
Ножка находится в активном состоянии при нахождении в состоянии ALARM.
- `#define EXTIO_SETUP_MODE_OUT_MOTOR_ON 0x40`  
Ножка находится в активном состоянии при подаче питания на обмотки.
- `#define EXTIO_SETUP_MODE_OUT_MOTOR_FOUND 0x50`  
Ножка находится в активном состоянии при обнаружении подключенного двигателя (первой обмотки).

Флаги границ

Типы границ и поведение позиционера на границах. Могут быть объединены с помощью побитового ИЛИ.

См. также

```
get_edges_settings
set_edges_settings
edges_settings_t::BorderFlags, get_edges_settings, set_edges_settings
```

- `#define BORDER_IS_ENCODER 0x01`  
Если флаг установлен, границы определяются предустановленными точками на шкале позиции.
- `#define BORDER_STOP_LEFT 0x02`

- Если флаг установлен, мотор останавливается при достижении левой границы.
- `#define BORDER_STOP_RIGHT 0x04`
- Если флаг установлен, мотор останавливается при достижении правой границы.
- `#define BORDERS_SWAP_MISSET_DETECTION 0x08`
- Если флаг установлен, мотор останавливается при достижении обеих границ.

#### Флаги концевых выключателей

Определяют направление и состояние границ. Могут быть объединены с помощью побитового ИЛИ.

См. также

```
get_edges_settings
set_edges_settings
edges_settings_t::EnderFlags, get_edges_settings, set_edges_settings
```

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

#### Флаги настроек тормоза

Определяют поведение тормоза. Могут быть объединены с помощью побитового ИЛИ.

См. также

```
get_brake_settings
set_brake_settings
brake_settings_t::BrakeFlags, get_brake_settings, set_brake_settings
```

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

#### Флаги управления

Определяют параметры управления мотором с помощью джойстика или кнопок. Могут быть объединены с помощью побитового ИЛИ.

См. также

```
get_control_settings
set_control_settings
control_settings_t::Flags, get_control_settings, set_control_settings
```

- `#define CONTROL_MODE_BITS 0x03`
- Биты управления мотором с помощью джойстика или кнопок влево/вправо.
- `#define CONTROL_MODE_OFF 0x00`
- Управление отключено.
- `#define CONTROL_MODE_JOY 0x01`
- Управление с помощью джойстика.
- `#define CONTROL_MODE_LR 0x02`
- Управление с помощью кнопок left/right.
- `#define CONTROL_BTN_LEFT_PUSHED_OPEN 0x04`
- Левая кнопка нормально разомкнутая, если флаг установлен.

- `#define CONTROL_BTN_RIGHT_PUSHED_OPEN 0x08`  
Правая кнопка нормально разомкнутая, если флаг установлен.

Флаги джойстика

Управляют состояниями джойстика.

См. также

```
set_joystick_settings
get_joystick_settings
joystick_settings_t::JoyFlags, get_joystick_settings, set_joystick_settings
```

- `#define JOY_REVERSE 0x01`  
Реверс воздействия джойстика.

Флаги контроля позиции

Определяют настройки контроля позиции. Могут быть объединены с помощью побитового ИЛИ.

См. также

```
get_ctp_settings
set_ctp_settings
ctp_settings_t::CTPFlags, get_ctp_settings, set_ctp_settings
```

- `#define CTP_ENABLED 0x01`  
Контроль позиции включен, если флаг установлен.
- `#define CTP_BASE 0x02`  
Опорой является датчик оборотов, если флаг установлен; иначе - энкодер.
- `#define CTP_ALARM_ON_ERROR 0x04`  
Войти в состояние ALARM при расхождении позиции, если флаг установлен.
- `#define REV_SENS_INV 0x08`  
Сенсор считается активным, когда на нём 0, а инвертирование делает активным уровнем 1.

Флаги настроек команды home

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

См. также

```
get_home_settings
set_home_settings
command_home
home_settings_t::HomeFlags, get_home_settings, set_home_settings
```

- `#define HOME_DIR_FIRST 0x01`  
Определяет направление первоначального движения мотора после поступления команды HOME.
- `#define HOME_DIR_SECOND 0x02`  
Определяет направление второго движения мотора.
- `#define HOME_MV_SEC_EN 0x04`  
Если флаг установлен, реализуется второй этап доводки в домашнюю позицию; иначе - этап пропускается.
- `#define HOME_HALF_MV 0x08`  
Если флаг установлен, в начале второго движения первые пол оборота сигналы завершения движения игнорируются.
- `#define HOME_STOP_FIRST_BITS 0x30`  
Биты, отвечающие за выбор сигнала завершения первого движения.
- `#define HOME_STOP_FIRST_REV 0x10`  
Первое движение завершается по сигналу с Revolution sensor.
- `#define HOME_STOP_FIRST_SYN 0x20`

- Первое движение завершается по сигналу со входа синхронизации.
- `#define HOME_STOP_FIRST_LIM 0x30`  
Первое движение завершается по сигналу с концевика.
- `#define HOME_STOP_SECOND_BITS 0xC0`  
Биты, отвечающие за выбор сигнала завершения второго движения.
- `#define HOME_STOP_SECOND_REV 0x40`  
Второе движение завершается по сигналу с Revolution sensor.
- `#define HOME_STOP_SECOND_SYN 0x80`  
Второе движение завершается по сигналу со входа синхронизации.
- `#define HOME_STOP_SECOND_LIM 0xC0`  
Второе движение завершается по сигналу с концевика.

Флаги настроек четности команды uart

См. также

`uart_settings_t::UARTSetupFlags, get_uart_settings, set_uart_settings`

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

Флаг типа двигателя

См. также

`motor_settings_t::MotorType, get_motor_settings, set_motor_settings`

- `#define MOTOR_TYPE_STEP 0x01`  
Шаговый двигатель
- `#define MOTOR_TYPE_DC 0x02`  
DC двигатель
- `#define MOTOR_TYPE_BLDC 0x03`  
BLDC двигатель

Флаги настроек энкодера

См. также

`accessories_settings_t::MBSettings, get_accessories_settings, set_accessories_settings`

- `#define ENCSET_DIFFERENTIAL_OUTPUT 0x001`  
Если флаг установлен, то энкодер имеет дифференциальный выход, иначе - несимметричный выход
- `#define ENCSET_PUSHPULL_OUTPUT 0x004`  
Если флаг установлен, то энкодер имеет двухтактный выход, иначе - выход с открытым коллектором
- `#define ENCSET_INDEXCHANNEL_PRESENT 0x010`

- Если флаг установлен, то энкодер имеет дополнительный индексный канал, иначе - он отсутствует
- #define `ENCSET_REVOLUTIONSENSOR_PRESENT` 0x040
  - Если флаг установлен, то энкодер имеет датчик оборотов, иначе - он отсутствует
- #define `ENCSET_REVOLUTIONSENSOR_ACTIVE_HIGH` 0x100
  - Если флаг установлен, то активное состояние датчика оборотов соответствует логической 1, иначе - логическому 0.
- #define `MB_AVAILABLE` 0x01
  - Если флаг установлен, то магнитный тормоз доступен
- #define `MB_POWERED_HOLD` 0x02
  - Если флаг установлен, то магнитный тормоз находится в режиме удержания (активен) при подаче питания

#### Флаги настроек температурного датчика

См. также

- `accessories_settings_t::LimitSwitchesSettings`, `get_accessories_settings`, `set_accessories_settings`
- #define `TS_TYPE_BITS` 0x07
  - Биты, отвечающие за тип температурного датчика.
- #define `TS_TYPE_THERMOCOUPLE` 0x01
  - Термопара
- #define `TS_TYPE_SEMICONDUCTOR` 0x02
  - Полупроводниковый температурный датчик
- #define `TS_AVAILABLE` 0x08
  - Если флаг установлен, то датчик температуры доступен
- #define `LS_ON_SW1_AVAILABLE` 0x01
  - Если флаг установлен, то концевик, подключенный к ножке SW1, доступен
- #define `LS_ON_SW2_AVAILABLE` 0x02
  - Если флаг установлен, то концевик, подключенный к ножке SW2, доступен
- #define `LS_SW1_ACTIVE_LOW` 0x04
  - Если флаг установлен, то концевик, подключенный к ножке SW1, считается сработавшим по низкому уровню на контакте
- #define `LS_SW2_ACTIVE_LOW` 0x08
  - Если флаг установлен, то концевик, подключенный к ножке SW2, считается сработавшим по низкому уровню на контакте
- #define `LS_SHORTED` 0x10
  - Если флаг установлен, то концевики закорочены.

#### Определения типов

- typedef int `device_t`
  - Тип идентификатора устройства
- typedef int `result_t`
  - Тип, определяющий результат выполнения команды.
- typedef uint32\_t `device_enumeration_t`
  - TODO.
- typedef struct `calibration_t` `calibration_t`
  - Calibration companion structure TODO docme.

## Функции

Группа команд настройки контроллера

Функции для чтения/записи большинства настроек контроллера.

- `result_t XIMC_API set_feedback_settings (device_t id, const feedback_settings_t *feedback_settings)`  
Запись настроек обратной связи.
- `result_t XIMC_API get_feedback_settings (device_t id, feedback_settings_t *feedback_settings)`  
Чтение настроек обратной связи
- `result_t XIMC_API set_home_settings (device_t id, const home_settings_t *home_settings)`  
Команда записи настроек для подхода в home position.
- `result_t XIMC_API set_home_settings_calb (device_t id, const home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_home_settings (device_t id, home_settings_t *home_settings)`  
Команда чтения настроек для подхода в home position.
- `result_t XIMC_API get_home_settings_calb (device_t id, home_settings_calb_t *home_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_move_settings (device_t id, const move_settings_t *move_settings)`  
Команда записи настроек перемещения (скорость, ускорение, threshold и скорость в режиме антилюфта).
- `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)`  
Команда чтения настроек перемещения (скорость, ускорение, threshold и скорость в режиме антилюфта).
- `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)`  
Запись настроек мотора.
- `result_t XIMC_API set_engine_settings_calb (device_t id, const engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_engine_settings (device_t id, engine_settings_t *engine_settings)`  
Чтение настроек мотора.
- `result_t XIMC_API get_engine_settings_calb (device_t id, engine_settings_calb_t *engine_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_entype_settings (device_t id, const entype_settings_t *entype_settings)`  
Запись информации о типе мотора и типе силового драйвера.
- `result_t XIMC_API get_entype_settings (device_t id, entype_settings_t *entype_settings)`  
Возвращает информацию о типе мотора и силового драйвера.
- `result_t XIMC_API set_power_settings (device_t id, const power_settings_t *power_settings)`  
Команда записи параметров питания мотора.
- `result_t XIMC_API get_power_settings (device_t id, power_settings_t *power_settings)`  
Команда чтения параметров питания мотора.
- `result_t XIMC_API set_secure_settings (device_t id, const secure_settings_t *secure_settings)`  
Команда записи установок защит.
- `result_t XIMC_API get_secure_settings (device_t id, secure_settings_t *secure_settings)`  
Команда записи установок защит.
- `result_t XIMC_API set_edges_settings (device_t id, const edges_settings_t *edges_settings)`  
Запись настроек границ и концевых выключателей.
- `result_t XIMC_API set_edges_settings_calb (device_t id, const edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)`



- `result_t XIMC_API get_edges_settings (device_t id, edges_settings_t *edges_settings)`  
Чтение настроек границ и концевых выключателей.
- `result_t XIMC_API get_edges_settings_calb (device_t id, edges_settings_calb_t *edges_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_pid_settings (device_t id, const pid_settings_t *pid_settings)`  
Запись ПИД коэффициентов.
- `result_t XIMC_API get_pid_settings (device_t id, pid_settings_t *pid_settings)`  
Чтение ПИД коэффициентов.
- `result_t XIMC_API set_sync_in_settings (device_t id, const sync_in_settings_t *sync_in_settings)`  
Запись настроек для входного импульса синхронизации.
- `result_t XIMC_API set_sync_in_settings_calb (device_t id, const sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_sync_in_settings (device_t id, sync_in_settings_t *sync_in_settings)`  
Чтение настроек для входного импульса синхронизации.
- `result_t XIMC_API get_sync_in_settings_calb (device_t id, sync_in_settings_calb_t *sync_in_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_sync_out_settings (device_t id, const sync_out_settings_t *sync_out_settings)`  
Запись настроек для выходного импульса синхронизации.
- `result_t XIMC_API set_sync_out_settings_calb (device_t id, const sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_sync_out_settings (device_t id, sync_out_settings_t *sync_out_settings)`  
Чтение настроек для выходного импульса синхронизации.
- `result_t XIMC_API get_sync_out_settings_calb (device_t id, sync_out_settings_calb_t *sync_out_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_extio_settings (device_t id, const extio_settings_t *extio_settings)`  
Команда записи параметров настройки режимов внешнего ввода/вывода.
- `result_t XIMC_API get_extio_settings (device_t id, extio_settings_t *extio_settings)`  
Команда чтения параметров настройки режимов внешнего ввода/вывода.
- `result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t *brake_settings)`  
Запись настроек управления тормозом.
- `result_t XIMC_API get_brake_settings (device_t id, brake_settings_t *brake_settings)`  
Чтение настроек управления тормозом.
- `result_t XIMC_API set_control_settings (device_t id, const control_settings_t *control_settings)`  
Запись настроек управления мотором.
- `result_t XIMC_API set_control_settings_calb (device_t id, const control_settings_calb_t *control_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API get_control_settings (device_t id, control_settings_t *control_settings)`  
Чтение настроек управления мотором.
- `result_t XIMC_API get_control_settings_calb (device_t id, control_settings_calb_t *control_settings_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_joystick_settings (device_t id, const joystick_settings_t *joystick_settings)`  
Запись настроек джойстика.
- `result_t XIMC_API get_joystick_settings (device_t id, joystick_settings_t *joystick_settings)`  
Чтение настроек джойстика.
- `result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t *ctp_settings)`  
Запись настроек контроля позиции(для шагового двигателя).
- `result_t XIMC_API get_ctp_settings (device_t id, ctp_settings_t *ctp_settings)`  
Чтение настроек контроля позиции(для шагового двигателя).
- `result_t XIMC_API set_uart_settings (device_t id, const uart_settings_t *uart_settings)`  
Команда записи настроек UART.

- `result_t XIMC_API get_uart_settings (device_t id, uart_settings_t *uart_settings)`  
Команда чтения настроек UART.
- `result_t XIMC_API set_controller_name (device_t id, const controller_name_t *controller_name)`  
Запись пользовательского имени контроллера и настроек в FRAM.
- `result_t XIMC_API get_controller_name (device_t id, controller_name_t *controller_name)`  
Чтение пользовательского имени контроллера и настроек из FRAM.

#### Группа команд управления движением

- `result_t XIMC_API command_stop (device_t id)`  
Немедленная остановка двигателя, переход в состояние STOP, ключи в режиме BREAK (обмотки накоротко замкнуты), режим "удержания" деактивируется для DC двигателей, удержание тока в обмотках для шаговых двигателей (с учётом Power management настроек).
- `result_t XIMC_API set_add_sync_in_action (device_t id, const add_sync_in_action_t *add_sync_in_action)`  
Это команда добавляет один элемент в буфер FIFO команд, выполняемых при получении входного импульса синхронизации.
- `result_t XIMC_API set_add_sync_in_action_calb (device_t id, const add_sync_in_action_calb_t *add_sync_in_action_calb, const calibration_t *calibration)`
- `result_t XIMC_API command_power_off (device_t id)`  
Немедленное отключение питания двигателя вне зависимости от его состояния.
- `result_t XIMC_API command_move (device_t id, int Position, int uPosition)`  
При получении команды "move" двигатель начинает перемещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), к точке указанной в полях Position, uPosition.
- `result_t XIMC_API command_move_calb (device_t id, float Position, const calibration_t *calibration)`
- `result_t XIMC_API command_movr (device_t id, int DeltaPosition, int uDeltaPosition)`  
При получении команды "movr" двигатель начинает смещаться (если не используется режим "ТТЛСинхроВхода"), с заранее установленными параметрами (скорость, ускорение, удержание), влево или вправо (зависит от знака DeltaPosition) на количество импульсов указанное в полях 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)`  
Поля скоростей знаковые.
- `result_t XIMC_API command_left (device_t id)`  
При получении команды "left" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), влево.
- `result_t XIMC_API command_right (device_t id)`  
При получении команды "right" двигатель начинает смещаться, с заранее установленными параметрами (скорость, ускорение), вправо.
- `result_t XIMC_API command_loft (device_t id)`  
При получении команды "loft" двигатель смещается из текущей точки на расстояние GENG::Antiplay, затем двигается в ту же точку.
- `result_t XIMC_API command_sstp (device_t id)`  
Плавная остановка.
- `result_t XIMC_API get_position (device_t id, get_position_t *the_get_position)`  
Считывает значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.
- `result_t XIMC_API get_position_calb (device_t id, get_position_calb_t *the_get_position_calb, const calibration_t *calibration)`
- `result_t XIMC_API set_position (device_t id, const set_position_t *the_set_position)`  
Устанавливает произвольное значение положения в шагах и микрошагах для шагового двигателя и в шагах энкодера всех двигателей.

- `result_t XIMC_API set_position_calb (device_t id, const set_position_calb_t *the_set_position_calb, const calibration_t *calibration)`
- `result_t XIMC_API command_zero (device_t id)`  
Устанавливает текущую позицию и позицию в которую осуществляется движение по командам move и movt равными нулю для всех случаев, кроме движения к позиции назначения.

Группа команд сохранения и загрузки настроек

- `result_t XIMC_API command_save_settings (device_t id)`  
При получении команды контроллер выполняет операцию сохранения текущих настроек во встроенную энергонезависимую память контроллера.
- `result_t XIMC_API command_read_settings (device_t id)`  
Чтение всех настроек контроллера из flash памяти в оперативную, заменяя текущие настройки.
- `result_t XIMC_API command_eesave_settings (device_t id)`  
Запись настроек контроллера в EEPROM память позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.
- `result_t XIMC_API command_eeread_settings (device_t id)`  
Чтение настроек контроллера из EEPROM памяти позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.
- `result_t XIMC_API get_chart_data (device_t id, chart_data_t *chart_data)`  
Команда чтения состояния обмоток и других не часто используемых данных.
- `result_t XIMC_API get_serial_number (device_t id, unsigned int *SerialNumber)`  
Чтение серийного номера контроллера.
- `result_t XIMC_API get_firmware_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)`  
Чтение номера версии прошивки контроллера.
- `result_t XIMC_API service_command_updf (device_t id)`  
Команда переводит контроллер в режим обновления прошивки.

Группа сервисных команд

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

Группа команд работы с EEPROM подвижки

- `result_t XIMC_API set_stage_name (device_t id, const stage_name_t *stage_name)`  
Запись пользовательского имени подвижки в EEPROM.
- `result_t XIMC_API get_stage_name (device_t id, stage_name_t *stage_name)`  
Чтение пользовательского имени подвижки из EEPROM.
- `result_t XIMC_API set_stage_information (device_t id, const stage_information_t *stage_information)`  
Запись информации о позиционере в EEPROM.
- `result_t XIMC_API get_stage_information (device_t id, stage_information_t *stage_information)`  
Чтение информации о позиционере из EEPROM.
- `result_t XIMC_API set_stage_settings (device_t id, const stage_settings_t *stage_settings)`

- Запись настроек позиционера в EEPROM.
- `result_t XIMC_API get_stage_settings (device_t id, stage_settings_t *stage_settings)`
- Чтение настроек позиционера из EEPROM.
- `result_t XIMC_API set_motor_information (device_t id, const motor_information_t *motor_information)`
- Запись информации о двигателе в EEPROM.
- `result_t XIMC_API get_motor_information (device_t id, motor_information_t *motor_information)`
- Чтение информации о двигателе из EEPROM.
- `result_t XIMC_API set_motor_settings (device_t id, const motor_settings_t *motor_settings)`
- Запись настроек двигателя в EEPROM.
- `result_t XIMC_API get_motor_settings (device_t id, motor_settings_t *motor_settings)`
- Чтение настроек двигателя из EEPROM.
- `result_t XIMC_API set_encoder_information (device_t id, const encoder_information_t *encoder_information)`
- Запись информации об энкодере в EEPROM.
- `result_t XIMC_API get_encoder_information (device_t id, encoder_information_t *encoder_information)`
- Чтение информации об энкодере из EEPROM.
- `result_t XIMC_API set_encoder_settings (device_t id, const encoder_settings_t *encoder_settings)`
- Запись настроек энкодера в EEPROM.
- `result_t XIMC_API get_encoder_settings (device_t id, encoder_settings_t *encoder_settings)`
- Чтение настроек энкодера из EEPROM.
- `result_t XIMC_API set_hallsensor_information (device_t id, const hallsensor_information_t *hallsensor_information)`
- Запись информации об датчиках Холла в EEPROM.
- `result_t XIMC_API get_hallsensor_information (device_t id, hallsensor_information_t *hallsensor_information)`
- Чтение информации об датчиках Холла из EEPROM.
- `result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t *hallsensor_settings)`
- Запись настроек датчиков Холла в EEPROM.
- `result_t XIMC_API get_hallsensor_settings (device_t id, hallsensor_settings_t *hallsensor_settings)`
- Чтение настроек датчиков Холла из EEPROM.
- `result_t XIMC_API set_gear_information (device_t id, const gear_information_t *gear_information)`
- Запись информации о редукторе в EEPROM.
- `result_t XIMC_API get_gear_information (device_t id, gear_information_t *gear_information)`
- Чтение информации о редукторе из EEPROM.
- `result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t *gear_settings)`
- Запись настроек редуктора в EEPROM.
- `result_t XIMC_API get_gear_settings (device_t id, gear_settings_t *gear_settings)`
- Чтение настроек редуктора из EEPROM.
- `result_t XIMC_API set_accessories_settings (device_t id, const accessories_settings_t *accessories_settings)`
- Запись информации о дополнительных аксессуарах в EEPROM.
- `result_t XIMC_API get_accessories_settings (device_t id, accessories_settings_t *accessories_settings)`
- Чтение информации о дополнительных аксессуарах из EEPROM.
- `result_t XIMC_API get_bootloader_version (device_t id, unsigned int *Major, unsigned int *Minor, unsigned int *Release)`
- Чтение номера версии прошивки контроллера.
- `result_t XIMC_API goto_firmware (device_t id, uint8_t *ret)`
- TODO Проверка наличия прошивки в контроллере

- `result_t XIMC_API has_firmware` (const char \*name, uint8\_t \*ret)  
Проверка наличия прошивки в контроллере
- `result_t XIMC_API command_update_firmware` (const char \*name, const uint8\_t \*data, uint32\_t data\_size)  
Обновление прошивки
- `result_t XIMC_API write_key` (const char \*name, uint8\_t \*key)  
Запись ключа защиты TODO fix docs Функция используется только производителем.
- `result_t XIMC_API command_reset` (device\_t id)  
Перезагрузка контроллера.
- `result_t XIMC_API command_clear_fram` (device\_t id)  
Очистка FRAM памяти контроллера.

## Управление устройством

### Функции поиска и открытия/закрытия устройств

- `typedef char * pchar`  
Не обращайтесь на меня внимание
- `typedef void(XIMC_CALLCONV * logging_callback_t)(int loglevel, const wchar_t *message)`  
Прототип функции обратного вызова для логирования
- `device_t XIMC_API open_device` (const char \*name)  
Открывает устройство по имени name и возвращает идентификатор, который будет использоваться для обращения к устройству.
- `result_t XIMC_API close_device` (device\_t \*id)  
Закрывает устройство
- `result_t XIMC_API probe_device` (const char \*name)  
Проверяет, является ли устройство с именем name XIMC-совместимым.
- `device_enumeration_t XIMC_API enumerate_devices` (int probe\_flags)  
Перечисляет все XIMC-совместимые устройства.
- `result_t XIMC_API free_enumerate_devices` (device\_enumeration\_t device\_enumeration)  
Освобождает память, выделенную enumerate\_devices.
- `int XIMC_API get_device_count` (device\_enumeration\_t device\_enumeration)  
Возвращает количество подключенных устройств.
- `pchar XIMC_API get_device_name` (device\_enumeration\_t device\_enumeration, int device\_index)  
Возвращает имя подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_serial` (device\_enumeration\_t device\_enumeration, int device\_index, uint32\_t \*serial)  
Возвращает серийный номер подключенного устройства из перечисления устройств.
- `result_t XIMC_API get_enumerate_device_information` (device\_enumeration\_t device\_enumeration, int device\_index, device\_information\_t \*device\_information)  
Возвращает серийный номер подключенного устройства из перечисления устройств.
- `result_t XIMC_API reset_locks` ()  
Снимает блокировку библиотеки в экстренном случае.
- `result_t XIMC_API ximc_fix_usbser_sys` (const char \*device\_name)  
Исправление ошибки драйвера USB в Windows.
- `void XIMC_API msec_sleep` (unsigned int msec)  
Приостанавливает работу на указанное время
- `void XIMC_API ximc_version` (char \*version)  
Возвращает версию библиотеки
- `void XIMC_API logging_callback_stderr_wide` (int loglevel, const wchar\_t \*message)  
Простая функция логирования на stderr в широких символах
- `void XIMC_API logging_callback_stderr_narrow` (int loglevel, const wchar\_t \*message)

- Простая функция логирования на stderr в узких (однобайтных) символах
- `void XIMC_API set_logging_callback (logging_callback_t logging_callback)`  
Устанавливает функцию обратного вызова для логирования.
  - `result_t XIMC_API get_status (device_t id, status_t *status)`  
Возвращает информацию о текущем состоянии устройства.
  - `result_t XIMC_API get_status_calb (device_t id, status_calb_t *status, const calibration_t *calibration)`  
TODO document me Эта структура содержит основные параметры текущего состояния контроллера такие как скорость, позиция и флаги состояния.
  - `result_t XIMC_API get_device_information (device_t id, device_information_t *device_information)`  
Возвращает информацию об устройстве.

### 5.1.1 Подробное описание

Заголовочный файл для библиотеки libximc.

### 5.1.2 Макросы

#### 5.1.2.1 #define ALARM\_ON\_DRIVER\_OVERHEATING 0x01

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

Иначе - игнорировать подступающий перегрев с драйвера.

#### 5.1.2.2 #define BORDER\_IS\_ENCODER 0x01

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

Если флаг сброшен, границы определяются концевыми выключателями.

#### 5.1.2.3 #define BORDERS\_SWAP\_MISSET\_DETECTION 0x08

Если флаг установлен, мотор останавливается при достижении обеих границ.

Нужен для предотвращения поломки двигателя при неправильных настройках концевиков

#### 5.1.2.4 #define DRIVER\_TYPE\_DISCRETE\_FET 0x01

Силовой драйвер на дискретных мосфет-ключках.

Используется по умолчанию.

#### 5.1.2.5 #define ENGINE\_ACCEL\_ON 0x10

Ускорение.

Если флаг установлен, движение происходит с ускорением.

#### 5.1.2.6 #define ENGINE\_ANTIPLAY 0x08

Компенсация люфта.

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

#### 5.1.2.7 `#define ENGINE_LIMIT_CURR 0x40`

Номинальный ток мотора.

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

#### 5.1.2.8 `#define ENGINE_LIMIT_RPM 0x80`

Номинальная частота вращения мотора.

Если флаг установлен, частота вращения ограничивается заданным номинальным значением.

#### 5.1.2.9 `#define ENGINE_LIMIT_VOLT 0x20`

Номинальное напряжение мотора.

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

#### 5.1.2.10 `#define ENGINE_MAX_SPEED 0x04`

Флаг максимальной скорости.

Если флаг установлен, движение происходит на максимальной скорости.

#### 5.1.2.11 `#define ENGINE_REVERSE 0x01`

Флаг реверса.

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

#### 5.1.2.12 `#define ENUMERATE_PROBE 0x01`

Проверить, является ли устройство XIMC-совместимым.

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

#### 5.1.2.13 `#define HOME_DIR_FIRST 0x01`

Определяет направление первоначального движения мотора после поступления команды HOME.

Если флаг установлен - вправо; иначе - влево.

#### 5.1.2.14 `#define HOME_DIR_SECOND 0x02`

Определяет направление второго движения мотора.

Если флаг установлен - вправо; иначе - влево.

5.1.2.15 `#define JOY_REVERSE 0x01`

Реверс воздействия джойстика.

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

5.1.2.16 `#define MVCMD_ERROR 0x40`

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

Имеет смысл если `MVCMD_RUNNING` указывает на завершение движения.

5.1.2.17 `#define POWER_OFF_ENABLED 0x02`

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

Иначе - не снимать.

5.1.2.18 `#define POWER_REDUCT_ENABLED 0x01`

Если флаг установлен, уменьшить ток по прошествии `CurrReductDelay`.

Иначе - не уменьшать.

5.1.2.19 `#define REV_SENS_INV 0x08`

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

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

5.1.2.20 `#define STATE_ALARM 0x00040`

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

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

5.1.2.21 `#define SYNCIN_INVERT 0x02`

Если установлен - срабатывает по переходу из 1 в 0.

Иначе - из 0 в 1.

5.1.2.22 `#define SYNCOUT_ENABLED 0x01`

Синхронизация выхода работает согласно настройкам, если флаг установлен.

В ином случае значение выхода фиксировано и подчиняется `SYNCOUT_STATE`.

5.1.2.23 `#define SYNCOUT_STATE 0x02`

Когда значение выхода управляется напрямую (см.

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



5.1.2.24 `#define XIMC_API`

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

It automatically expands to `__declspec(dllexport)` on `msvc` when including header file

## 5.1.3 Типы

5.1.3.1 `typedef void(XIMC_CALLBACK * logging_callback_t)(int loglevel, const wchar_t *message)`

Прототип функции обратного вызова для логирования

Аргументы

|                       |                     |
|-----------------------|---------------------|
| <code>loglevel</code> | уровень логирования |
| <code>message</code>  | сообщение           |

## 5.1.4 Функции

5.1.4.1 `result_t XIMC_API close_device ( device_t * id )`

Закрывает устройство

Аргументы

|                 |                            |
|-----------------|----------------------------|
| <code>id</code> | - идентификатор устройства |
|-----------------|----------------------------|

5.1.4.2 `result_t XIMC_API command_clear_fram ( device_t id )`

Очистка FRAM памяти контроллера.

Функция используется только производителем.

Аргументы

|                 |                          |
|-----------------|--------------------------|
| <code>id</code> | идентификатор устройства |
|-----------------|--------------------------|

5.1.4.3 `result_t XIMC_API command_eeread_settings ( device_t id )`

Чтение настроек контроллера из EEPROM памяти позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.

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

Аргументы

|                 |                          |
|-----------------|--------------------------|
| <code>id</code> | идентификатор устройства |
|-----------------|--------------------------|

5.1.4.4 `result_t XIMC_API command_eesave_settings ( device_t id )`

Запись настроек контроллера в EEPROM память позиционера, которые непосредственно связаны с позиционером и не меняются без его механической переделки.

Функция должна использоваться только производителем.

## Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.5 result\_t XIMC\_API command\_home ( device\_t id )

Поля скоростей знаковые.

Положительное направление это вправо. Нулевое значение флага направления инвертирует направление, заданное скоростью. Ограничение, накладываемые концевиками, действуют так же, за исключением того, что касание концевика не приводит к остановке. Ограничения максимальной скорости, ускорения и замедления действуют. 1) Двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME\_DIR\_FAST до достижения концевика, если флаг HOME\_STOP\_ENDS установлен, до достижения сигнала с входа синхронизации, если установлен флаг HOME\_STOP\_SYNC (важно как можно точнее поймать момент срабатывания концевика) или до поступления сигнала с датчика оборотов, если установлен флаг HOME\_STOP\_REV\_SN 2) далее двигает согласно скоростям SlowHome, uSlowHome и флагу HOME\_DIR\_SLOW до достижения сигнала с входа синхронизации, если установлен флаг HOME\_MV\_SEC. Если флаг HOME\_MV\_SEC сброшен, пропускаем этот пункт. 3) далее двигает мотор согласно скоростям FastHome, uFastHome и флагу HOME\_DIR\_SLOW на расстояние HomeDelta, uHomeDelta. Описание флагов и переменных см. описание команд GHOM/SHOM

## Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

См. также

[home\\_settings\\_t](#)  
[get\\_home\\_settings](#)  
[set\\_home\\_settings](#)

## 5.1.4.6 result\_t XIMC\_API command\_left ( device\_t id )

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

## Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.7 result\_t XIMC\_API command\_loft ( device\_t id )

При получении команды "loft" двигатель смещается из текущей точки на расстояние GENG:-Antiplay, затем двигается в ту же точку.

## Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.8 result\_t XIMC\_API command\_move ( device\_t id, int Position, int uPosition )

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

точке указанной в полях Position, uPosition.

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

Аргументы

|           |                                                      |
|-----------|------------------------------------------------------|
| Position  | заданная позиция. Диапазон: -2147483647..2147483647. |
| uPosition | часть позиции в микрошагах. Диапазон: -255..255.     |
| id        | идентификатор устройства                             |

5.1.4.9 result\_t XIMC\_API command\_movr ( device\_t id, int DeltaPosition, int uDeltaPosition )

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

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

Аргументы

|                |                                                   |
|----------------|---------------------------------------------------|
| DeltaPosition  | смещение. Диапазон: -2147483647..2147483647.      |
| uDeltaPosition | часть смещения в микрошагах. Диапазон: -255..255. |
| id             | идентификатор устройства                          |

5.1.4.10 result\_t XIMC\_API command\_power\_off ( device\_t id )

Немедленное отключение питания двигателя вне зависимости от его состояния.

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

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

См. также

[get\\_power\\_settings](#)  
[set\\_power\\_settings](#)

5.1.4.11 result\_t XIMC\_API command\_read\_settings ( device\_t id )

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

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.12 result\_t XIMC\_API command\_reset ( device\_t id )

Перезагрузка контроллера.

Функция используется только производителем.

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.13 result\_t XIMC\_API command\_right ( device\_t id )

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

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.14 result\_t XIMC\_API command\_save\_settings ( device\_t id )

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

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.15 result\_t XIMC\_API command\_sstp ( device\_t id )

Плавная остановка.

Двигатель останавливается с ускорением замедления.

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.16 result\_t XIMC\_API command\_stop ( device\_t id )

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

Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.17 result\_t XIMC\_API command\_update\_firmware ( const char \* name, const uint8\_t \* data, uint32\_t data\_size )

Обновление прошивки

## Аргументы

|           |                                     |
|-----------|-------------------------------------|
| name      | идентификатор устройства            |
| data      | указатель на массив байтов прошивки |
| data_size | размер массива в байтах             |

## 5.1.4.18 result\_t XIMC\_API command\_zero ( device\_t id )

Устанавливает текущую позицию и позицию в которую осуществляется движение по командам move и movt равными нулю для всех случаев, кроме движения к позиции назначения.

В последнем случае установить нулём текущую позицию, а позицию назначения пересчитать так, что в абсолютном положении точка назначения не меняется. То есть если мы находились в точке 400 и двигались к 500, то команда Zero делает текущую позицию 0, а позицию назначения - 100. Не изменяет режим движения т.е. если движение осуществлялось, то оно продолжается; если мотор находился в режиме "удержания", то тип удержания сохраняется.

## Аргументы

|    |                          |
|----|--------------------------|
| id | идентификатор устройства |
|----|--------------------------|

## 5.1.4.19 device\_enumeration\_t XIMC\_API enumerate\_devices ( int probe\_flags )

Перечисляет все XIMC-совместимые устройства.

## Аргументы

|    |             |                        |
|----|-------------|------------------------|
| in | probe_flags | флаги поиска устройств |
|----|-------------|------------------------|

## 5.1.4.20 result\_t XIMC\_API free\_enumerate\_devices ( device\_enumeration\_t device\_enumeration )

Освобождает память, выделенную enumerate\_devices.

## Аргументы

|    |                    |                                                          |
|----|--------------------|----------------------------------------------------------|
| in | device_enumeration | закрытый указатель на данные о перечисленных устройствах |
|----|--------------------|----------------------------------------------------------|

## 5.1.4.21 result\_t XIMC\_API get\_accessories\_settings ( device\_t id, accessories\_settings\_t \* accessories\_settings )

Чтение информации о дополнительных аксессуарах из EEPROM.

## Аргументы

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

## 5.1.4.22 result\_t XIMC\_API get\_analog\_data ( device\_t id, analog\_data\_t \* analog\_data )

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

Эта функция используется для тестирования и калибровки устройства.

## Аргументы

|     |             |                          |
|-----|-------------|--------------------------|
|     | id          | идентификатор устройства |
| out | analog_data | аналоговые данные        |

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

Чтение номера версии прошивки контроллера.

## Аргументы

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

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

Чтение настроек управления тормозом.

## Аргументы

|     |                |                                                     |
|-----|----------------|-----------------------------------------------------|
|     | id             | идентификатор устройства                            |
| out | brake_settings | структура, содержащая настройки управления тормозом |

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

Команда чтения состояния обмоток и других не часто используемых данных.

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

См. также

[chart\\_data\\_t](#)

## Аргументы

|     |            |                          |
|-----|------------|--------------------------|
|     | id         | идентификатор устройства |
| out | chart_data | структура chart_data.    |

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

Чтение настроек управления мотором.

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

## Аргументы

|     |                      |                                                                                                 |
|-----|----------------------|-------------------------------------------------------------------------------------------------|
|     | id                   | идентификатор устройства                                                                        |
| out | control_<br>settings | структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо. |

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

Чтение пользовательского имени контроллера и настроек из FRAM.

## Аргументы

|     |                     |                                                                                       |
|-----|---------------------|---------------------------------------------------------------------------------------|
|     | id                  | идентификатор устройства                                                              |
| out | controller_<br>name | структура, содержащая установленное пользовательское имя контроллера и флаги настроек |

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

Чтение настроек контроля позиции(для шагового двигателя).

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

## Аргументы

|     |              |                                                  |
|-----|--------------|--------------------------------------------------|
|     | id           | идентификатор устройства                         |
| out | ctp_settings | структура, содержащая настройки контроля позиции |

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

Чтение данных из прошивки для отладки и поиска неисправностей.

Получаемые данные зависят от версии прошивки, истории и контекста использования.

## Аргументы

|     |                     |                          |
|-----|---------------------|--------------------------|
|     | id                  | идентификатор устройства |
| out | Debug-<br>Data[128] | Данные для отладки.      |

5.1.4.30 `int XIMC_API get_device_count ( device_enumeration_t device_enumeration )`

Возвращает количество подключенных устройств.

## Аргументы

|    |                          |                                                          |
|----|--------------------------|----------------------------------------------------------|
| in | device_ -<br>enumeration | закрытый указатель на данные о перечисленных устройствах |
|----|--------------------------|----------------------------------------------------------|

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

Возвращает информацию об устройстве.

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

Аргументы

|     |                          |                                                    |
|-----|--------------------------|----------------------------------------------------|
|     | id                       | идентификатор устройства.                          |
| out | device_ -<br>information | информация об устройстве Информация об устройстве. |

См. также

[get\\_device\\_information](#)

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

Возвращает имя подключенного устройства из перечисления устройств.

Возвращает имя устройства с номером device\_index.

Аргументы

|    |                          |                                                          |
|----|--------------------------|----------------------------------------------------------|
| in | device_ -<br>enumeration | закрытый указатель на данные о перечисленных устройствах |
| in | device_index             | номер устройства                                         |

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

Чтение настроек границ и концевых выключателей.

См. также

[set\\_edges\\_settings](#)

Аргументы

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

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

Чтение информации об энкодере из EEPROM.



## Аргументы

|     |                           |                                              |
|-----|---------------------------|----------------------------------------------|
|     | id                        | идентификатор устройства                     |
| out | encoder_ -<br>information | структура, содержащая информацию об энкодере |

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

Чтение настроек энкодера из EEPROM.

## Аргументы

|     |                        |                                          |
|-----|------------------------|------------------------------------------|
|     | id                     | идентификатор устройства                 |
| out | encoder_ -<br>settings | структура, содержащая настройки энкодера |

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

Чтение настроек мотора.

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

См. также

[set\\_engine\\_settings](#)

## Аргументы

|     |                       |                                |
|-----|-----------------------|--------------------------------|
|     | id                    | идентификатор устройства       |
| out | engine_ -<br>settings | структура с настройками мотора |

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

Возвращает информацию о типе мотора и силового драйвера.

## Аргументы

|     |            |                          |
|-----|------------|--------------------------|
|     | id         | идентификатор устройства |
| out | EngineType | тип мотора               |
| out | DriverType | тип силового драйвера    |

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

Возвращает серийный номер подключенного устройства из перечисления устройств.

Возвращает серийный номер устройства с номером device\_index.

## Аргументы

|     |                          |                                                         |
|-----|--------------------------|---------------------------------------------------------|
| in  | device_ -<br>enumeration | закрытый указатель на данные о перечисленных устояствах |
| in  | device_index             | номер устройства                                        |
| out | device_ -<br>information | информация об устройстве                                |

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

Возвращает серийный номер подключенного устройства из перечисления устройств.

Возвращает серийный номер устройства с номером device\_index.

## Аргументы

|    |                          |                                                         |
|----|--------------------------|---------------------------------------------------------|
| in | device_ -<br>enumeration | закрытый указатель на данные о перечисленных устояствах |
| in | device_index             | номер устройства                                        |
| in | serial                   | серийный номер устройства                               |

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

Команда чтения параметров настройки режимов внешнего ввода/вывода.

См. также

[set\\_extio\\_settings](#)

## Аргументы

|     |                |                          |
|-----|----------------|--------------------------|
|     | id             | идентификатор устройства |
| out | extio_settings | настройки EXTIO          |

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

Чтение настроек обратной связи

## Аргументы

|     |               |                                                                       |
|-----|---------------|-----------------------------------------------------------------------|
|     | id            | идентификатор устройства                                              |
| out | IPS           | Количество измеряемых отсчётов энкодера на оборот. Диапазон: 1..65535 |
| out | FeedbackType  | тип обратной связи                                                    |
| out | FeedbackFlags | флаги обратной связи                                                  |

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

Чтение номера версии прошивки контроллера.

## Аргументы

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

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

Чтение информации о редукторе из EEPROM.

## Аргументы

|     |                  |                                              |
|-----|------------------|----------------------------------------------|
|     | id               | идентификатор устройства                     |
| out | gear_information | структура, содержащая информацию о редукторе |

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

Чтение настроек редуктора из EEPROM.

## Аргументы

|     |               |                                           |
|-----|---------------|-------------------------------------------|
|     | id            | идентификатор устройства                  |
| out | gear_settings | структура, содержащая настройки редуктора |

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

Чтение информации об датчиках Холла из EEPROM.

## Аргументы

|     |                        |                                                    |
|-----|------------------------|----------------------------------------------------|
|     | id                     | идентификатор устройства                           |
| out | hallsensor_information | структура, содержащая информацию об датчиках Холла |

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

Чтение настроек датчиков Холла из EEPROM.

## Аргументы

|     |                     |                                                |
|-----|---------------------|------------------------------------------------|
|     | id                  | идентификатор устройства                       |
| out | hallsensor_settings | структура, содержащая настройки датчиков Холла |

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

Команда чтения настроек для подхода в home position.

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

См. также

[home\\_settings\\_t](#)

Аргументы

|     | id            | идентификатор устройства     |
|-----|---------------|------------------------------|
| out | home_settings | настройки калибровки позиции |

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

Чтение настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует MaxSpeed *i*, где *i*=0, если предыдущим использованием этого режима не было выбрано другое *i*. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Расчёт DeadZone проиллюстрирован на графике: [!/attachments/download/5563/range25p.png!](#) Зависимость между отклонением и скоростью экспоненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность и точность. На графике ниже показан пример экспоненциальной зависимости скорости и работы мертвой зоны. [!/attachments/download/3092/ExpJoystick.png!](#) Параметр нелинейности можно менять. Нулевой параметр нелинейности соответствует линейной зависимости.

Аргументы

|     | id                | идентификатор устройства                  |
|-----|-------------------|-------------------------------------------|
| out | joystick_settings | структура, содержащая настройки джойстика |

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

Чтение информации о двигателе из EEPROM.

Аргументы

|     | id                | идентификатор устройства                     |
|-----|-------------------|----------------------------------------------|
| out | motor_information | структура, содержащая информацию о двигателе |

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

Чтение настроек двигателя из EEPROM.

Аргументы

|     |                |                                           |
|-----|----------------|-------------------------------------------|
|     | id             | идентификатор устройства                  |
| out | motor_settings | структура, содержащая настройки двигателя |

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

Команда чтения настроек перемещения (скорость, ускорение, threshold и скорость в режиме антилюфта).

Аргументы

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

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

Чтение ПИД коэффициентов.

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

См. также

[set\\_pid\\_settings](#)

Аргументы

|     |              |                          |
|-----|--------------|--------------------------|
|     | id           | идентификатор устройства |
| out | pid_settings | настройки ПИД            |

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

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

Аргументы

|     |          |                                                                       |
|-----|----------|-----------------------------------------------------------------------|
|     | id       | идентификатор устройства                                              |
| out | position | структура, содержащая настройки движения: скорость, ускорение, и т.д. |

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

Команда чтения параметров питания мотора.

Используется только с шаговым двигателем. Используется только с шаговым двигателем.

Аргументы

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

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

Команда записи установок защит.

Аргументы

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

См. также

status\_t::flags

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

Чтение серийного номера контроллера.

Аргументы

|     |              |                            |
|-----|--------------|----------------------------|
|     | id           | идентификатор устройства   |
| out | SerialNumber | серийный номер контроллера |

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

Чтение информации о позиционере из EEPROM.

Аргументы

|     |                   |                                                |
|-----|-------------------|------------------------------------------------|
|     | id                | идентификатор устройства                       |
| out | stage_information | структура, содержащая информацию о позиционере |

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

Чтение пользовательского имени подвижки из EEPROM.

Аргументы

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

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

Чтение настроек позиционера из EEPROM.

Аргументы

|     |                |                                             |
|-----|----------------|---------------------------------------------|
|     | id             | идентификатор устройства                    |
| out | stage_settings | структура, содержащая настройки позиционера |

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

Возвращает информацию о текущем состоянии устройства.

Аргументы

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

См. также

[get\\_status](#)

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

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

См. также

[get\\_status](#)

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

Чтение настроек для входного импульса синхронизации.

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

См. также

[set\\_sync\\_in\\_settings](#)

Аргументы

|     |                  |                          |
|-----|------------------|--------------------------|
|     | id               | идентификатор устройства |
| out | sync_in_settings | настройки синхронизации  |

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

Чтение настроек для выходного импульса синхронизации.

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

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

Команда чтения настроек UART.

Эта функция заполняет структуру настроек UART.

См. также

[uart\\_settings\\_t](#)

Аргументы

|     |               |                |
|-----|---------------|----------------|
|     | Speed         | Скорость UART  |
| out | uart_settings | настройки UART |

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

TODO Проверка наличия прошивки в контроллере

Аргументы

|     |     |                                     |
|-----|-----|-------------------------------------|
|     | id  | идентификатор устройства            |
| out | ret | не ноль, если прошивка присутствует |

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

Проверка наличия прошивки в контроллере

Аргументы

|     |      |                                     |
|-----|------|-------------------------------------|
|     | name | имя устройства                      |
| out | ret  | не ноль, если прошивка присутствует |

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

Простая функция логирования на stderr в узких (однобайтных) символах

Аргументы

|          |                     |
|----------|---------------------|
| loglevel | уровень логирования |
| message  | сообщение           |

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

Простая функция логирования на stderr в широких символах

Аргументы

|          |                     |
|----------|---------------------|
| loglevel | уровень логирования |
| message  | сообщение           |

5.1.4.69 `void XIMC_API msec_sleep ( unsigned int msec )`

Приостанавливает работу на указанное время



## Аргументы

|      |                       |
|------|-----------------------|
| msec | время в миллисекундах |
|------|-----------------------|

## 5.1.4.70 device\_t XIMC\_API open\_device ( const char \* name )

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

## Аргументы

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

## 5.1.4.71 result\_t XIMC\_API probe\_device ( const char \* name )

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

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

## Аргументы

|    |      |                  |
|----|------|------------------|
| in | name | - имя устройства |
|----|------|------------------|

## 5.1.4.72 result\_t XIMC\_API service\_command\_updf ( device\_t id )

Команда переводит контроллер в режим обновления прошивки.

Получив такую команду, прошивка платы устанавливает флаг (для загрузчика), отправляет эхо-ответ и перезагружает контроллер.

## 5.1.4.73 result\_t XIMC\_API set\_accessories\_settings ( device\_t id, const accessories\_settings\_t \* accessories\_settings )

Запись информации о дополнительных аксессуарах в EEPROM.

Функция должна использоваться только производителем.

## Аргументы

|    |                      |                                                               |
|----|----------------------|---------------------------------------------------------------|
|    | id                   | идентификатор устройства                                      |
| in | accessories_settings | структура, содержащая информацию о дополнительных аксессуарах |

## 5.1.4.74 result\_t XIMC\_API set\_add\_sync\_in\_action ( device\_t id, const add\_sync\_in\_action\_t \* add\_sync\_in\_action )

Это команда добавляет один элемент в буфер FIFO команд, выполняемых при получении входного импульса синхронизации.

Каждый импульс синхронизации либо выполнится то действие, которое описано в SSNI, если буфер пуст, либо самое старое из загруженных в буфер действий временно подменяет скорость и координату в SSNI. В последнем случае это действие стирается из буфера. Количество оставшихся пустыми элементов буфера можно узнать в структуре GETS.

## Аргументы

|  |    |                          |
|--|----|--------------------------|
|  | id | идентификатор устройства |
|--|----|--------------------------|

```
5.1.4.75 result_t XIMC_API set_brake_settings (device_t id, const brake_settings_t *
 brake_settings)
```

Запись настроек управления тормозом.

## Аргументы

|    |                |                                                     |
|----|----------------|-----------------------------------------------------|
|    | id             | идентификатор устройства                            |
| in | brake_settings | структура, содержащая настройки управления тормозом |

```
5.1.4.76 result_t XIMC_API set_control_settings (device_t id, const control_settings_t *
 control_settings)
```

Запись настроек управления мотором.

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

## Аргументы

|    |                  |                                                                                                 |
|----|------------------|-------------------------------------------------------------------------------------------------|
|    | id               | идентификатор устройства                                                                        |
| in | control_settings | структура, содержащая настройки управления мотором с помощью джойстика или кнопок влево/вправо. |

```
5.1.4.77 result_t XIMC_API set_controller_name (device_t id, const controller_name_t *
 controller_name)
```

Запись пользовательского имени контроллера и настроек в FRAM.

## Аргументы

|    |                        |                                                |
|----|------------------------|------------------------------------------------|
|    | id                     | идентификатор устройства                       |
| in | controller_information | структура, содержащая информацию о контроллере |

```
5.1.4.78 result_t XIMC_API set_ctp_settings (device_t id, const ctp_settings_t * ctp_settings)
```

Запись настроек контроля позиции(для шагового двигателя).

При управлении ШД с энкодером (CTP\_BASE 0) появляется возможность обнаруживать потерю шагов. Контроллер знает кол-во шагов на оборот (GENG::StepsPerRev) и разрешение энкодера (GFBS::IPT). При включении контроля (флаг CTP\_ENABLED), контроллер запоминает текущую позицию в шагах ШД и текущую позицию энкодера. Далее, на каждом шаге позиция энкодера преобразовывается в шаги и если разница оказывается больше CTPMinError, устанавливается флаг STATE\_CTP\_ERROR. При управлении ШД с датчиком оборотов (CTP\_BASE 1), позиция контролируется по нему. По активному фронту на входе синхронизации контроллер запоминает те-

кущее значение шагов. Далее, при каждом обороте проверяет, на сколько шагов сместились. При рассогласовании более CTPMinError устанавливается флаг STATE\_CTP\_ERROR.

Аргументы

|    |              |                                                  |
|----|--------------|--------------------------------------------------|
|    | id           | идентификатор устройства                         |
| in | ctp_settings | структура, содержащая настройки контроля позиции |

```
5.1.4.79 result_t XIMC_API set_edges_settings (device_t id, const edges_settings_t *
edges_settings)
```

Запись настроек границ и концевых выключателей.

См. также

[get\\_edges\\_settings](#)

Аргументы

|    |                |                                                                                                          |
|----|----------------|----------------------------------------------------------------------------------------------------------|
|    | id             | идентификатор устройства                                                                                 |
| in | edges_settings | настройки, определяющие тип границ, поведение мотора при их достижении и параметры концевых выключателей |

```
5.1.4.80 result_t XIMC_API set_encoder_information (device_t id, const encoder_information_t
* encoder_information)
```

Запись информации об энкодере в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |                     |                                              |
|----|---------------------|----------------------------------------------|
|    | id                  | идентификатор устройства                     |
| in | encoder_information | структура, содержащая информацию об энкодере |

```
5.1.4.81 result_t XIMC_API set_encoder_settings (device_t id, const encoder_settings_t *
encoder_settings)
```

Запись настроек энкодера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |                  |                                          |
|----|------------------|------------------------------------------|
|    | id               | идентификатор устройства                 |
| in | encoder_settings | структура, содержащая настройки энкодера |

```
5.1.4.82 result_t XIMC_API set_engine_settings (device_t id, const engine_settings_t *
engine_settings)
```

Запись настроек мотора.

Настройки определяют номинальные значения напряжения, тока, скорости мотора, характер дви-

жения и тип мотора. Пожалуйста, загружайте новые настройки когда вы меняете мотор, энкодер или позиционер. Помните, что неправильные настройки мотора могут повредить оборудование.

См. также

[get\\_engine\\_settings](#)

Аргументы

|    |                 |                                |
|----|-----------------|--------------------------------|
|    | id              | идентификатор устройства       |
| in | engine_settings | структура с настройками мотора |

5.1.4.83 `result_t XIMC_API set_entype_settings ( device_t id, const entype_settings_t * entype_settings )`

Запись информации о типе мотора и типе силового драйвера.

Аргументы

|    |            |                          |
|----|------------|--------------------------|
|    | id         | идентификатор устройства |
| in | EngineType | тип мотора               |
| in | DriverType | тип силового драйвера    |

5.1.4.84 `result_t XIMC_API set_extio_settings ( device_t id, const extio_settings_t * extio_settings )`

Команда записи параметров настройки режимов внешнего ввода/вывода.

Входные события обрабатываются по фронту. Выходные состояния сигнализируются логическим состоянием. По умолчанию нарастающий фронт считается моментом подачи входного сигнала, а единичное состояние считается активным выходом.

См. также

[get\\_extio\\_settings](#)

Аргументы

|    |                |                          |
|----|----------------|--------------------------|
|    | id             | идентификатор устройства |
| in | extio_settings | настройки EXTIO          |

5.1.4.85 `result_t XIMC_API set_feedback_settings ( device_t id, const feedback_settings_t * feedback_settings )`

Запись настроек обратной связи.

Аргументы

|    |               |                                                                       |
|----|---------------|-----------------------------------------------------------------------|
|    | id            | идентификатор устройства                                              |
| in | IPS           | Количество измеряемых отсчётов энкодера на оборот. Диапазон: 1..65535 |
| in | FeedbackType  | тип обратной связи                                                    |
| in | FeedbackFlags | флаги обратной связи                                                  |

```
5.1.4.86 result_t XIMC_API set_gear_information (device_t id, const gear_information_t *
 gear_information)
```

Запись информации о редукторе в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |                  |                                              |
|----|------------------|----------------------------------------------|
|    | id               | идентификатор устройства                     |
| in | gear_information | структура, содержащая информацию о редукторе |

```
5.1.4.87 result_t XIMC_API set_gear_settings (device_t id, const gear_settings_t * gear_settings
)
```

Запись настроек редуктора в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |               |                                           |
|----|---------------|-------------------------------------------|
|    | id            | идентификатор устройства                  |
| in | gear_settings | структура, содержащая настройки редуктора |

```
5.1.4.88 result_t XIMC_API set_hallsensor_information (device_t id, const
 hallsensor_information_t * hallsensor_information)
```

Запись информации об датчиках Холла в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |                        |                                                    |
|----|------------------------|----------------------------------------------------|
|    | id                     | идентификатор устройства                           |
| in | hallsensor_information | структура, содержащая информацию об датчиках Холла |

```
5.1.4.89 result_t XIMC_API set_hallsensor_settings (device_t id, const hallsensor_settings_t *
 hallsensor_settings)
```

Запись настроек датчиков Холла в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |                     |                                                |
|----|---------------------|------------------------------------------------|
|    | id                  | идентификатор устройства                       |
| in | hallsensor_settings | структура, содержащая настройки датчиков Холла |

```
5.1.4.90 result_t XIMC_API set_home_settings (device_t id, const home_settings_t *
 home_settings)
```

Команда записи настроек для подхода в home position.

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

См. также

[home\\_settings\\_t](#)

Аргументы

|     |               |                              |
|-----|---------------|------------------------------|
|     | id            | идентификатор устройства     |
| out | home_settings | настройки калибровки позиции |

5.1.4.91 `result_t XIMC_API set_joystick_settings ( device_t id, const joystick_settings_t * joystick_settings )`

Запись настроек джойстика.

При отклонении джойстика более чем на DeadZone от центрального положения начинается движение со скоростью, определяемой отклонением джойстика от DeadZone до 100% отклонения, причем отклонению DeadZone соответствует нулевая скорость, а 100% отклонения соответствует MaxSpeed *i*, где *i*=0, если предыдущим использованием этого режима не было выбрано другое *i*. Если следующая скорость в таблице скоростей нулевая (целая и микрошаговая части), то перехода на неё не происходит. DeadZone вычисляется в десятых долях процента отклонения от центра (JoyCenter) до правого или левого максимума. Расчёт DeadZone проиллюстрирован на графике: [!/attachments/download/5563/range25p.png!](#) Зависимость между отклонением и скоростью экспоненциальная, что позволяет без переключения режимов скорости сочетать высокую подвижность и точность. На графике ниже показан пример экспоненциальной зависимости скорости и работы мертвой зоны. [!/attachments/download/3092/ExpJoystick.png!](#) Параметр нелинейности можно менять. Нулевой параметр нелинейности соответствует линейной зависимости.

Аргументы

|    |                   |                                           |
|----|-------------------|-------------------------------------------|
|    | id                | идентификатор устройства                  |
| in | joystick_settings | структура, содержащая настройки джойстика |

5.1.4.92 `void XIMC_API set_logging_callback ( logging_callback_t logging_callback )`

Устанавливает функцию обратного вызова для логирования.

Вызов назначает стандартный логгер (stderr, syslog), если передан NULL

Аргументы

|                  |                                       |
|------------------|---------------------------------------|
| logging_callback | указатель на функцию обратного вызова |
|------------------|---------------------------------------|

5.1.4.93 `result_t XIMC_API set_motor_information ( device_t id, const motor_information_t * motor_information )`

Запись информации о двигателе в EEPROM.

Функция должна использоваться только производителем.

## Аргументы

|    |                       |                                              |
|----|-----------------------|----------------------------------------------|
|    | id                    | идентификатор устройства                     |
| in | motor_<br>information | структура, содержащая информацию о двигателе |

5.1.4.94 `result_t XIMC_API set_motor_settings ( device_t id, const motor_settings_t * motor_settings )`

Запись настроек двигателя в EEPROM.

Функция должна использоваться только производителем.

## Аргументы

|    |                |                                           |
|----|----------------|-------------------------------------------|
|    | id             | идентификатор устройства                  |
| in | motor_settings | структура, содержащая настройки двигателя |

5.1.4.95 `result_t XIMC_API set_move_settings ( device_t id, const move_settings_t * move_settings )`

Команда записи настроек перемещения (скорость, ускорение, threshold и скорость в режиме анти-люфта).

## Аргументы

|    |               |                                                                       |
|----|---------------|-----------------------------------------------------------------------|
|    | id            | идентификатор устройства                                              |
| in | move_settings | структура, содержащая настройки движения: скорость, ускорение, и т.д. |

5.1.4.96 `result_t XIMC_API set_pid_settings ( device_t id, const pid_settings_t * pid_settings )`

Запись ПИД коэффициентов.

Эти коэффициенты определяют поведение напряжения. Коэффициенты различны для разных позиционеров. Пожалуйста, загружайте новые настройки, когда вы меняете мотор или позиционер.

См. также

[get\\_pid\\_settings](#)

## Аргументы

|    |              |                          |
|----|--------------|--------------------------|
|    | id           | идентификатор устройства |
| in | pid_settings | настройки ПИД            |

5.1.4.97 `result_t XIMC_API set_position ( device_t id, const set_position_t * the_set_position )`

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

То есть меняется основной показатель положения.

## Аргументы

|     |          |                                                                       |
|-----|----------|-----------------------------------------------------------------------|
|     | id       | идентификатор устройства                                              |
| out | position | структура, содержащая настройки движения: скорость, ускорение, и т.д. |

5.1.4.98 `result_t XIMC_API set_power_settings ( device_t id, const power_settings_t * power_settings )`

Команда записи параметров питания мотора.

Используется только с шаговым двигателем.

## Аргументы

|    |                |                                                         |
|----|----------------|---------------------------------------------------------|
|    | id             | идентификатор устройства                                |
| in | power_settings | структура, содержащая настройки питания шагового мотора |

5.1.4.99 `result_t XIMC_API set_secure_settings ( device_t id, const secure_settings_t * secure_settings )`

Команда записи установок защит.

## Аргументы

|                 |    |                                              |
|-----------------|----|----------------------------------------------|
|                 | id | идентификатор устройства                     |
| secure_settings |    | структура с настройками критических значений |

См. также

`status_t::flags`

5.1.4.100 `result_t XIMC_API set_serial_number ( device_t id, const serial_number_t * serial_number )`

Запись серийного номера во flash память контроллера.

Вместе с новым серийным номером передаётся "Ключ", только при совпадении которого происходит изменение и сохранение серийного номера. Функция используется только производителем.

## Аргументы

|    |        |                                                     |
|----|--------|-----------------------------------------------------|
|    | id     | идентификатор устройства                            |
| in | serial | number структура, содержащая серийный номер и ключ. |

5.1.4.101 `result_t XIMC_API set_stage_information ( device_t id, const stage_information_t * stage_information )`

Запись информации о позиционере в EEPROM.

Функция должна использоваться только производителем.

## Аргументы

|  |    |                          |
|--|----|--------------------------|
|  | id | идентификатор устройства |
|--|----|--------------------------|



|    |                         |                                                |
|----|-------------------------|------------------------------------------------|
| in | stage_ -<br>information | структура, содержащая информацию о позиционере |
|----|-------------------------|------------------------------------------------|

5.1.4.102 `result_t XIMC_API set_stage_name ( device_t id, const stage_name_t * stage_name )`

Запись пользовательского имени подвижки в EEPROM.

Аргументы

|    |            |                                                                      |
|----|------------|----------------------------------------------------------------------|
|    | id         | идентификатор устройства                                             |
| in | stage_name | структура, содержащая установленное пользовательское имя позиционера |

5.1.4.103 `result_t XIMC_API set_stage_settings ( device_t id, const stage_settings_t * stage_settings )`

Запись настроек позиционера в EEPROM.

Функция должна использоваться только производителем.

Аргументы

|    |                |                                             |
|----|----------------|---------------------------------------------|
|    | id             | идентификатор устройства                    |
| in | stage_settings | структура, содержащая настройки позиционера |

5.1.4.104 `result_t XIMC_API set_sync_in_settings ( device_t id, const sync_in_settings_t * sync_in_settings )`

Запись настроек для входного импульса синхронизации.

Эта функция записывает структуру с настройками входного импульса синхронизации, определяющими поведение входа синхронизации, в память контроллера.

См. также

[get\\_sync\\_in\\_settings](#)

Аргументы

|    |                        |                          |
|----|------------------------|--------------------------|
|    | id                     | идентификатор устройства |
| in | sync_in_ -<br>settings | настройки синхронизации  |

5.1.4.105 `result_t XIMC_API set_sync_out_settings ( device_t id, const sync_out_settings_t * sync_out_settings )`

Запись настроек для выходного импульса синхронизации.

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

См. также

[get\\_sync\\_in\\_settings](#)

Аргументы

|    |                  |                          |
|----|------------------|--------------------------|
|    | id               | идентификатор устройства |
| in | sync_in_settings | настройки синхронизации  |

5.1.4.106 `result_t XIMC_API set_uart_settings ( device_t id, const uart_settings_t * uart_settings )`

Команда записи настроек UART.

Эта функция записывает структуру настроек UART в память контроллера.

См. также

[uart\\_settings\\_t](#)

Аргументы

|    |               |                |
|----|---------------|----------------|
|    | Speed         | Скорость UART  |
| in | uart_settings | настройки UART |

5.1.4.107 `result_t XIMC_API write_key ( const char * name, uint8_t * key )`

Запись ключа защиты TODO fix docs Функция используется только производителем.

Аргументы

|    |      |                                      |
|----|------|--------------------------------------|
|    | name | имя устройства                       |
| in | key  | ключ защиты. Диапазон: 0..4294967295 |

5.1.4.108 `result_t XIMC_API ximc_fix_usbser_sys ( const char * device_name )`

Исправление ошибки драйвера USB в Windows.

Перезагружает драйвер, если устройство существует, но в зависшем состоянии.

5.1.4.109 `void XIMC_API ximc_version ( char * version )`

Возвращает версию библиотеки

Аргументы

|         |                                                |
|---------|------------------------------------------------|
| version | буфер для строки с версией, 32 байт достаточно |
|---------|------------------------------------------------|

# Предметный указатель

Accel  
    move\_settings\_t, 37  
accessories\_settings\_t, 8  
    MBRatedCurrent, 9  
    MBRatedVoltage, 9  
    MBTorque, 9  
    TSGrad, 9  
    TSMIn, 9  
Accuracy  
    sync\_out\_settings\_t, 50  
add\_sync\_in\_action\_calb\_t, 9  
add\_sync\_in\_action\_t, 9  
    Speed, 10  
    uSpeed, 10  
analog\_data\_t, 10  
Antiplay  
    engine\_settings\_calb\_t, 21  
    engine\_settings\_t, 22  
AntiplaySpeed  
    move\_settings\_t, 37  
  
BORDER\_IS\_ENCODER  
    ximc.h, 74  
brake\_settings\_t, 11  
    t1, 12  
    t2, 12  
    t3, 12  
    t4, 12  
  
CTPMinError  
    ctp\_settings\_t, 16  
calibration\_t, 12  
chart\_data\_t, 13  
close\_device  
    ximc.h, 77  
ClutterTime  
    sync\_in\_settings\_calb\_t, 47  
    sync\_in\_settings\_t, 48  
command\_clear\_fram  
    ximc.h, 77  
command\_eeread\_settings  
    ximc.h, 77  
command\_eesave\_settings  
    ximc.h, 77  
command\_home  
    ximc.h, 78  
command\_left  
    ximc.h, 78  
command\_loft  
    ximc.h, 78  
command\_move  
    ximc.h, 78  
command\_movr  
    ximc.h, 79  
command\_power\_off  
    ximc.h, 79  
command\_read\_settings  
    ximc.h, 79  
command\_reset  
    ximc.h, 79  
command\_right  
    ximc.h, 80  
command\_save\_settings  
    ximc.h, 80  
command\_sstp  
    ximc.h, 80  
command\_stop  
    ximc.h, 80  
command\_update\_firmware  
    ximc.h, 80  
command\_zero  
    ximc.h, 81  
control\_settings\_calb\_t, 14  
    MaxClickTime, 14  
    Timeout, 14  
control\_settings\_t, 14  
    MaxClickTime, 15  
    MaxSpeed, 15  
    Timeout, 15  
    uMaxSpeed, 15  
controller\_name\_t, 15  
    ControllerName, 16  
ControllerName  
    controller\_name\_t, 16  
CriticalIpwr  
    secure\_settings\_t, 40  
CriticalIusb  
    secure\_settings\_t, 40  
CriticalT  
    secure\_settings\_t, 40  
CriticalUpwr  
    secure\_settings\_t, 40  
CriticalUusb  
    secure\_settings\_t, 40  
ctp\_settings\_t, 16  
    CTPMinError, 16  
CurPosition  
    status\_calb\_t, 45

- status\_t, 47
- CurrReductDelay
  - power\_settings\_t, 39
- CurrentSetTime
  - power\_settings\_t, 39
- DeadZone
  - joystick\_settings\_t, 32
- debug\_read\_t, 17
- Decel
  - move\_settings\_t, 37
- DetentTorque
  - motor\_settings\_t, 34
- device\_information\_t, 17
- ENGINE\_ACCEL\_ON
  - ximc.h, 74
- ENGINE\_ANTIPLAY
  - ximc.h, 74
- ENGINE\_LIMIT\_CURR
  - ximc.h, 75
- ENGINE\_LIMIT\_RPM
  - ximc.h, 75
- ENGINE\_LIMIT\_VOLT
  - ximc.h, 75
- ENGINE\_MAX\_SPEED
  - ximc.h, 75
- ENGINE\_REVERSE
  - ximc.h, 75
- ENUMERATE\_PROBE
  - ximc.h, 75
- edges\_settings\_calb\_t, 17
- edges\_settings\_t, 18
  - LeftBorder, 18
  - RightBorder, 18
  - uLeftBorder, 18
  - uRightBorder, 19
- Efficiency
  - gear\_settings\_t, 26
- encoder\_information\_t, 19
  - Manufacturer, 19
  - PartNumber, 19
- encoder\_settings\_t, 19
  - MaxCurrentConsumption, 20
  - MaxOperatingFrequency, 20
  - SupplyVoltageMax, 20
  - SupplyVoltageMin, 20
- engine\_settings\_calb\_t, 21
  - Antiplay, 21
  - NomCurrent, 21
  - NomSpeed, 21
  - NomVoltage, 21
  - StepsPerRev, 21
- engine\_settings\_t, 22
  - Antiplay, 22
  - NomCurrent, 22
  - NomSpeed, 22
  - NomVoltage, 23
  - StepsPerRev, 23
- uNomSpeed, 23
- entype\_settings\_t, 23
- enumerate\_devices
  - ximc.h, 81
- extio\_settings\_t, 24
- FastHome
  - home\_settings\_t, 30
- feedback\_settings\_t, 24
- free\_enumerate\_devices
  - ximc.h, 81
- gear\_information\_t, 25
  - Manufacturer, 25
  - PartNumber, 25
- gear\_settings\_t, 25
  - Efficiency, 26
  - InputInertia, 26
  - MaxOutputBacklash, 26
  - RatedInputSpeed, 26
  - RatedInputTorque, 26
  - ReductionIn, 26
  - ReductionOut, 27
- get\_accessories\_settings
  - ximc.h, 81
- get\_analog\_data
  - ximc.h, 81
- get\_bootloader\_version
  - ximc.h, 82
- get\_brake\_settings
  - ximc.h, 82
- get\_chart\_data
  - ximc.h, 82
- get\_control\_settings
  - ximc.h, 82
- get\_controller\_name
  - ximc.h, 83
- get\_ctp\_settings
  - ximc.h, 83
- get\_debug\_read
  - ximc.h, 83
- get\_device\_count
  - ximc.h, 83
- get\_device\_information
  - ximc.h, 84
- get\_device\_name
  - ximc.h, 84
- get\_edges\_settings
  - ximc.h, 84
- get\_encoder\_information
  - ximc.h, 84
- get\_encoder\_settings
  - ximc.h, 85
- get\_engine\_settings
  - ximc.h, 85
- get\_entype\_settings
  - ximc.h, 85
- get\_enumerate\_device\_information
  - ximc.h, 85

- get\_enumerate\_device\_serial
  - ximc.h, 86
- get\_extio\_settings
  - ximc.h, 86
- get\_feedback\_settings
  - ximc.h, 86
- get\_firmware\_version
  - ximc.h, 86
- get\_gear\_information
  - ximc.h, 87
- get\_gear\_settings
  - ximc.h, 87
- get\_hallsensor\_information
  - ximc.h, 87
- get\_hallsensor\_settings
  - ximc.h, 87
- get\_home\_settings
  - ximc.h, 87
- get\_joystick\_settings
  - ximc.h, 88
- get\_motor\_information
  - ximc.h, 88
- get\_motor\_settings
  - ximc.h, 88
- get\_move\_settings
  - ximc.h, 89
- get\_pid\_settings
  - ximc.h, 89
- get\_position
  - ximc.h, 89
- get\_position\_calb\_t, 27
- get\_position\_t, 27
- get\_power\_settings
  - ximc.h, 89
- get\_secure\_settings
  - ximc.h, 89
- get\_serial\_number
  - ximc.h, 90
- get\_stage\_information
  - ximc.h, 90
- get\_stage\_name
  - ximc.h, 90
- get\_stage\_settings
  - ximc.h, 90
- get\_status
  - ximc.h, 90
- get\_status\_calb
  - ximc.h, 91
- get\_sync\_in\_settings
  - ximc.h, 91
- get\_sync\_out\_settings
  - ximc.h, 91
- get\_uart\_settings
  - ximc.h, 91
- goto\_firmware
  - ximc.h, 92
- HOME\_DIR\_FIRST
  - ximc.h, 75
- HOME\_DIR\_SECOND
  - ximc.h, 75
- hallsensor\_information\_t, 27
  - Manufacturer, 28
  - PartNumber, 28
- hallsensor\_settings\_t, 28
  - MaxCurrentConsumption, 29
  - MaxOperatingFrequency, 29
  - SupplyVoltageMax, 29
  - SupplyVoltageMin, 29
- has\_firmware
  - ximc.h, 92
- HoldCurrent
  - power\_settings\_t, 39
- home\_settings\_calb\_t, 29
- home\_settings\_t, 29
  - FastHome, 30
  - HomeDelta, 30
  - SlowHome, 30
  - uFastHome, 30
  - uHomeDelta, 30
  - uSlowHome, 31
- HomeDelta
  - home\_settings\_t, 30
- HorizontalLoadCapacity
  - stage\_settings\_t, 44
- InputInertia
  - gear\_settings\_t, 26
- JOY\_REVERSE
  - ximc.h, 75
- JoyCenter
  - joystick\_settings\_t, 32
- JoyHighEnd
  - joystick\_settings\_t, 32
- JoyLowEnd
  - joystick\_settings\_t, 32
- joystick\_settings\_t, 31
  - DeadZone, 32
  - JoyCenter, 32
  - JoyHighEnd, 32
  - JoyLowEnd, 32
- LeadScrewPitch
  - stage\_settings\_t, 44
- LeftBorder
  - edges\_settings\_t, 18
- logging\_callback\_stderr\_narrow
  - ximc.h, 92
- logging\_callback\_stderr\_wide
  - ximc.h, 92
- logging\_callback\_t
  - ximc.h, 77
- LowUpwrOff
  - secure\_settings\_t, 40
- MBRatedCurrent
  - accessories\_settings\_t, 9

- MBRatedVoltage
  - accessories\_settings\_t, 9
- MBTorque
  - accessories\_settings\_t, 9
- MVCMD\_ERROR
  - ximc.h, 76
- Manufacturer
  - encoder\_information\_t, 19
  - gear\_information\_t, 25
  - hallsensor\_information\_t, 28
  - motor\_information\_t, 32
  - stage\_information\_t, 42
- MaxClickTime
  - control\_settings\_calb\_t, 14
  - control\_settings\_t, 15
- MaxCurrent
  - motor\_settings\_t, 34
- MaxCurrentConsumption
  - encoder\_settings\_t, 20
  - hallsensor\_settings\_t, 29
  - stage\_settings\_t, 44
- MaxCurrentTime
  - motor\_settings\_t, 34
- MaxOperatingFrequency
  - encoder\_settings\_t, 20
  - hallsensor\_settings\_t, 29
- MaxOutputBacklash
  - gear\_settings\_t, 26
- MaxSpeed
  - control\_settings\_t, 15
  - motor\_settings\_t, 34
  - stage\_settings\_t, 44
- MechanicalTimeConstant
  - motor\_settings\_t, 34
- MinimumUusb
  - secure\_settings\_t, 40
- motor\_information\_t, 32
  - Manufacturer, 32
  - PartNumber, 32
- motor\_settings\_t, 33
  - DetentTorque, 34
  - MaxCurrent, 34
  - MaxCurrentTime, 34
  - MaxSpeed, 34
  - MechanicalTimeConstant, 34
  - NoLoadCurrent, 34
  - NoLoadSpeed, 35
  - NominalCurrent, 35
  - NominalPower, 35
  - NominalSpeed, 35
  - NominalTorque, 35
  - NominalVoltage, 35
  - RotorInertia, 35
  - SpeedConstant, 35
  - SpeedTorqueGradient, 35
  - StallTorque, 36
  - TorqueConstant, 36
  - WindingInductance, 36
  - WindingResistance, 36
- move\_settings\_calb\_t, 36
- move\_settings\_t, 36
  - Accel, 37
  - AntiplaySpeed, 37
  - Decel, 37
  - Speed, 37
  - uAntiplaySpeed, 37
  - uSpeed, 37
- msec\_sleep
  - ximc.h, 92
- NoLoadCurrent
  - motor\_settings\_t, 34
- NoLoadSpeed
  - motor\_settings\_t, 35
- NomCurrent
  - engine\_settings\_calb\_t, 21
  - engine\_settings\_t, 22
- NomSpeed
  - engine\_settings\_calb\_t, 21
  - engine\_settings\_t, 22
- NomVoltage
  - engine\_settings\_calb\_t, 21
  - engine\_settings\_t, 23
- NominalCurrent
  - motor\_settings\_t, 35
- NominalPower
  - motor\_settings\_t, 35
- NominalSpeed
  - motor\_settings\_t, 35
- NominalTorque
  - motor\_settings\_t, 35
- NominalVoltage
  - motor\_settings\_t, 35
- open\_device
  - ximc.h, 93
- POWER\_OFF\_ENABLED
  - ximc.h, 76
- POWER\_REDUCT\_ENABLED
  - ximc.h, 76
- PartNumber
  - encoder\_information\_t, 19
  - gear\_information\_t, 25
  - hallsensor\_information\_t, 28
  - motor\_information\_t, 32
  - stage\_information\_t, 42
- pid\_settings\_t, 38
- PositionerName
  - stage\_name\_t, 43
- power\_settings\_t, 38
  - CurrReductDelay, 39
  - CurrentSetTime, 39
  - HoldCurrent, 39
  - PowerOffDelay, 39
- PowerOffDelay
  - power\_settings\_t, 39

- probe\_device
  - ximc.h, [93](#)
- REV\_SENS\_INV
  - ximc.h, [76](#)
- RatedInputSpeed
  - gear\_settings\_t, [26](#)
- RatedInputTorque
  - gear\_settings\_t, [26](#)
- ReductionIn
  - gear\_settings\_t, [26](#)
- ReductionOut
  - gear\_settings\_t, [27](#)
- RightBorder
  - edges\_settings\_t, [18](#)
- RotorInertia
  - motor\_settings\_t, [35](#)
- STATE\_ALARM
  - ximc.h, [76](#)
- SYNCIN\_INVERT
  - ximc.h, [76](#)
- SYNCOUT\_ENABLED
  - ximc.h, [76](#)
- SYNCOUT\_STATE
  - ximc.h, [76](#)
- secure\_settings\_t, [39](#)
  - CriticalIpwr, [40](#)
  - CriticalIusb, [40](#)
  - CriticalT, [40](#)
  - CriticalUpwr, [40](#)
  - CriticalUusb, [40](#)
  - LowUpwrOff, [40](#)
  - MinimumUusb, [40](#)
- serial\_number\_t, [41](#)
- service\_command\_updf
  - ximc.h, [93](#)
- set\_accessories\_settings
  - ximc.h, [93](#)
- set\_add\_sync\_in\_action
  - ximc.h, [93](#)
- set\_brake\_settings
  - ximc.h, [94](#)
- set\_control\_settings
  - ximc.h, [94](#)
- set\_controller\_name
  - ximc.h, [94](#)
- set\_ctp\_settings
  - ximc.h, [94](#)
- set\_edges\_settings
  - ximc.h, [95](#)
- set\_encoder\_information
  - ximc.h, [95](#)
- set\_encoder\_settings
  - ximc.h, [95](#)
- set\_engine\_settings
  - ximc.h, [95](#)
- set\_entype\_settings
  - ximc.h, [96](#)
- set\_extio\_settings
  - ximc.h, [96](#)
- set\_feedback\_settings
  - ximc.h, [96](#)
- set\_gear\_information
  - ximc.h, [96](#)
- set\_gear\_settings
  - ximc.h, [97](#)
- set\_hallsensor\_information
  - ximc.h, [97](#)
- set\_hallsensor\_settings
  - ximc.h, [97](#)
- set\_home\_settings
  - ximc.h, [97](#)
- set\_joystick\_settings
  - ximc.h, [98](#)
- set\_logging\_callback
  - ximc.h, [98](#)
- set\_motor\_information
  - ximc.h, [98](#)
- set\_motor\_settings
  - ximc.h, [99](#)
- set\_move\_settings
  - ximc.h, [99](#)
- set\_pid\_settings
  - ximc.h, [99](#)
- set\_position
  - ximc.h, [99](#)
- set\_position\_calb\_t, [41](#)
- set\_position\_t, [41](#)
- set\_power\_settings
  - ximc.h, [100](#)
- set\_secure\_settings
  - ximc.h, [100](#)
- set\_serial\_number
  - ximc.h, [100](#)
- set\_stage\_information
  - ximc.h, [100](#)
- set\_stage\_name
  - ximc.h, [101](#)
- set\_stage\_settings
  - ximc.h, [101](#)
- set\_sync\_in\_settings
  - ximc.h, [101](#)
- set\_sync\_out\_settings
  - ximc.h, [101](#)
- set\_uart\_settings
  - ximc.h, [102](#)
- SlowHome
  - home\_settings\_t, [30](#)
- Speed
  - add\_sync\_in\_action\_t, [10](#)
  - move\_settings\_t, [37](#)
  - sync\_in\_settings\_t, [48](#)
- SpeedConstant
  - motor\_settings\_t, [35](#)
- SpeedTorqueGradient
  - motor\_settings\_t, [35](#)

- stage\_information\_t, [42](#)
  - Manufacturer, [42](#)
  - PartNumber, [42](#)
- stage\_name\_t, [42](#)
  - PositionerName, [43](#)
- stage\_settings\_t, [43](#)
  - HorizontalLoadCapacity, [44](#)
  - LeadScrewPitch, [44](#)
  - MaxCurrentConsumption, [44](#)
  - MaxSpeed, [44](#)
  - SupplyVoltageMax, [44](#)
  - SupplyVoltageMin, [44](#)
  - TravelRange, [44](#)
  - VerticalLoadCapacity, [44](#)
- StallTorque
  - motor\_settings\_t, [36](#)
- status\_calb\_t, [45](#)
  - CurPosition, [45](#)
- status\_t, [46](#)
  - CurPosition, [47](#)
  - uCurPosition, [47](#)
  - uCurSpeed, [47](#)
- StepsPerRev
  - engine\_settings\_calb\_t, [21](#)
  - engine\_settings\_t, [23](#)
- SupplyVoltageMax
  - encoder\_settings\_t, [20](#)
  - hallsensor\_settings\_t, [29](#)
  - stage\_settings\_t, [44](#)
- SupplyVoltageMin
  - encoder\_settings\_t, [20](#)
  - hallsensor\_settings\_t, [29](#)
  - stage\_settings\_t, [44](#)
- sync\_in\_settings\_calb\_t, [47](#)
  - ClutterTime, [47](#)
- sync\_in\_settings\_t, [48](#)
  - ClutterTime, [48](#)
  - Speed, [48](#)
  - uSpeed, [48](#)
- sync\_out\_settings\_calb\_t, [49](#)
  - SyncOutPeriod, [49](#)
  - SyncOutPulseSteps, [49](#)
- sync\_out\_settings\_t, [49](#)
  - Accuracy, [50](#)
  - SyncOutPeriod, [50](#)
  - SyncOutPulseSteps, [50](#)
  - uAccuracy, [50](#)
- SyncOutPeriod
  - sync\_out\_settings\_calb\_t, [49](#)
  - sync\_out\_settings\_t, [50](#)
- SyncOutPulseSteps
  - sync\_out\_settings\_calb\_t, [49](#)
  - sync\_out\_settings\_t, [50](#)
- t1
  - brake\_settings\_t, [12](#)
- t2
  - brake\_settings\_t, [12](#)
- t3
  - brake\_settings\_t, [12](#)
- t4
  - brake\_settings\_t, [12](#)
- TSGrad
  - accessories\_settings\_t, [9](#)
- TSMIn
  - accessories\_settings\_t, [9](#)
- Timeout
  - control\_settings\_calb\_t, [14](#)
  - control\_settings\_t, [15](#)
- TorqueConstant
  - motor\_settings\_t, [36](#)
- TravelRange
  - stage\_settings\_t, [44](#)
- uAccuracy
  - sync\_out\_settings\_t, [50](#)
- uAntiplaySpeed
  - move\_settings\_t, [37](#)
- uCurPosition
  - status\_t, [47](#)
- uCurSpeed
  - status\_t, [47](#)
- uFastHome
  - home\_settings\_t, [30](#)
- uHomeDelta
  - home\_settings\_t, [30](#)
- uLeftBorder
  - edges\_settings\_t, [18](#)
- uMaxSpeed
  - control\_settings\_t, [15](#)
- uNomSpeed
  - engine\_settings\_t, [23](#)
- uRightBorder
  - edges\_settings\_t, [19](#)
- uSlowHome
  - home\_settings\_t, [31](#)
- uSpeed
  - add\_sync\_in\_action\_t, [10](#)
  - move\_settings\_t, [37](#)
  - sync\_in\_settings\_t, [48](#)
- uart\_settings\_t, [50](#)
- VerticalLoadCapacity
  - stage\_settings\_t, [44](#)
- WindingInductance
  - motor\_settings\_t, [36](#)
- WindingResistance
  - motor\_settings\_t, [36](#)
- write\_key
  - ximc.h, [102](#)
- XIMC\_API
  - ximc.h, [76](#)
- ximc.h, [52](#)
  - BORDER\_IS\_ENCODER, [74](#)
  - close\_device, [77](#)
  - command\_clear\_fram, [77](#)



- command\_eeread\_settings, 77
- command\_eesave\_settings, 77
- command\_home, 78
- command\_left, 78
- command\_loft, 78
- command\_move, 78
- command\_movr, 79
- command\_power\_off, 79
- command\_read\_settings, 79
- command\_reset, 79
- command\_right, 80
- command\_save\_settings, 80
- command\_sstp, 80
- command\_stop, 80
- command\_update\_firmware, 80
- command\_zero, 81
- ENGINE\_ACCEL\_ON, 74
- ENGINE\_ANTIPLAY, 74
- ENGINE\_LIMIT\_CURR, 75
- ENGINE\_LIMIT\_RPM, 75
- ENGINE\_LIMIT\_VOLT, 75
- ENGINE\_MAX\_SPEED, 75
- ENGINE\_REVERSE, 75
- ENUMERATE\_PROBE, 75
- enumerate\_devices, 81
- free\_enumerate\_devices, 81
- get\_accessories\_settings, 81
- get\_analog\_data, 81
- get\_bootloader\_version, 82
- get\_brake\_settings, 82
- get\_chart\_data, 82
- get\_control\_settings, 82
- get\_controller\_name, 83
- get\_ctp\_settings, 83
- get\_debug\_read, 83
- get\_device\_count, 83
- get\_device\_information, 84
- get\_device\_name, 84
- get\_edges\_settings, 84
- get\_encoder\_information, 84
- get\_encoder\_settings, 85
- get\_engine\_settings, 85
- get\_entype\_settings, 85
- get\_enumerate\_device\_information, 85
- get\_enumerate\_device\_serial, 86
- get\_extio\_settings, 86
- get\_feedback\_settings, 86
- get\_firmware\_version, 86
- get\_gear\_information, 87
- get\_gear\_settings, 87
- get\_hallsensor\_information, 87
- get\_hallsensor\_settings, 87
- get\_home\_settings, 87
- get\_joystick\_settings, 88
- get\_motor\_information, 88
- get\_motor\_settings, 88
- get\_move\_settings, 89
- get\_pid\_settings, 89
- get\_position, 89
- get\_power\_settings, 89
- get\_secure\_settings, 89
- get\_serial\_number, 90
- get\_stage\_information, 90
- get\_stage\_name, 90
- get\_stage\_settings, 90
- get\_status, 90
- get\_status\_calb, 91
- get\_sync\_in\_settings, 91
- get\_sync\_out\_settings, 91
- get\_uart\_settings, 91
- goto\_firmware, 92
- HOME\_DIR\_FIRST, 75
- HOME\_DIR\_SECOND, 75
- has\_firmware, 92
- JOY\_REVERSE, 75
- logging\_callback\_stderr\_narrow, 92
- logging\_callback\_stderr\_wide, 92
- logging\_callback\_t, 77
- MVCMD\_ERROR, 76
- msec\_sleep, 92
- open\_device, 93
- POWER\_OFF\_ENABLED, 76
- probe\_device, 93
- REV\_SENS\_INV, 76
- STATE\_ALARM, 76
- SYNCIN\_INVERT, 76
- SYNCOUT\_ENABLED, 76
- SYNCOUT\_STATE, 76
- service\_command\_updf, 93
- set\_accessories\_settings, 93
- set\_add\_sync\_in\_action, 93
- set\_brake\_settings, 94
- set\_control\_settings, 94
- set\_controller\_name, 94
- set\_ctp\_settings, 94
- set\_edges\_settings, 95
- set\_encoder\_information, 95
- set\_encoder\_settings, 95
- set\_engine\_settings, 95
- set\_entype\_settings, 96
- set\_extio\_settings, 96
- set\_feedback\_settings, 96
- set\_gear\_information, 96
- set\_gear\_settings, 97
- set\_hallsensor\_information, 97
- set\_hallsensor\_settings, 97
- set\_home\_settings, 97
- set\_joystick\_settings, 98
- set\_logging\_callback, 98
- set\_motor\_information, 98
- set\_motor\_settings, 99
- set\_move\_settings, 99
- set\_pid\_settings, 99
- set\_position, 99
- set\_power\_settings, 100
- set\_secure\_settings, 100

- set\_serial\_number, [100](#)
- set\_stage\_information, [100](#)
- set\_stage\_name, [101](#)
- set\_stage\_settings, [101](#)
- set\_sync\_in\_settings, [101](#)
- set\_sync\_out\_settings, [101](#)
- set\_uart\_settings, [102](#)
- write\_key, [102](#)
- XIMC\_API, [76](#)
- ximc\_fix\_usbser\_sys, [102](#)
- ximc\_version, [102](#)
- ximc\_fix\_usbser\_sys
  - ximc.h, [102](#)
- ximc\_version
  - ximc.h, [102](#)